Gould V6 and V9 Central Processing Unit

Reference Manual

October 1985

Publication Order Number: 301-003410-000

Supersedes: 301-003280-000

GOULD
Electronics

This manual is supplied without representation or warranty of any kind. Gould Inc., Computer Systems Division therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

HISTORY

The Gould V6 and V9 Central Processing Unit Reference Manual, Publication Order Number **301-003410-000**, was printed November, 1984.

Publication Order Number **301-003410-001** (Change Package 1) was printed March, 1985.

Publication Order Number **301-003410-002** (Change Package 2) was printed October 1985. The updated manual contains the following pages:

	*Change Number
Title page	2
Copyright page	0
History page Change 2	2
History page Change 1	1
History page	0
iii through ix(x Blank)	0
1-1 through 1-4	0
2-1 through 2-13	0
2-14 and 2-15	2
2-16 through 2-18	0
3-1 through 3-3	0
3-4 through 3-6	2
4-1 through 4-13	0
4-14	2
4-15 through 4-27	0
4-28 and 4-29	2
4-30	0
5-1 through 5-24	0
6-1 through 6-112	0
6-112A	2
6-112B through 6-112C	0
6-112D	2
6-112E through 6-129	0
6-130	1
6-131 through 6-133	0
6-134	1
6-135 through 6-241	0
6-242	1
6-243 through 6-424	0
6-425 and 6-426	1
A-1 through A-7(A-8 Blank)	0
B-1 through B-7(B-8 Blank)	0
C-1 through C-6	0
D-1(D-2 Blank)	0

* Zero in this column indicates an original page.

On a change page, the portion of the page affected by the latest change is indicated by a vertical bar in the outer margin of the page. However, a completely changed page will not have a full length bar, but will have the change notation by the page number.

HISTORY

The Gould V6 and V9 Central Processing Unit Reference Manual, Publication Order Number **301-003410-000,** was printed November, 1984.

Publication Order Number **301-003410-001** (Change Package 1) was printed March, 1985. The updated manual contains the following pages:

	*Change Number
Title page	1
Copyright page	0
History page Change 1	1
History page	0
iii through ix(x Blank)	0
1-1 through 1-4	0
2-1 through 2-18	0
3-1 through 3-6	0
4-1 through 4-30	0
5-1 through 5-24	0
6-1 through 6-129	0

	*Change Number
6-130	1
6-131 through 6-133	0
6-134	1
6-135 through 6-241	0
6-242	1
6-243 through 6-424	0
6-425 and 6-426	1
A-1 through A-7(A-8 Blank)	0
B-1 through B-7(B-8 Blank)	0
C-1 through C-6	0
D-1(D-2 Blank)	0

* Zero in this column indicates an original page.

On a change page, the portion of the page affected by the latest change is indicated by a vertical bar in the outer margin of the page. However, a completely changed page will not have a full length bar, but will have the change notation by the page number.

HISTORY

The Gould V6 and V9 Reference Manual, Publication Order Number **301-003410-000**, was printed November 1984.

This manual contains the following pages:

Title page
Copyright page
History page
iii through ix (x Blank)
1-1 through 1-4
2-1 through 2-18
3-1 through 3-6
4-1 through 4-30
5-1 through 5-24
6-1 through 6-426
A-1 through A-7 (A-8 Blank)
B-1 through B-7 (B-8 Blank)
C-1 through C-6
D-1 (D-2 Blank)

TABLE OF CONTENTS

Chapter Page

5 INPUT/OUTPUT SYSTEM

6 INSTRUCTION REPERTOIRE

APPENDIX A

APPENDIX B

APPENDIX C

APPENDIX D

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

GENERAL DESCRIPTION

1.1 Introduction

Chapter 1 provides a general description of the Gould V6 and the Gould V9 Central Processing Units (CPUs) as well as the Internal Processing Units (IPUs). Unless otherwise noted in this reference manual, the term CPU refers to the V6 CPU and the V9 CPU. The term IPU refers to the V6 IPU and the V9 IPU.

The results obtained by executing a given instruction in either the V6 or V9 CPU are the same except where otherwise noted.

1.2 System Overview

1.2.1 General Information

The V6 and V9 CPUs are high performance computers, designed with large machine architecture. The combination of a CPU with an IPU permits higher system throughput as two programs may be run simultaneously.

The CPU and IPU have equal computational abilities; however, the processor selected to function as the CPU retains control of the I/O and interrupt operations. Either processor may be selected to function as the CPU or IPU. Should the CPU fail, changing a front panel switch will convert the IPU to a CPU and operation may be continued by rebooting the system.

1.2.2 Instruction Cycle Times

The V6 CPU has a 150-nanosecond machine cycle time and the V9 CPU has a 75-nanosecond machine cycle time. Both CPUs are enhanced by hardware logic that permits multiple CPU functions to be performed in one machine cycle. The precise instruction times are dependent upon the nature of the operation and the characteristics of the data involved. Typical instruction execution times are provided in Appendix B.

NOTES

1. Gould CSD software does not support all the standard Class E I/O devices on these computer systems. References to Class E operation in this document are for informational purposes only.

2. Gould CSD software manuals refer to a Class D I/O device. This nomenclature is for software referencing only since Class D is in reality Class E protocol with an IOCD format for all hardware/ firmware uses. These are the only Class E I/O devices that are supported by Gould CSD software.

1.3 Central Processing Unit (CPU)

1.3.1 Operating Mode

The CPU operates under control of the program status doubleword (PSD) and is capable of either privileged or unprivileged operation. During privileged operation, the CPU is permitted to perform control functions and input/output (I/O) instructions. Unprivileged operation is the normal user execution mode.

1.3.2 Nonbase and Base Register Modes

The CPU hardware supports two instruction sets: one for nonbase register mode and one for base register mode. The mode is controlled by bit 6 in the PSD (0 = nonbase register mode, 1 = base register mode). Nonbase register mode is used to maintain software compatibility with previous Gould CONCEPT and 32 SERIES computer systems. It is more restrictive than base register mode because software programs are limited to the first 128K words of logical address space. In base register mode, software programs can occupy the entire logical address space of 4M words (16M bytes). The CPU is provided with a set of eight high-speed base registers. These registers are used in base register mode instructions to calculate the logical address.

NOTE

> The V6 CPU and V9 CPU are designed to run in the Base Register mode of operation ONLY. Non-Base Register mode of operation can run, but this mode is not fully supported. Indirect addressing in mapped environment may not function.

1.3.3 General Purpose Register

The CPU is provided with a set of eight high-speed general purpose registers (GPR). These registers are used in most instructions, such as arithmetic, logical, and shift operations. Register R0 is also used as a link register between software subroutines; register R4 is used as the mask register. In the nonbase register mode, registers R1 through R3 may be used as index registers. In the base register mode, registers R1 through R7 may be used as index registers.

1.3.4 Memory Management and Address Generation

The memory management of the CPU permits full utilization of all available memory. This feature includes hardware memory allocation and protection (MAP). The memory management scheme of the CPU comprises the following:

. 2048 word MAP block (or page)
. 2048 word write protect granularity
. 2048 entry hardware MAP
. 4M word (16M byte) maximum logical address space
. 4M word (16M byte) maximum physical space

There are two memory addressing environments: mapped and unmapped. Under each, there are two options (when in the nonbase register mode): extended and nonextended. The user controls the selection of the options under each environment, and it is these options which determine the rules for logical address generation.

1.3.5 Interrupts

Interrupts are the means by which real-time events, external to the CPU, are reported to software. Interrupts are events that are prioritized, scheduled, and in some cases deferred. There are minor differences between the V6 and the V9 interrupts. Refer to Chapter 4.

1.3.6 Traps

Traps are exceptional conditions that are identified and reported to software by the CPU or IPU. All traps have the same priority, with the exception of the power fail trap. This trap overrides all other traps. Additionally, all traps are non-deferrable.

1.3.7 Input/Output (I/O) Operations

I/O operations consist of transferring data in blocks of bytes, halfwords, or words between peripheral devices and main memory. Once initiated, such transfers occur automatically, which leaves the CPU free for other tasks.

1.4 Internal Processing Unit (IPU)

1.4.1 V6 IPU

The V6 IPU is functionally identical to the V6 CPU. Either processor may be selected to function as the CPU with a switch on the turnkey/display panel. The V6 IPU has the same capabilities as the CPU with the following exceptions:

- The IPU traps all class 3, E and F instructions
- The IPU traps all BEI instructions

Because the IPU is identical to the CPU, all information contained in this reference manual pertains to both the CPU and the IPU except where specifically noted.

1.4.2 V9 IPU

The V9 IPU is functionally identical to the V9 CPU. Either processor may be selected to function as the CPU with a switch on the turnkey/display panel. The V9 IPU has the same capabilities as the CPU with the following exceptions:

- The IPU has no I/O capabilities
- The IPU ignores all interrupt requests
- The IPU traps all I/O requests to the CPU
- All IOP functions are handled by the CPU

Hardware interaction between the CPU and IPU is limited to the Start-Stop IPU Trap which is set by the SIPU instruction under software control. The Start/Stop IPU Trap and the SIPU instructions are defined in Chapters 4 and 6, respectively, of this manual.

CHAPTER 2

CENTRAL PROCESSOR

2.1 Introduction

This chapter includes basic information concerning the Central Processing Unit (CPU) control modes, followed by a discussion of the instruction repertoire, the memory reference instruction format, and addressing modes. Brief descriptions of the program status doubleword (PSD), condition codes, and privileged/unprivileged operation are given. The instruction repertoire portion includes an accounting of the instruction set with a discussion of memory word boundaries. The memory reference instruction is compared for base register mode and nonbase register mode, followed by the direct, indirect, and indexed addressing techniques.

2.2 CPU Control Modes

2.2.1 Program Status Doubleword

The CPU operates under the control of the program status doubleword (PSD). The PSD records machine conditions that must be preserved prior to the context switching. The format of the PSD is shown in Figure 2-1.

2.2.2 Condition Codes

The four condition code bits in the PSD (bits 1 through 4) are set upon execution of most instructions. For arithmetic operations, the condition codes are set as follows:

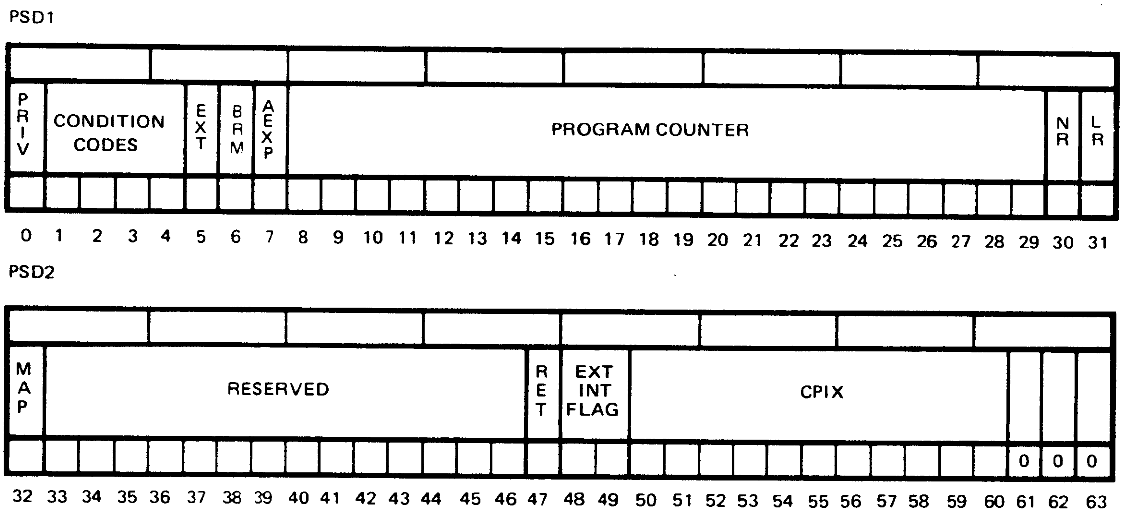
- CC1 is set if an arithmetic exception occurs.
- CC2 is set if the result is greater than zero.
- CC3 is set if the result is less than zero.
- CC4 is set if the result is equal to zero.

The branch condition true (BCT), branch condition false (BCF), and the branch function true (BFT) instructions allow testing and branching on the condition codes.

2.2.3 Privileged/Unprivileged Operations

The CPU is capable of either privileged or unprivileged operation. During privileged operation, the CPU is permitted to perform control functions and input/output (I/O) instructions. Unprivileged operation is the normal user execution mode and all privileged operations are prohibited.

Bit 0 in the PSD is the privileged state bit. If the privileged state bit is set, privileged instructions can be executed and a privileged user may write to protected memory. If the privileged state bit is reset, any attempt to execute a privileged instruction will cause a privilege violation trap. The following instructions are privileged:



BIT 0 = 0 UNPRIVILEGED STATE
= 1 PRIVILEGED STATE

BITS 1-4 CONDITION CODES
1 = CC1
2 = CC2
3 = CC3
4 = CC4

BIT 5 = 0 EXTENDED OPTION (OFF)
= 1 EXTENDED OPTION (ON)

BIT 6 = 0 NON BASE REGISTER MODE
= 1 BASE REGISTER MODE

BIT 7 = 0 ARITHMETIC EXCEPTION TRAP DISABLED
= 1 ARITHMETIC EXCEPTION TRAP ENABLED

BITS 8-30 PROGRAM COUNTER FIELD (128KW WORDS MAXIMUM COUNT FOR NON-BASE REGISTER MODE, 4MW WORDS MAXIMUM COUNT FOR BASE REGISTER MODE)

BIT 30 NEXT INSTRUCTION IS A RIGHT HALFWORD
BIT 31 LAST INSTRUCTION WAS A RIGHT HALFWORD
BIT 32 = 0 UNMAPPED ENVIRONMENT
= 1 MAPPED ENVIRONMENT

BIT 47 RETAIN CURRENT MAP

BITS 48-49 INTERRUPT CONTROL FLAGS

BITS		
48	49	
0	0	OPERATE WITH UNBLOCKED INTERRUPTS
0	1	OPERATE WITH BLOCKED INTERRUPTS
1	0	RETAIN CURRENT BLOCKING MODE
1	1	RETAIN CURRENT BLOCKING MODE

BITS 50-63 INDEX INTO MASTER PROCESS LIST FOR CURRENT PROCESS

BITS 31, AND 33-46 ARE RESERVED (MUST BE ZERO FILLED)

820724

Figure 2-1. Program Status Doubleword Format

1. All interrupt related instructions such as enable interrupt or request interrupt.

2. All instructions that can modify the memory mapping registers.

3. All input/output instructions.

4. All instructions that can place the machine in a state that requires operator intervention to continue processing, such as HALT, or change the machine operating environment.

Certain events can change the CPU from the unprivileged to the privileged state by loading a new program status doubleword. These events are as follows:

1. An interrupt from an external event or the I/O system.
2. A hardware trap caused by addressing nonpresent memory, executing an undefined instruction, the execution of a privileged instruction by a nonprivileged program, or writing to protected memory.
3. A hardware trap caused by a nonrecoverable condition such as an uncorrectable error on a memory read, or an arithmetic exception.
4. The execution of the supervisor call instruction by a user requesting monitor services.
5. System reset sets the privileged state bit.

The execution of either a load program status doubleword (LPSD) or a load program status doubleword and change MAP (LPSDCM) instruction can cause the system to change from the privileged to the unprivileged state.

2.3 Instruction Repertoire and Formats

The functional classifications and corresponding number of instructions for the V6 and the V9 CPUs are as follows:

	Number of Instructions	
Classification	V6	V9
Bit manipulation	8	8
Boolean (logical)	17	17
Branch	14	14
Class F I/O	13	13
Compare	11	11
Control	20	19
Fixed-point arithmetic	30	30
Floating-point arithmetic	16	16
Floating-point conversion	4	4
Input/output	2	2
Interrupt	7	7
Load/store	37	37
Memory management	4	4
Register transfer	15	15
Shift	13	13
Writable control storage	3	0
Total	214	210

The instructions are classified as either word instructions (32 bits) or halfword instructions (16 bits). The word instructions primarily refer to memory operands; the halfword instructions primarily deal with register operands. Program memory can be conserved by packing two consecutive halfword instructions into one memory location (word).

The instruction lookahead technique allows for fast instruction execution. Instruction fetches are made concurrently with instruction execution and the decoding of a previously fetched instruction.

Of particular significance are the bit manipulation instructions because they provide the capability to selectively set, zero, add, or test any bit in memory or in a register.

2.4 Memory Boundaries

2.4.1 Instructions

Each fullword instruction (32 bits) must be stored in memory on a word boundary (address with bits 30 and 31 equal to zero). Memory information boundaries are illustrated in Figure 2-2.

Halfword instructions may be stored two per word. However, when a halfword is followed by a word instruction, the assembler positions the halfword instruction in the left half of the word and stores a no operation (NOP) instruction in the right half of the word. This maintains the word boundary discipline.

Memory reference instructions which address a byte in memory do not alter the other three bytes in the memory word containing the specified byte. Memory instructions which address a halfword do not alter the other halfword of the memory location. The exception to the preceding is the add bit in memory instruction. This is actually a 32-bit add; therefore, it may propagate a carry as far as the most-significant bit of the word containing the specified bit.

2.4.2 Operands

Word operands must be stored in memory on a word boundary. The most significant word of a doubleword operand must be stored in a memory location having an even word address with the least significant word stored in the next sequentially higher (i.e., odd word) location. Some examples of memory addressing follow:

Byte	Halfword	Word	Doubleword
00000	00000	00000	00000
00001			
00002	00002		
00003			
00004	00004	00004	
00005			
00006	00006		
00007			
00008	00008	00008	00008
00009			
0000A	0000A		
0000B			
0000C	0000C	0000C	
0000D			
0000E	0000E		
0000F			
00010	00010	00010	00010

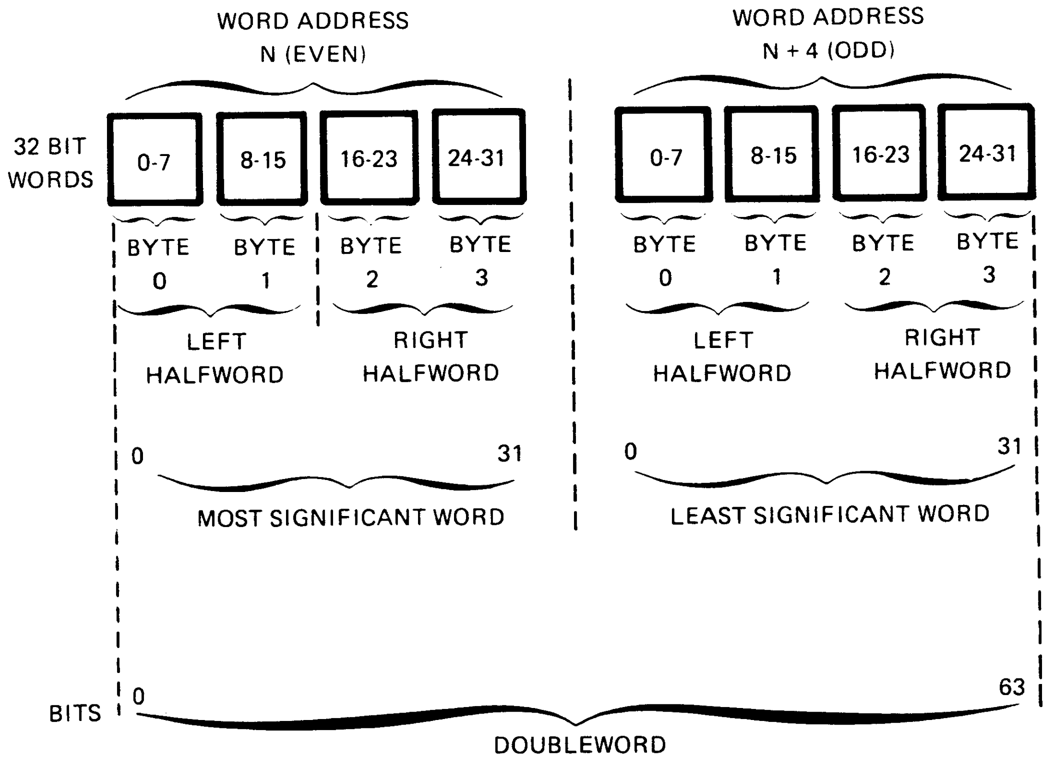


Figure 2-2. Information Boundaries in Memory

Byte, halfword, or word operands are always right-justified when handled in the 32-bit data structure. Doublewords are processed as two single words (LSW processed first then MSW) with carry and zero detect transmission between LSW and MSW if applicable.

2.4.3 Instruction Formats

The CPU supports two instruction sets: one when utilizing the base register mode, and the other when operating with the nonbase register mode. The mode is controlled by bit 6 in the PSD (0= nonbase register mode, 1= base register mode).

2.4.4 Program Counter

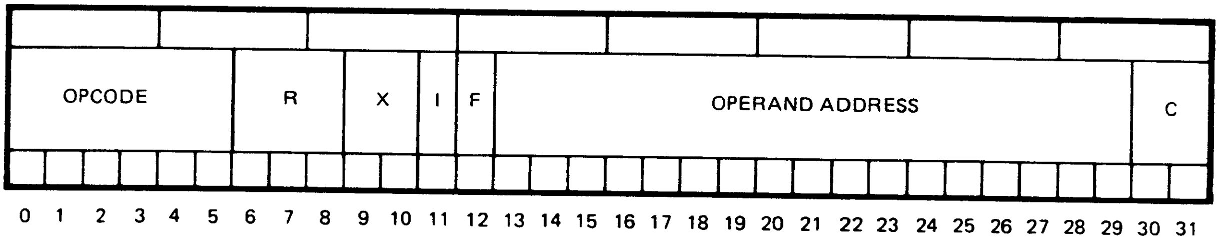
The program counter (PC) contains the logical address of the instruction about to be executed. The PC field occupies bits 8 through 30 of the PSD (see Figure 2-1). In the base register mode, the maximum PC value is 16M bytes; in the nonbase register mode, the maximum PC value allowed is 512 K bytes.

2.5 Memory Reference Instructions

With the implementation of the base register mode, the application of memory reference instructions requires two distinct formats. The operation code (bits 0 through 5) in both formats is identical for recognition of each separate instruction. Also, bits 6 through 8 identify the GPR to be used as an operand source or destination.

2.5.1 Nonbase Register Mode

In the nonbase register mode, bits 9 and 10 specify the GPR to be used as an index register, bit 11 is the indirect bit, and bits 12-31 define the operand address and data type. The effective address of the operand depends on the values of I, X, and bits 12-31.



830555

2.5.1.1 F and C Bits

The format of the F and C bits is designed so that any selected data type (byte, halfword, word, or doubleword) can be conveniently specified by combinations of the F and C bits as follows:

F	C	Data Type
0	00	32-bit word
0	01	Left halfword (bits 0-15)
0	10	64-bit doubleword
0	11	Right halfword (bits 16-31)
1	00	Byte 0 (bits 0-7)
1	01	Byte 1 (bits 8-15)
1	10	Byte 2 (bits 16-23)
1	11	Byte 3 (bits 24-31)

NOTE

For restrictions of the F and C bits refer to indirect addressing and address specification traps in chapters 2 and 4 respectively.

2.5.1.2 Direct Addressing

When bits 9 and 10 (X field) are zero (no indexing), and bit 11 (I field) is zero (no indirect), the effective memory address is taken directly from bits 13 through 29 of the memory reference instruction.

For example, the store word instruction is coded: STW 0,0 and is assembled as hexadecimal D4000000. When executed, this instruction stores the contents of GPR0 directly into memory word location 0.

Likewise, the store byte instruction is coded: STB 0,1 and is assembled as hexadecimal D4080001. Note that the F and C fields of this instruction have been altered. When executed, this instruction stores the least significant byte of GPR0 directly into memory byte location 1.

2.5.1.3 Indexed Addressing

When bits 9 and 10 (X field) are nonzero, indexed addressing is in effect. Bits 13 through 31 of the instruction are used to produce a memory address by adding to these bits the contents of the GPR, bits 12 through 31, specified by the X field. Only GPRs 1, 2, and 3 function as index registers.

Any data type may be indexed in sequential fashion by adjusting the index value by the size of the data type. This can be done by adding the bit position that corresponds to the displacement value for the applicable data type to the index register. These are as follows:

Data Type	Bit Position	Displacement Value
Byte	31	1
Halfword	30	2
Word	29	4
Doubleword	28	8

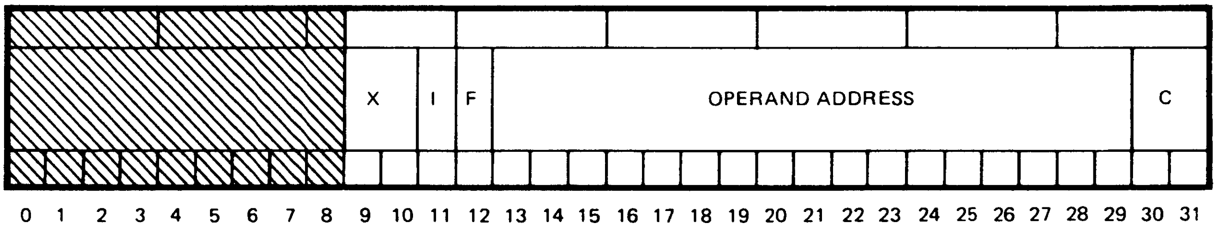
For indexed addressing, except for extended indexing, the displacement value is a twos complement integer within one of the GPRs used for indexing. For word indexing, bit 29 of the index register is the least significant bit of the address. If bit 29 of GPR3 is set to one to provide a displacement of one word, the indexed store word instruction is coded as: STW 0, 0, 3. This now stores the contents of GPR0 in the memory location indexed by the contents of GPR3. The instruction would assemble as D4600000. The calculated logical effective word operand address (after indexing) is 00004. Therefore, the contents of GPR0 are stored in memory location 0004.

During indexing, only the C bits may change; the F bit is unaffected. If the original mode of addressing was the byte mode, indexing may only specify which particular byte is being addressed. If the mode was not originally a byte attribute, indexing can select a halfword, word, or doubleword attribute, depending on which C bits are set.

For example, the load word instruction for indexing is coded: LW 5, X'1000', 2. The contents of register 2 contain X'0000 2000'. The word from memory location 3000 will be loaded into register 5.

2.5.1.4 Indirect Addressing

When bit 11 (I field) is one, addressing is indirect, and the CPU retrieves an indirect word specified by the operand address. In this indirect word, bits 9 and 10 select the index register, bit 11 is the indirect bit, and bits 12 through 31 specify an operand address.



830556

For example, to use the indirect addressing capability, the store word instruction would be coded: STW 0,*0. This causes bit 11, the indirect bit, to be set to one. When this instruction is executed, it stores the contents of GPR0 into the location specified by the address found in memory location 0.

Multilevel indirect addressing can be performed when each new address taken from memory has the indirect bit set to one. The process of fetching indirect addresses continues until a memory address has bit 11 set to zero. This address is the logical effective operand address.

An indirect fetch is always a word fetch, and an indirect word can specify a byte, halfword, or doubleword attribute. An indirect word can specify new F and C bits, or if the indirect word has F bit equal to 0 and C bits equal to 00, then the previous addressing attribute will be used. Indirect addressing is the only way to change the F and C bits.

For example, to use indirect addressing, the load word instruction is coded: LW 5,*X'1000'. Memory location X'1000' contains X'0008 3000'; therefore, byte 0 from memory location 3000 is loaded, right justified and zero filled, into register 5.

NOTE

> The V6 CPU and V9 CPU are designed to run in the Base Register mode of operation ONLY. Non-Base Register mode of operation can run, but this mode is not fully supported. Indirect addressing in mapped environment may not function.

2.5.1.5 Indirect and Indexed Addressing

Indirect addressing can be combined with indexing at any indirect level. An example of indirect addressing with indexing follows.

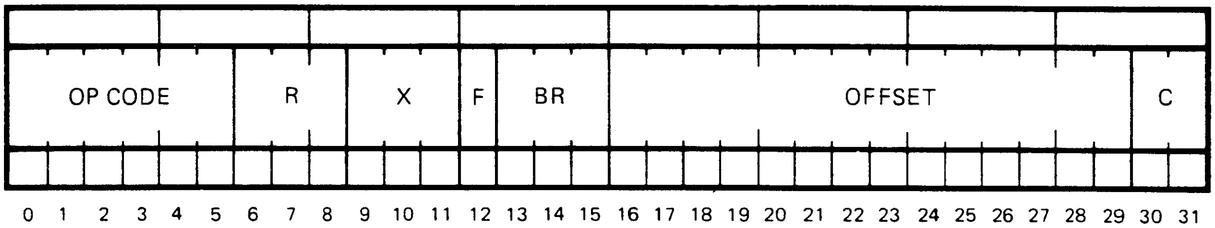
```
Location   Machine      Byte
Counter    Instruction  Address    Label   Operation   Operand

                                           PROGRAM
P00000                                     REL
P00000     C9800004                STRT    LI          R3,4
P00004     AC90000C     P0000C             LW          R1,*LOC1
P00008     C8061055                        SVC         1,X'55'
P0000C     00100010     P00010     LOC1    ACW         *LOC2
P00010     00700014     P00014     LOC2    ACW         *LOC3,R3
P00014     00000000                LOC3    DATAW       0
P00018     0000001C     P0001C             ACW         LOC4
P0001C     0000FFFF                LOC4    DATAW       X'0000FFFF'
P00020                  P00000             END         STRT
```

The first executable instruction is the load immediate (LI) to load a value of 4 into GPR3. The next instruction to be executed is the load word (LW). The indirect bit of this instruction is set; therefore, the operand address in the LW points to an indirect word at location P0000C (LOC1). Since the indirect bit is set in the indirect word at LOC1, the indirect addressing chain is continued. The next indirect word at label LOC2 has the indirect bit set and also specifies GPR3 as the index register. So, the contents of the address word field of this indirect word are added to the contents of GPR3 to form the address of the third indirect word in the indirect addressing sequence. The resulting address points to location P00018. The indirect bit in the indirect word at this location is not set, indicating that the contents of the indirect word is the effective address of the target operand. The effective address points to label LOC4. The contents at this location are loaded into GPR1. At this point in the execution of the instructions, GPR1 contains the hexadecimal value 0000FFFF.

The ACW statement is a macro assembler directive used to generate an address constant. The DATAW is also a macro assembler directive, and the SVC 1,X'55' is a call to the monitor exit service in MPX.

2.5.2 Base Register Mode

In the base register mode, the CPU allows GPRs 1 through 7 to be used as index registers. Bits 9 through 11 of the memory reference instruction represent the index register field. The F bit (bit 12) is a part of the operation code.



820430

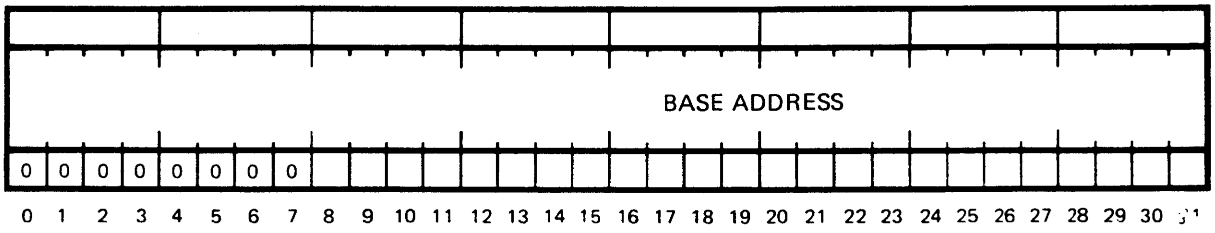
The contents of the base register field (bits 13 through 15) identify one of the seven registers to be used as a reference address within the program address space. The offset field contains the positive displacement value that is added to the contents of the specified base register to form the address of the operand. If the base register field contains all zeros, a 32-bit value of zero is used as the base address in the logical address calculation.

2.5.2.1 Address Alignment

The CPU checks the alignment of the effective address of the operand against the alignment specified in the instruction. If the compare does not agree, the hardware will generate the address specification trap.

2.5.2.2 Base Register Format

The base address field of the base register format is 24 bits.



840034

2.6 Memory Address Generation

There are two memory addressing environments: mapped and unmapped. There are two options when in nonbase register mode: extended and nonextended. The user controls the selection of the options under each environment, and it is these options which determine the rules for logical address generation.

The memory environment is controlled by the software operating system which determines the rules for transposing logical addresses into physical addresses.

When the user's task is loaded into memory, it may be dispersed into noncontiguous map blocks throughout physical memory. All of the MAP blocks used for a specific user's task are considered the physical (real) space of that task.

Physical blocks of memory can be common to many logical address spaces. Thus, multiple users may have access to some of the same physical address space and share those common blocks of memory.

2.6.1 Mapped Environment

The memory management hardware is used to convert a task's address space to the assigned physical MAP blocks. The set of valid addresses is known as the logical address space of the task. Figures 2-3 through 2-5 illustrate how the memory management hardware uses a memory MAP (random access memory) to transform logical space (addresses) into physical space (addresses) for nonbase-nonextended (Figure 2-3), nonbase-extended (Figure 2-4) and base register mode (Figure 2-5).

The CPU is operating in the mapped environment when bit 32 of the PSD2 is set. The mapped environment specifies that the user's program has been partitioned into MAP blocks of 2K words per block and the blocks may be distributed throughout physical memory. In the mapped environment, the CPU loads the physical MAP registers through a table called the MAP image descriptor list (MIDL). Each MAP image descriptor of the MIDL defines a unique MAP register image. Thus, all MAP entries may be linked together contiguously in the logical memory space. Consecutive entries in the MIDL will produce a contiguous logical address space.

Indirect addressing in mapped environment may not function.

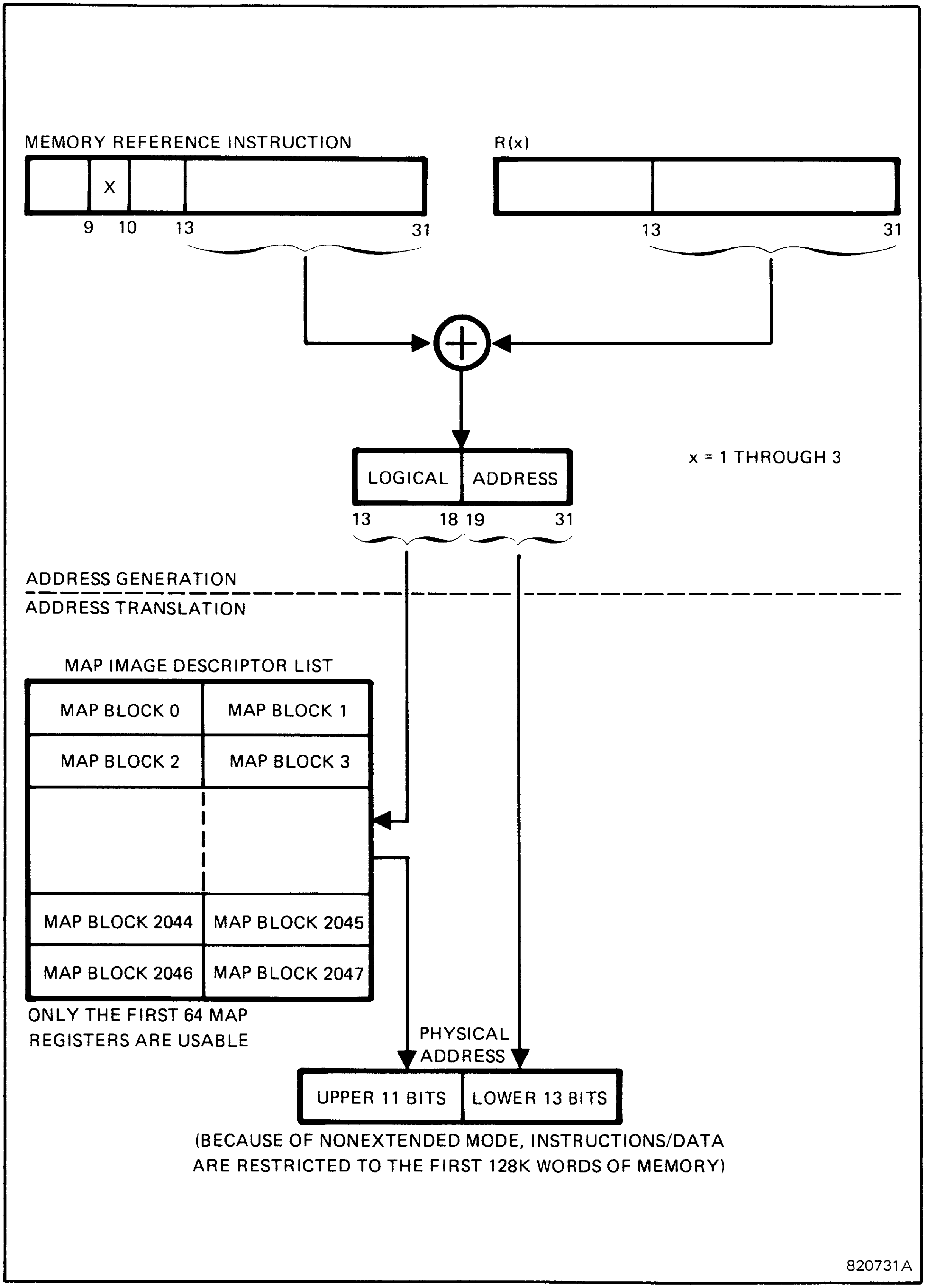


Figure 2-3. Mapped Environment for Nonbase Nonextended Mode

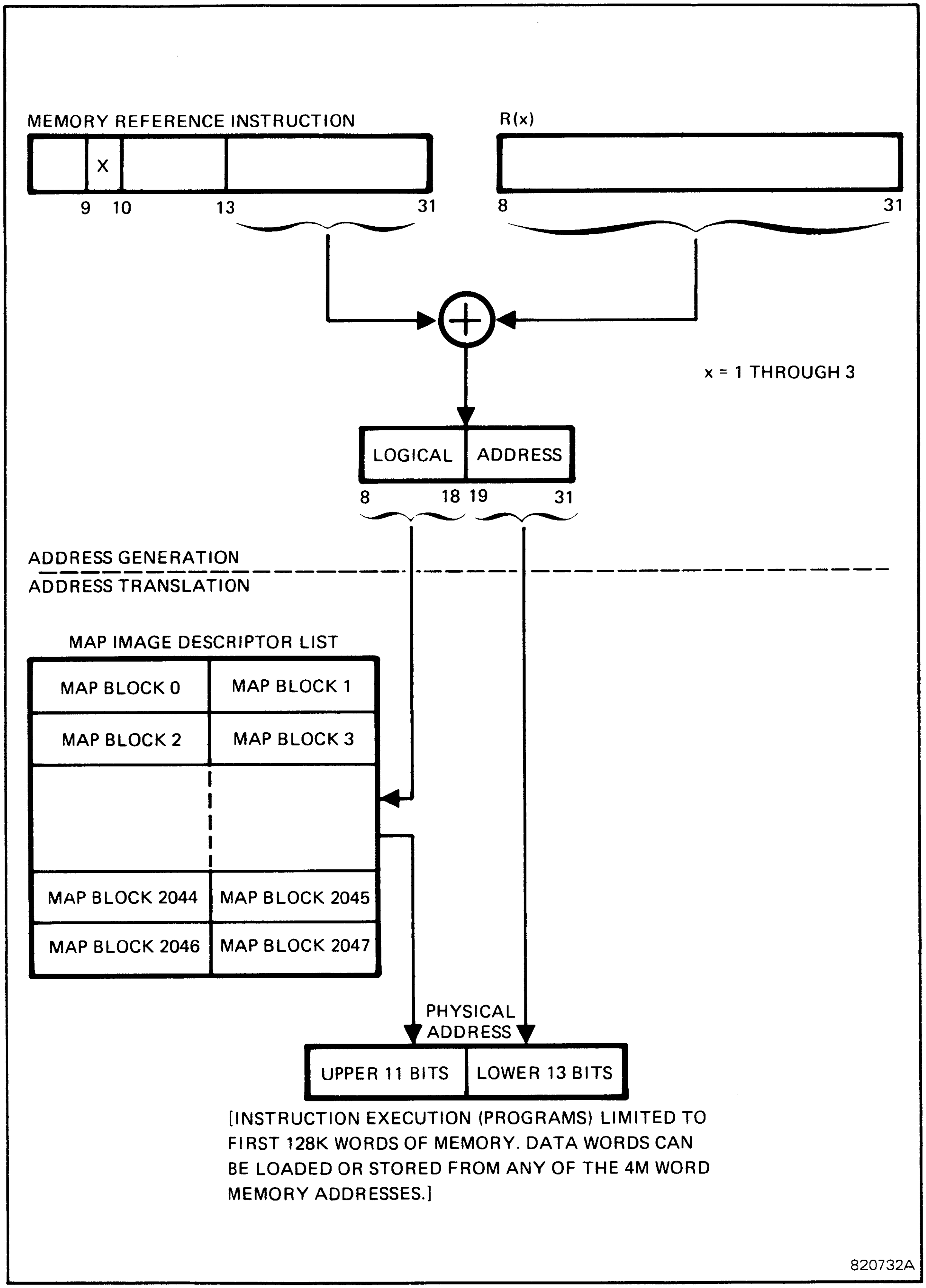


Figure 2-4. Mapped Environment for Nonbase Extended Mode

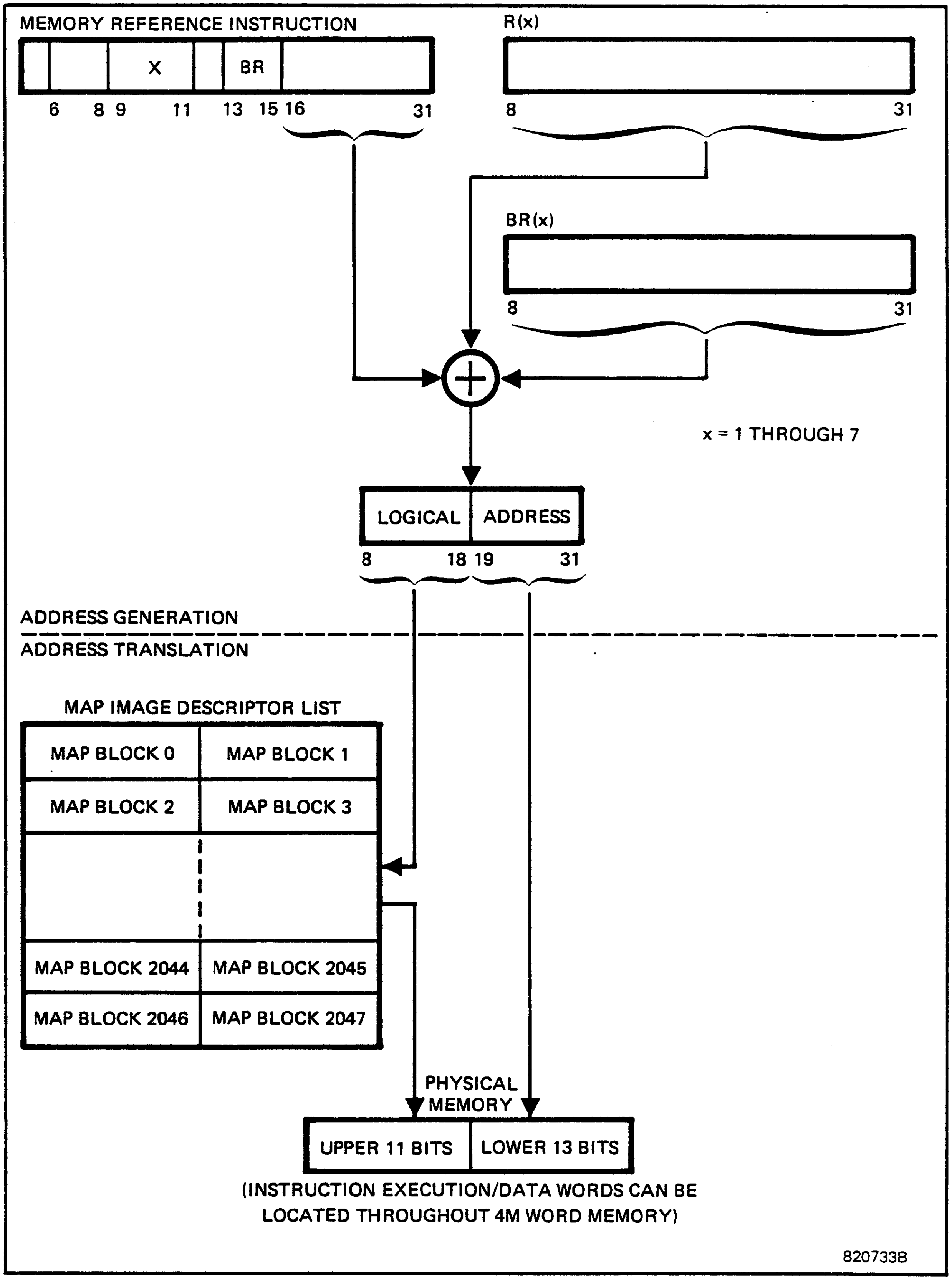


Figure 2-5. Mapped Environment for Base Register Mode

2.6.2 Unmapped Environment

When the CPU is operating in the unmapped environment, the MAP registers are not used. No address transformation takes place; therefore, the logical address is identical to the physical address. The CPU is in the unmapped environment when bit 32 of the PSD2 is zero. No memory protection is provided in this environment. Figures 2-6 through 2-8 illustrate the translation of a logical address to a physical address for nonbase-nonextended (Figure 2-6), nonbase-extended (Figure 2-7), and base register mode (Figure 2-8).

2.6.3 Nonextended Addressing Option

The nonextended option is in effect when bits 5 and 6 of the PSD1 are zero.

In the nonbase register mode, the nonextended option allows the CPU to access only those instructions and operands (bit, byte, halfword, word, or doubleword) in the primary address space. A 19-bit address field is provided in all memory reference instructions for this purpose. Refer to Figures 2-3 and 2-6 for the nonextended addressing calculation.

When bit 6 is set to one, bit 5 has no relevancy and the system is in the base register mode, which provides a 24-bit address field. Refer to Figures 2-5 and 2-8 for the base register mode address calculation.

2.6.4 Extended Addressing Option

Refer to Figures 2-4 and 2-7 for the extended addressing calculation. The logical address space beyond the first 128K words in the nonbase register mode may be used for operands only.

The upper limit of the extended space is 4M words.

In the nonbase register mode, indexed addressing is necessary to achieve addressing above 128K words. In each memory reference instruction, bits 9 and 10 designate one of three general purpose registers to be used as an index register. The extended option is in effect when bit 5 of the PSD1 is set to one and bit 6 is zero. Only bits 8 through 31 of the index register are used.

In the base register mode, the extended addressing option is not required and therefore, not used.

2.6.5 Read/Write Protection

The memory protection system provides read/write protection for individual memory map blocks. See Figure 3-4 for the security protection scheme.

When the CPU is operating in the mapped environment, any map block (2048 W) of logical program address space can be read/write-protected. This is done by setting the appropriate protect bits in the MAP image descriptor for that particular map block. Details of the MAP image descriptor are provided in Chapter 3.

If a task attempts to read or write to a location which is not defined in its logical address space, an illegal MAP access (read/write protect violation) trap will occur. This prevents a task from accessing memory that is not part of its address space. If a task attempts to write to a location within its own address space and the operating system has defined that location as protected, a privilege violation trap will occur. This provides security and prevents the task from destroying critical locations within its own logical address space.

No protection is provided in the unmapped environment; direct access to all available physical address space is attainable.

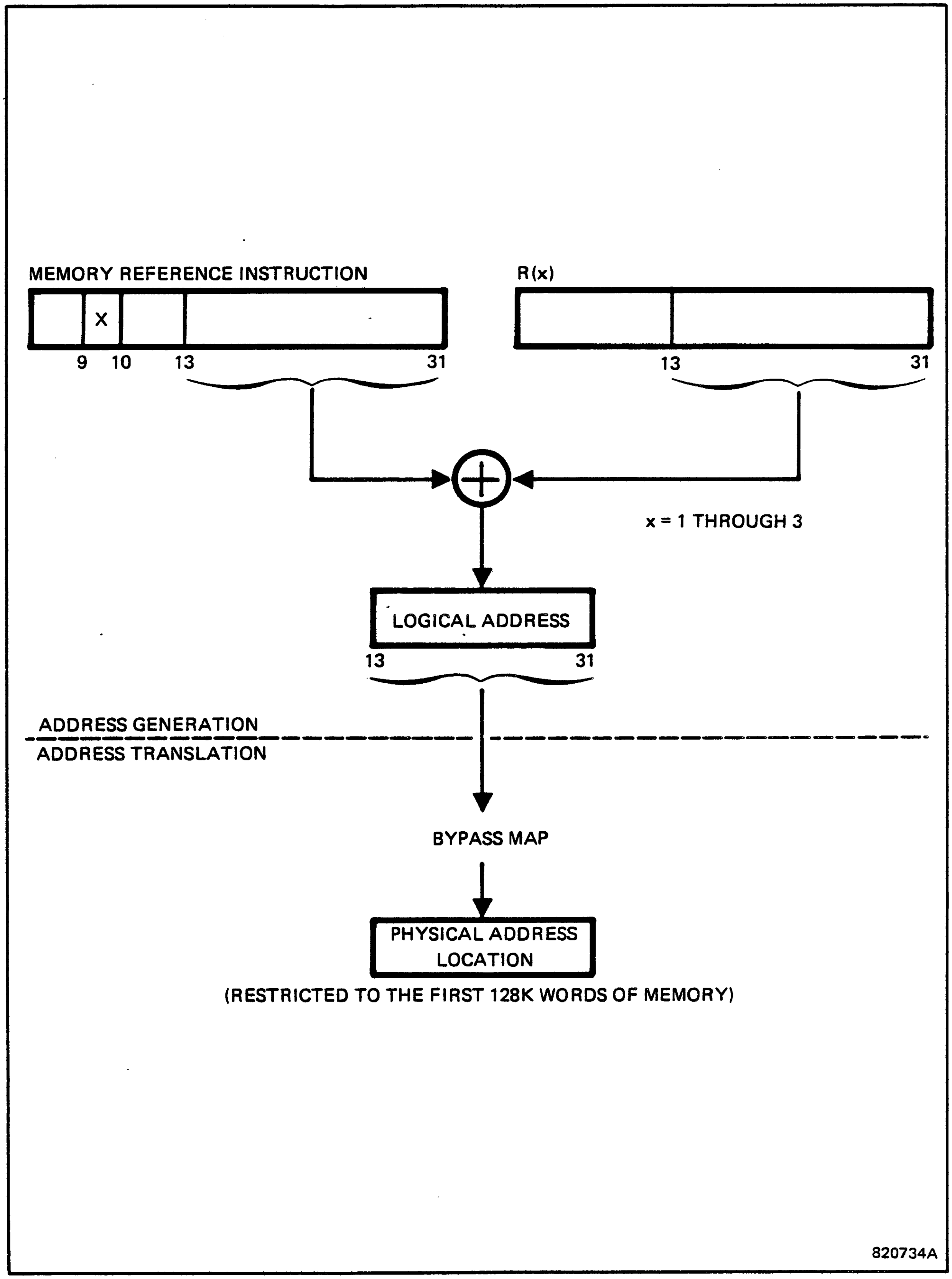


Figure 2-6. Unmapped Environment for Nonbase Nonextended Mode

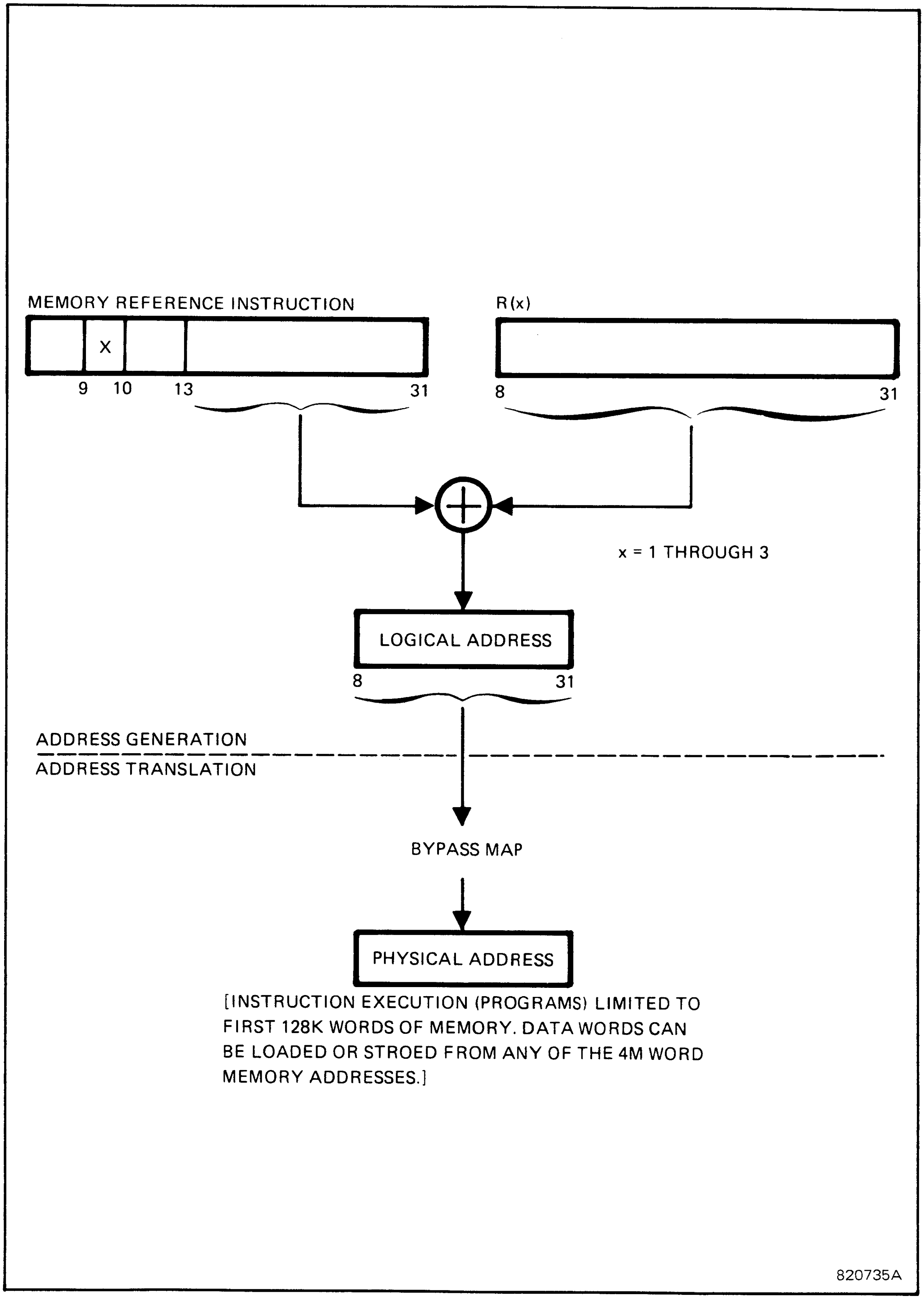

MEMORY REFERENCE INSTRUCTION
X
9
10
13
31
R(x)
8
31
x = 1 THROUGH 3
LOGICAL ADDRESS
8
31
ADDRESS GENERATION
ADDRESS TRANSLATION
BYPASS MAP
PHYSICAL ADDRESS
[INSTRUCTION EXECUTION (PROGRAMS) LIMITED TO FIRST 128K WORDS OF MEMORY. DATA WORDS CAN BE LOADED OR STROED FROM ANY OF THE 4M WORD MEMORY ADDRESSES.]
820735A

Figure 2-7. Unmapped Environment for Nonbase Extended Mode

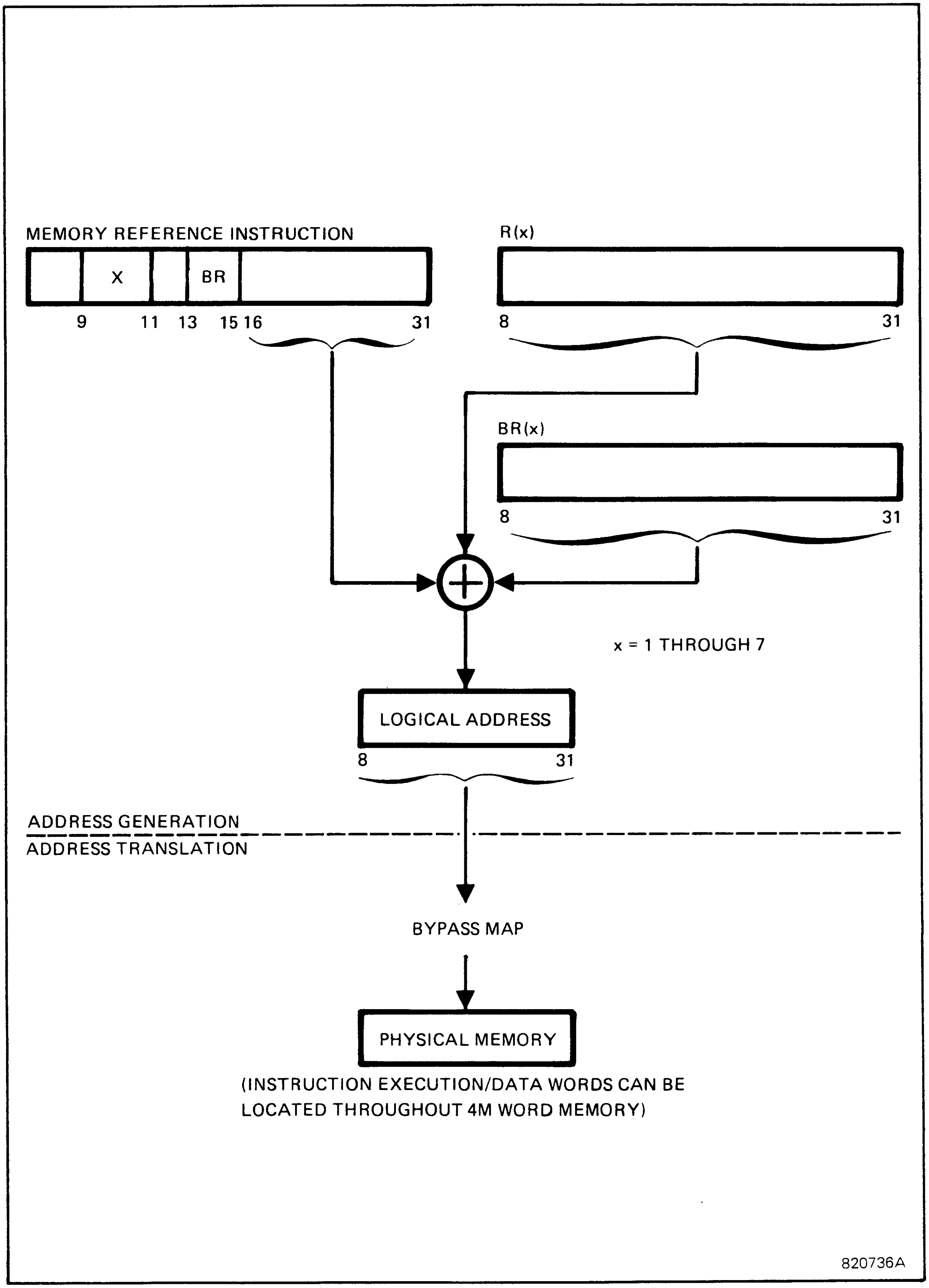


Figure 2-8. Unmapped Environment for Base Register Mode

CHAPTER 3

MEMORY MANAGEMENT

3.1 Hardware Memory Management

The hardware memory management of the CPU permits full utilization of all available memory. This feature includes hardware memory allocation and protection (MAP). The hardware memory management allows user programs to be loaded into and executed from anywhere in physical memory.

3.2 Memory Mapping Scheme

The memory mapping scheme consists of the following principal parameters:

- 2048 MAP blocks (pages) each of 2KW length
- 2048 word write protect granularity.
- 4M word (16M byte) maximum logical address space.
- 4M word (16M byte) maximum physical space.

3.3 Memory Mapping Data Structures

Figure 3-1 depicts the software memory mapping data structures used by the CPU to load its MAP. The master process list (MPL) and the MAP image descriptor lists (MIDLs) must be kept in memory on doubleword boundaries. These contain the information for the CPU to initialize the MAP. MPL 0 is normally reserved for the operating system (OS). The remaining MPLs are used for tasks (programs) within the OS.

3.4 Current Process Index

The second word of the program status doubleword (PSD) contains the 14-bit current process index (CPIX) field. The CPIX is the index that locates the MAP segment descriptors (MSDs) in the master process list (MPL) in order to provide a link from the PSD to the MAP image descriptors (MIDs). As the CPIX must point to a doubleword boundary, the three least-significant bits of the 14-bit CPIX field are always zero.

3.5 Master Process List

The MAP segment descriptors (MSDs) are contained in the MPL. The address of the MPL is set at system reset time by loading a predetermined scratchpad cell (F3-hex) with the 24-bit physical MPL address. This location points to MSD 0. Therefore, when the CPIX equals 0, the MIDs for MSD 0 are used. If the CPIX is not equal to 0, then the CPIX and location F3 (hex) are added together and the resultant points to an MSD entry other than zero (on a doubleword boundary).

The format of an MSD entry is illustrated in Figure 3-2. Bit 0 of word 0 in an MSD is interpreted one way for MSD 0 and another way for all other MSDs. For MSD 0, bit 0 is called the retain bit, and for any other MSD, this bit is called the include bit. When a MAP change is required as a result of a Load Program Status Doubleword and Change MAP (LPSDCM) instruction, the firmware determines the appropriate MSD to retrieve by adding the CPIX portion of PSD word 2 to the MPL base address located in scratchpad (location F3-hex). The resultant MSD is retrieved. Firmware analyzes bit 0 (include bit) of the retrieved MSD.

If the CPIX is not equal to zero and bit 0 of the selected MSD is set, then firmware retrieves MDS 0. Bit 0 (retain bit) of MSD 0 is analyzed. If bit 0 is zero, an absolute load of all MAPs described by MSD 0 occurs.

If the CPIX is not equal to zero and bit 0 of MSD 0 is set, the MAP blocks of MSD 0 are retained. The user task which is executing uses the MAP blocks defined by the CPIX field of the PSD to translate the logical address of the instruction/operand into a physical memory address while retaining the MAP blocks of MSD 0.

If the CPIX is not equal to zero and bit 0 of the selected MSD is zero, the entire map is described by the selected MSD (no MSD 0). For this case, no map registers are absolute loaded, but are loaded when referenced. A context switch under these conditions will result in the entire map being cleared.

If the CPIX is equal to zero and bit 0 of MSD 0 is set, the entire map is described by MSD 0. For this case, all map registers are absolute loaded. A context switch under these conditions will result in the map being fully loaded during the context switch and there will be no need for loading a map at the time of reference.

Bits 1-15 of word 0 in the MSD format are reserved for future use. Bits 16-31, the segment count, contain the number of MAP block entries in the MIDL.

Word 1 of the MSD contains the MIDL pointer which is the physical address of the first MAP image descriptor (MID) in the MIDL. The MIDL pointer must point to a word address (bits 30 and 31 are zero).

3.6 Map Image Descriptor List

The MIDL is used to map logical addresses into physical memory addresses. Each MIDL entry associates a MAP block of the logical address space with a MAP block of physical memory. The logical address space is defined by building the MIDL as shown in figure 3-3.

3.7 Map Image Descriptor

Each MAP image descriptor (MID) shown in Figure 3-4 is a halfword that contains a MAP block valid bit, two protect bits, a memory modify bit, a memory access bit, and an 11-bit MAP block entry.

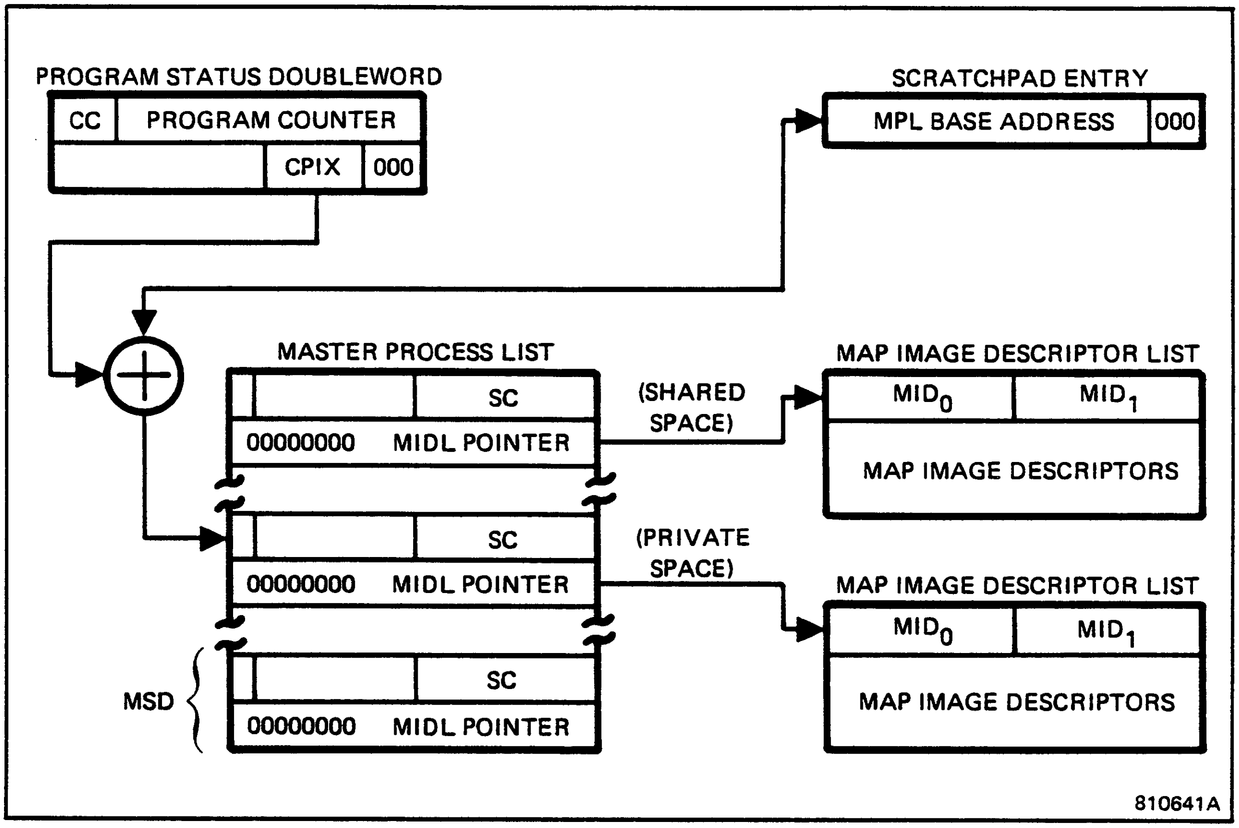
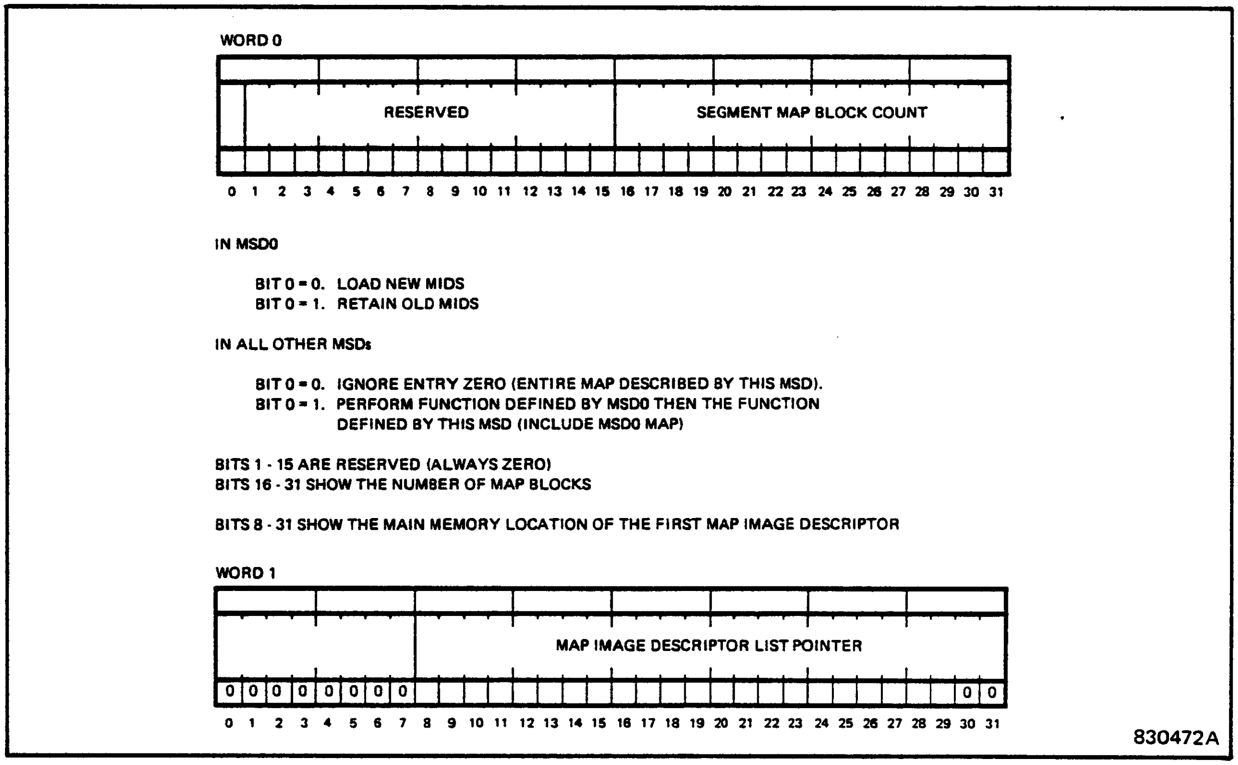


Figure 3-1. Memory Management Data Structures



IN MSD0

BIT 0 = 0. LOAD NEW MIDS
BIT 0 = 1. RETAIN OLD MIDS

IN ALL OTHER MSDs

BIT 0 = 0. IGNORE ENTRY ZERO (ENTIRE MAP DESCRIBED BY THIS MSD).
BIT 0 = 1. PERFORM FUNCTION DEFINED BY MSD0 THEN THE FUNCTION DEFINED BY THIS MSD (INCLUDE MSD0 MAP)

BITS 1 - 15 ARE RESERVED (ALWAYS ZERO)
BITS 16 - 31 SHOW THE NUMBER OF MAP BLOCKS

BITS 8 - 31 SHOW THE MAIN MEMORY LOCATION OF THE FIRST MAP IMAGE DESCRIPTOR

Figure 3-2. MAP Segment Descriptor (MSD)

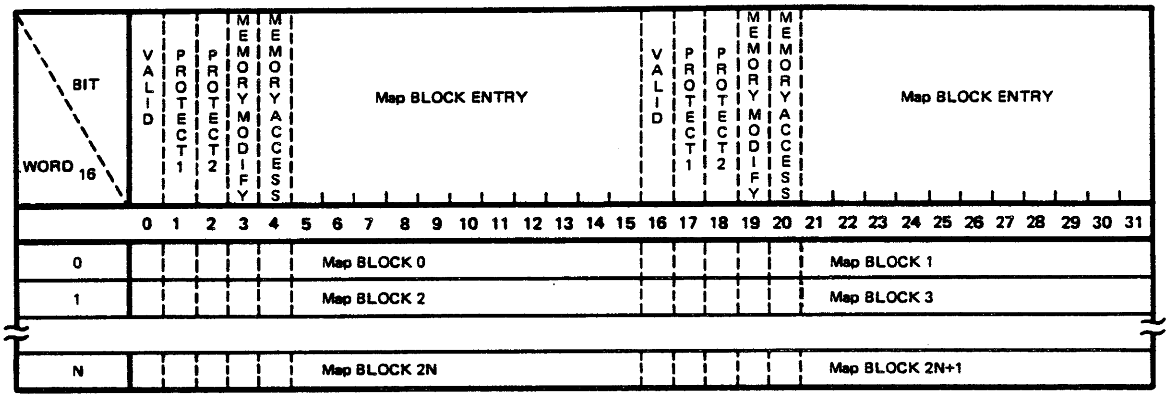
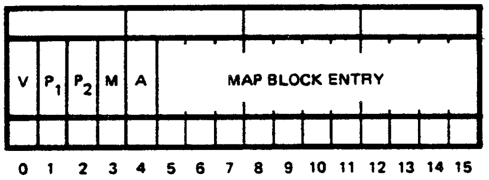


Figure 3-3. MAP Image Descriptor List



BIT 0 = 0 INVALID MAP BLOCK ENTRY
= 1 VALID MAP BLOCK ENTRY

1 - 2 = PROTECT BITS

PRIVILEGED BIT (PSW)	BIT 1 P1	BIT 2 P2	
0	0	0	NO ACCESS ALLOWED
0	0	1	NO ACCESS ALLOWED
0	1	0	READ/WRITE/EXECUTE
0	1	1	READ/EXECUTE ACCESS ONLY
1	0	0	READ/WRITE/EXECUTE
1	0	1	READ/EXECUTE ACCESS ONLY
1	1	0	READ/WRITE/EXECUTE
1	1	1	READ/EXECUTE ACCESS ONLY

3 = 0 A FIRST WRITE (MODIFY) TO THE MAP BLOCK HAS NOT OCCURRED
= 1 A FIRST WRITE (MODIFY) TO THE MAP BLOCK HAS OCCURRED

4 = 0 A FIRST READ OR WRITE (ACCESS) TO THE MAP BLOCK HAS NOT OCCURRED
= 1 A FIRST READ OR WRITE (ACCESS) TO THE MAP BLOCK HAS OCCURRED

5 - 15 = 11 MOST SIGNIFICANT BITS OF THE 24-BIT PHYSICAL ADDRESS
FOR THIS 2K WORD BLOCK

850880

Figure 3-4. MAP Image Descriptor (MID)

The MAP block entry in the MID may or may not contain valid information for address translation. The MAP valid bit (bit 0), when set, indicates that the MAP block entry is valid. A look-aside buffer technique (see paragraph 3.9) is used to determine whether a MAP block entry is valid. If the entry is valid, a MAP hit occurs and logical to physical address translation is performed. If the entry is not valid, a demand page fault results and software is notified of the faulting MAP register.

The protect bits (bits 1 and 2) are used in combination with the privileged bit to determine memory access. Figure 3-4 defines these bit combinations.

The modify bit (bit 3), when set, indicates that a first write to the MAP block has occurred. The access bit (bit 4), when set, indicates that either a first read or a first write to the MAP block has taken place. The modify and access bits must be set for MIDs associated with MSD 0.

The MAP block entry bits (bits 5 through 15) contain the 11 most-significant bits of the 24-bit physical address of the MAP block.

3.8 Map Initialization

When a new PSD is being entered into the CPU, the CPU is faced with one of three possible actions relating to the MAP.

1. When the unmapped mode is set, the CPU deactivates the MAP for the duration of the execution of this PSD. (An unmapped indication in the PSD overrides the load program status doubleword and change MAP (LPSDCM) instruction.)

2. When the LPSD instruction is used to load the PSD and the mapped mode is set, the CPU activates the MAP circuitry and uses whatever is in the MAP.

3. With the exception of the two preceding cases, the entry of a new PSD into the CPU results in new information being loaded into the MAP.

The CPU remembers the number of MAP entries that have been loaded and will not allow the process to access an entry in the MAP above that number. If a logical address of the process causes the CPU to generate a MAP index that is greater than that number, the CPU will assert the MAP fault trap.

3.9 The Look-Aside Buffer

The CPU has the capability of addressing 16 MB of memory when operating in the mapped environment. The intent of the look-aside buffer technique is to minimize load MAP time (context switch time) without seriously impacting memory access time.

When entering the mapped environment, firmware determines the need to load MSD 0. If MSD 0 is required the firmware fetches the MSD 0 related MIDs and writes them into the MAP. The page address of the last MID described by MSD 0 is loaded into the MSD 0 page limit cell located in scratchpad. The CPIX page limit cell in scratchpad and the CPIX base address located in scratchpad must be loaded with their appropriate values.

This process is handled entirely by the firmware. The CPIX page limit cell contains the highest numbered MID available to the active task. The CPIX base address cell contains the starting address of the CPIX's MIDL minus the depth of the MIDL for MSD 0.

In the V9 CPU, the MSD 0 page limit cell, the CPIX page limit cell, and the CPIX base address are loaded into registers located on the look-aside buffer boards, instead of being loaded into scratchpad.

When an operand or an instruction address is sent to the look-aside buffer for translation, the page address field (bits 08-18) is sent to the MAP array and to the hit MAP RAM. If the MAP entry is available and has no errors associated with it, the translated address is passed on to memory. If there are errors detected, the memory reference is aborted and the appropriate bit in the TRAP register is set.

If the MAP register is not loaded, the MAP miss sequencer adds the CPIX base address value to the missing page address to find the location in memory of the missing CPIX MAP entry. The resulting address is sent to memory as a standard operand read request. The word returned contains the required MID and the adjacent MID and is loaded into the MAP register array. The associated MAP hit/miss flags are set to hit. The original memory reference operation is reinitiated by the hardware and the normal sequence is continued. One set of hit MAP RAMs exist in the V6 CPU. Two sets of hit MAP RAMs (foreground/background) exist in the V9 CPU.

At context switch time, the firmware examines the new PSD to determine the need for a MAP reload. If the retain bit (bit 47 of the new PSD) is set, the contents of the hit/miss RAM and MAP are retained. The CPIX base address and CPIX page limit registers are not changed. If the retain bit is not set and the new PSD is a result of the load PSD and change MAP (LPSDCM) instruction, the hit/miss RAM is cleared and the process described above is repeated. MSD 0 is not affected by this clear routine and remains ready for the next context switch.

In the V9 CPU, the background hit RAM is switched to foreground, and the other hit RAM begins clearing in the background in preparation for the next context switch.

The operating system must be aware of the need to maintain the CPIX image in memory as long as the task is active, because the look-aside buffer needs to reference that table whenever the MAP miss occurs.

CHAPTER 4

INTERRUPTS AND TRAPS

4.1 Introduction

This chapter describes the trap and interrupt structure of the CPU. Interrupts and traps are defined, methods of handling are described, and formats are provided. Traps and interrupts report asynchronous or synchronous events to the software. Interrupts are requests that are generated external to the CPU, whereas traps are either internally generated error conditions or requests, such as supervisor call, which warrant an immediate response. The events that caused the trap or interrupt can be generated asynchronously by hardware or synchronously scheduled by software when a trap or interrupt control instruction is executed. A trap or interrupt causes the transfer of control (context switching) to a trap or interrupt handler.

4.2 Interrupts

Interrupts are the means by which real-time events, external to the CPU, are reported to software. The notification of these events is prioritized, scheduled and, in some cases deferrable. An individual interrupt request may result from an asynchronous event that was originally initiated under software control. Interrupts must contend for recognition by the CPU. Of those interrupts contending for recognition by the CPU at any given time, only the highest priority interrupt will be recognized and executed. This activity is a hardware function that is transparent to the software. There is no hardware stack in the CPU for pending interrupts. Only the highest priority interrupt level is recognized by the CPU hardware. Devices seeking service that do not have the highest priority level must continue to assert their interrupt priority level until they become the highest priority interrupt requesting service.

Interrupts are always associated with one of the following:

1. Extended I/O channels (class F).
2. Nonextended I/O channels (class 3 and E).
3. Real-time option module (RTOM) or input/output processor (IOP), user defined, real-time interrupts.
4. Real-time clock.
5. Interval timer (class 3).

It is important to note the distinction between the terms interrupt level and interrupt request. An interrupt level is the prioritized seven-bit number assigned to individual SelBUS devices (i.e., extended and nonextended I/O channels, RTOM, IOP, and interval timer). When a SelBUS device wants to contend for access to the CPU, it drives this seven-bit number on discrete SelBUS lines provided for this purpose. This is known as the priority interrupt level polling sequence. This process is transparent to both software and the CPU hardware. At the end of the polling sequence, the highest priority interrupt level will be driving the interrupt level lines. The lower priority level will stop contending. At this point, the SelBUS device driving the highest priority level will drive the interrupt request line, provided that level has not been activated by software (the activated condition is defined later). This interrupt request actually interrupts the CPU. Once the CPU firmware is ready to respond to the interrupt request, it will then read the interrupt level associated with the current interrupt request and use this number to address the CPU scratchpad. The CPU scratchpad contains information, termed an interrupt entry, which further identifies the device that has made the interrupt request. There are 112 prioritized interrupt levels available in the CPU. They are assigned to devices external to the CPU that must interact directly with software. Table 4-1 shows the interrupt assignments and their relative priority. Priority 00 is the highest and priority 6F is the lowest. The purpose of an interrupt request, once it is recognized, is to cause the firmware to:

1. Temporarily suspend execution of the current routine.
2. Save specific data relevant to the suspended routine.
3. Vector to the appropriate routine for servicing the interrupt.

Software will then:

1. Execute the interrupt service routine.
2. Restore specific data relevant to the suspended routine.
3. Return to the suspended routine at the point of interruption and continue.

4.3 Interrupt Control Instructions

Macro level instructions are provided to control and schedule the reporting of interrupts. The interrupt control instructions and other interrupt related instructions are listed and briefly described below. These instructions are presented in three groups:

1. Interrupt control instructions for nonextended I/O, RTOM, and IOP real-time interrupts.
2. Interrupt control instructions for extended I/O channels.
3. Interrupt related instructions.

4.3.1 Interrupt Control Instructions For Non-Extended I/O, RTOM, and IOP Real-Time Interrupts

The interrupt control instructions for nonextended I/O, RTOM and IOP real-time interrupts are privileged. None of these instructions in this group affects extended I/O (class F) devices.

4.3.1.1 Enable Interrupt Instruction (EI)

The enable interrupt instruction enables the interrupt level specified in the instruction.

4.3.1.2 Disable Interrupt Instruction (DI)

The disable interrupt instruction disables the interrupt level specified in the instruction and clears any unserviced interrupt request associated with that interrupt level.

Table 4-1
Default Interrupt Vector Locations

Interrupt Vector Table

Relative Priority	Default Interrupt Vector Location (IVL)	Associated Interrupt
00	100	External/software interrupt 0
01	104	External/software interrupt 1
02	108	External/software interrupt 2
03	10C	External/software interrupt 3
04	110	I/O channel 0 interrupt
05	114	I/O channel 1 interrupt
06	118	I/O channel 2 interrupt
07	11C	I/O channel 3 interrupt
08	120	I/O channel 4 interrupt
09	124	I/O channel 5 interrupt
0A	128	I/O channel 6 interrupt
0B	12C	I/O channel 7 interrupt
0C	130	I/O channel 8 interrupt
0D	134	I/O channel 9 interrupt
0E	138	I/O channel A interrupt
0F	13C	I/O channel B interrupt
10	140	I/O channel C interrupt
11	144	I/O channel D interrupt
12	148	I/O channel E interrupt
13	14C	I/O channel F interrupt
14	150	External/software interrupt
15	154	External/software interrupt
16	158	External/software interrupt
17	15C	External/software interrupt
18	160	Real-time clock interrupt
19	164	External/software interrupts
↓	↓	
6E	2B8	External/software interrupts
6F	2BC	*Interval timer interrupt

* The interval timer interrupt level always has the highest number (lowest priority level) assigned.

Highest interrupt level number, lowest priority = 6F

Extended I/O (F-Class) may occupy any interrupt level

Non-extended I/O E-Class (D-Class) must occupy levels 4 through 13_{16} (I/O channels 1 through F_{16})

4.3.1.3 Request Interrupt Instruction (RI)

This instruction causes an interrupt request signal to be generated for the interrupt level specified in the instruction. If the interrupt level is not enabled the request remains pending until enabled or cleared by a DI instruction.

4.3.1.4 Activate Interrupt Instruction (AI)

This instruction activates the interrupt priority level specified by the instruction regardless of whether that interrupt level is enabled or disabled. The active condition of the specified priority level blocks interrupts at that level and all lower priority levels from being serviced until the specified level is deactivated.

4.3.1.5 Deactivate Interrupt Instruction (DAI)

This instruction deactivates the interrupt level specified in the instruction regardless of whether the interrupt level is enabled or disabled. It does not affect any current or pending interrupt requests. The deactivate interrupt instruction and the instruction immediately following are executed as an uninterruptible pair.

4.3.2 Interrupt Control Instructions for Extended I/O Channels

The instructions in this group are privileged.

4.3.2.1 Enable Channel Interrupt Instruction (ECI)

This instruction enables the addressed channel to create interrupt requests. The channel should be initialized via SIO initialize channel (INCH) before executing an ECI to provide status address.

4.3.2.2 Disable Channel Interrupt Instruction (DCI)

This instruction disables the addressed channel from creating interrupt requests. The instruction does not clear unserviced interrupts.

4.3.2.3 Activate Channel Interrupt Instruction (ACI)

This instruction causes the addressed channel to contend for service by asserting its interrupt priority level, but prohibits the addressed channel from driving the interrupt request line once it gains access to the CPU. This blocks the addressed channel and all lower priority levels from requesting an interrupt. This instruction is executed regardless of whether the addressed channel is enabled or disabled.

4.3.2.4 Deactivate Channel Interrupt Instruction (DACI)

This instruction causes the addressed channel to remove its interrupt priority level from contention. If an interrupt request is already queued, it will begin actively requesting service provided it has been enabled. The deactivate channel interrupt instruction is non deferrable. The deactivate channel interrupt instruction and the instruction immediately following are executed as an uninterruptible pair.

4.3.2.5 Deferment

Extended I/O channels have the option of either accepting or deferring the following interrupt control instructions:

1. Enable channel interrupt (enables pending requests).
2. Disable channel interrupt.
3. Activate channel interrupt (blocks the same level and all lower level interrupts).

The condition codes are used to notify software of the acceptance or deferment of the interrupt control request. The software may either immediately execute the instruction again or queue it for execution later. This differs from interrupt control of non extended I/O devices where the CPU is required to wait until the interrupt control request is accepted.

4.3.3 Interrupt Related Instructions

4.3.3.1 Block External Interrupt Instruction (BEI)

This instruction prevents the CPU hardware from acknowledging any interrupt requests that are generated by I/O channels or RTOMs. In addition, the IPU trap and console attention trap are not acknowledged during blocked status. When executed in the IPU, the BEI is trapped as an undefined IPU instruction trap.

4.3.3.2 Unblock External Interrupts Instruction (UEI)

This instruction allows the CPU hardware to acknowledge all interrupt requests generated by I/O channels or RTOMs. In the IPU, this instruction clears blocking invoked by load program status doubleword (LSPD), context switching, etc.

4.3.3.3 Load Program Status Doubleword (LPSD)

This instruction causes the program status doubleword (PSD), addressed by the instruction, to be loaded into the PSD register in the CPU. Bits 48 and 49 of the PSD are used to specify whether interrupts are blocked or unblocked.

4.3.3.4 Load Program Status Doubleword and Change Map (LPSDCM)

This instruction causes the program status doubleword (PSD), addressed by the instruction, to be loaded into the PSD register. Bits 48 and 49 of the PSD are used to specify whether interrupts are blocked or unblocked. In addition, this instruction causes the MAP to be loaded provided certain conditions are met (refer to LPSDCM instruction in Chapter 6).

4.4 Interrupt Context Switching

Interrupt context switching occurs after an interrupt is recognized by the CPU. It is a process that involves capturing the parameters of the current operating environment (specified in the PSD), saving them for later use, and vectoring to the interrupt service routine.

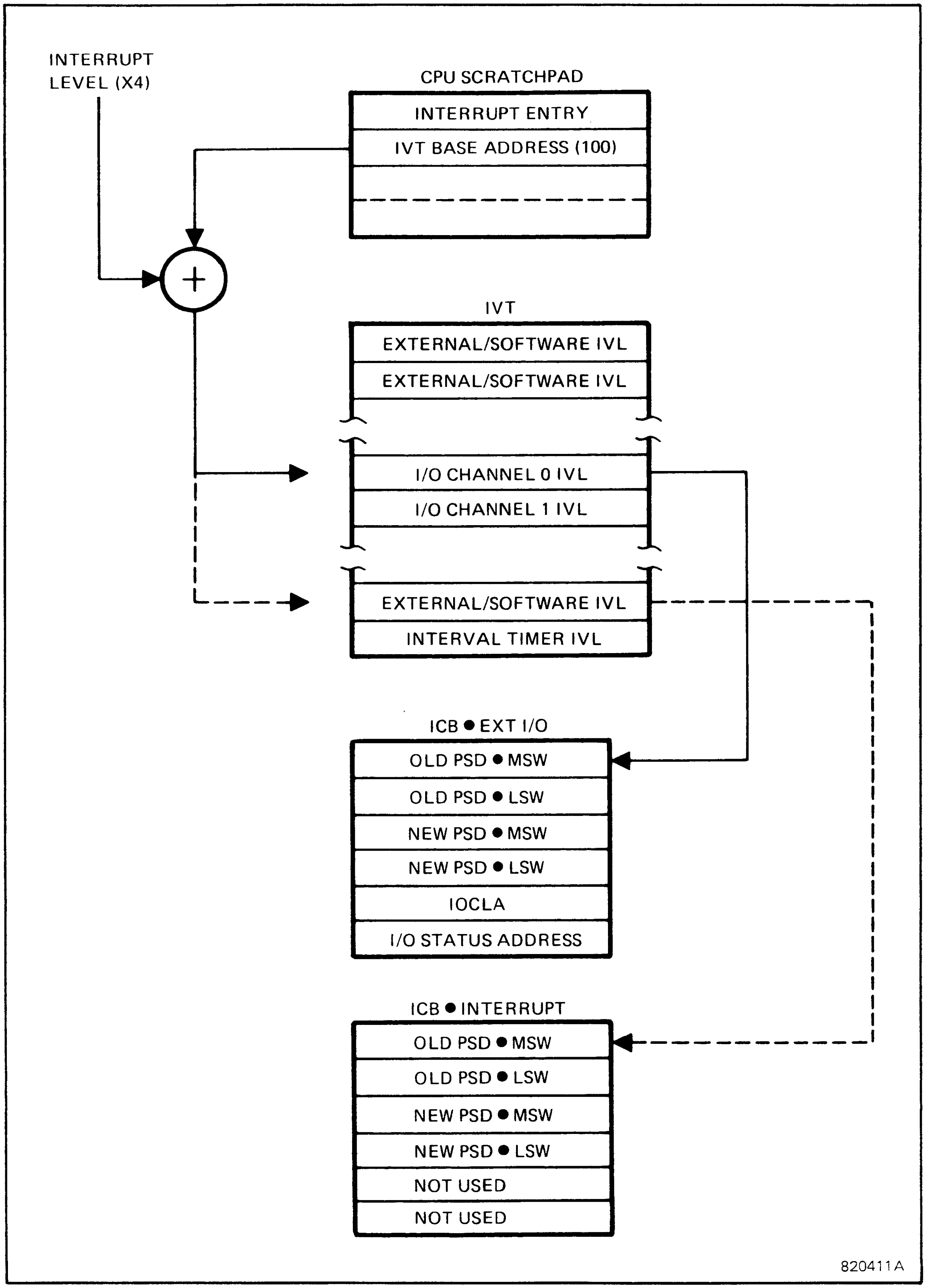


Figure 4-1. Interrupt Structure

The basic elements used to execute an interrupt context switch are the following:

1. CPU Scratchpad.
2. Interrupt vector table (IVT).
3. Interrupt vector location (IVL).
4. Interrupt context block (ICB).

The scratchpad is physically located in the CPU; however, the IVT, IVL, and ICB are located in main memory. These basic elements use a 24-bit real address. Figure 4-1 illustrates the interrelationship among these elements.

4.4.1 CPU Scratchpad

The scratchpad is illustrated in table 4-2. Locations $F0_{16}$ and $F1_{16}$ contain a trap vector table base address and an interrupt vector table base address respectively. These two tables are in the main memory and contain the real addresses of the trap and interrupt context blocks.

The CPU scratchpad contains interrupt entries and the base address of the interrupt vector table. The interrupt entry includes the physical address of the device that issued the interrupt, as well as other information about that device; the physical address is required in order to communicate with the device. The CPU uses the base address of the IVT and the interrupt level to calculate the address of the desired location within the IVT. The base address of the IVT may be assigned by software; if it is not, then the CPU uses the default address of X'100' as the IVT base address. Once a software assignment is made, a system reset does not reestablish the default address if software loads the correct scratchpad keyword.

The purpose of the device entry (see figure 4-2) is to map a software channel/device address into a SelBUS physical address. The interrupt entry (see figure 4-3) is used to map a software interrupt level number into a SelBUS physical address.

4.4.2 Interrupt Vector Table (IVT)

The interrupt vector table (IVT), whose base address is in the scratchpad, contains a pointer or vector address for each assigned interrupt level. Each vector address points to the main memory location of the ICB associated with a given interrupt. To find the correct IVL, the CPU aligns the interrupt level (received with the interrupt request) on a word boundary (effectively multiplying it by 4) and adds this value to the base address of the IVT. The result is a main memory address (IVL within the IVT) that contains the address of the correct ICB. Table 4-1 lists the default addresses for all IVLs.

4.4.3 Interrupt Context Block (ICB)

An interrupt context block consists of six consecutive word locations in memory. Refer to figures 4-4 and 4-5. The first two word locations provide a place to store the old PSD. This contains parameters of the CPU operating environment that existed when the program was interrupted as well as the program count which indicates the point of return after interrupt servicing is completed. The third and fourth words always contain the new PSD. The new PSD is used to establish the operating environment for the interrupt service routine and to supply the address (program count) of the first instruction in that service routine. The fifth and sixth words of the ICB are only used in ICBs that are associated with extended I/O interrupts. For extended I/O ICBs (refer to Figure 4-5), word five provides the input/output command list address (IOCLA), which is a 24-bit

Table 4-2
Scratchpad

Memory Image Address	Scratch-pad Address	Function	Default Value	Decimal
300	00	Channel - Device 00		2, 5
		Device Entries		
4FC	7F	Channel - Device 7F		2, 5
500	80	Interrupt Level 00		2, 5
		Interrupt Entries		
6BC	EF	Interrupt Level 6F		
6C0	F0	Trap Vector Table Address	80/20	1, 2, 3
6C4	F1	Interrupt Vector Table Base Address	100	2, 4
6C8	F2	IOCD Base Address (Class E)	700	2, 4
6CC	F3	Master Process List (MPL) Base Address	788	2, 3
6D0	F4	CPIX Base Address/Default IPL Address		6
6D4	F5	Current PSD2		8
6D8	F6	Reserved		
6DC	F7	Scratchpad Key=X'ECDAB897' (CPU) X'13254768' (IPU)		
6E0	F8	Identify Device Protocol DRT		5, 7
6E4	F9	CPU Status Word		7
6E8	FA	Current Active Interrupt		7, 8
6EC	FB	Number of Active Interrupts		6, 7
6F0	FC	Auto IPL DVC Address or 0 for manual IPL		6
6F4	FD	Reserved		8
6F8	FE	Reserved		8
6FC	FF	Interrupt Level 7F=00FFFFFF (pseudo IPL interrupts)		6

Note 1. Default value is 80 for CPU, and 20 for IPU.

Note 2. Denotes locations that must be provided by software for ICL.

Note 3. The Trap Vector Table and the Master Process List must reside in the first 128K words of main memory.

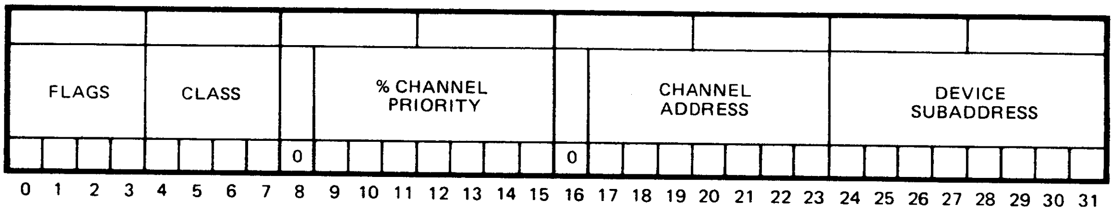
Note 4. The Interrupt Vector Table and Input/Output Command Doubleword must reside in the first 128K words of main memory.

Note 5. Not used in the V9 IPU.

Note 6. Not used in the IPU.

Note 7. During Power Failure, these scratchpad locations are used for a different purpose. See paragraph on Power Fail Trap (4.5.1.1)

Note 8. V6 only. Used as a temporary work location in V9.



FLAGS (UNARY, BITS 0-3)

BIT 0	RAM LOADED
BIT 1	PROGRAM VIOLATION (CLASS 0, 1, 2, AND E)
BIT 2	ENABLE CHANNEL WCS EXECUTED (CLASS F)
BIT 3	NOT USED

CLASS (BINARY, BITS 4-7)

VALUE

3	IOP RTOM INTERVAL TIMER (CPU ONLY)
7	EXTENDED I/O (CLASS F) IPU CONSOLE IOP (IPU ONLY)
B	IPU CONSOLE IOP INTERVAL TIMER (CLASS 3,IPU ONLY)
E	STANDARD CD-TD I/O (CPU ONLY)
F	EXTENDED I/O (CPU ONLY)

CHANNEL PRIORITY LEVEL (BINARY, BITS 8-15)

BIT 8	ALWAYS ZERO
BITS 9-15	SERVICE INTERRUPT PRIORITY (ONES COMPLEMENT)

CHANNEL ADDRESS (BINARY, BITS 16-23)

BIT 16	ALWAYS ZERO
BITS 17-23	CHANNEL PHYSICAL ADDRESS (NOTE: IPU CONSOLE IOP PHYSICAL ADDRESS MUST BE DIVISIBLE BY 2)

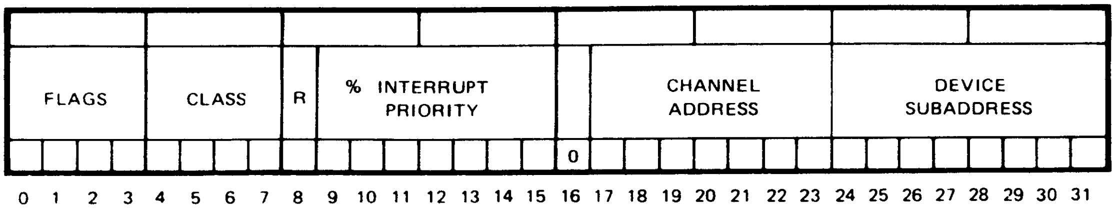
DEVICE SUBADDRESS (BINARY, BITS 24-31)

BITS 24-31 DEVICE SUBADDRESS

NOTE: FOR F-CLASS AND 7-CLASS THE DEVICE SUBADDRESS MUST BE ZERO FILLED.

820722

Figure 4-2. Scratchpad I/O Device Entry Format



FLAGS (UNARY, BITS 0-3)

BIT 0	RAM LOADED
BIT 1	ENABLE CHANNEL WCS EXECUTED (CLASS F)
BIT 2	INTERRUPT ACTIVE
BIT 3	INTERRUPT ENABLED

CLASS (BINARY, BITS 4-7)

VALUE

3	IOP/RTOM INTERVAL TIMER (CPU ONLY)
6	IPU/IOP NON-I/O INTERRUPTS (IPU ONLY)
7	EXTENDED I/O (CLASS F) IPU CONSOLE IOP (IPU ONLY)
B	IPU CONSOLE IOP INTERVAL TIMER (CLASS 3,IPU ONLY)
E	STANDARD CD-TD I/O (CPU ONLY)
F	EXTENDED I/O (CPU ONLY)

R(BIT 8) = 1 IOP/RTOM INTERRUPT (NON-I/O) (CPU/IPU)
= 0 I/O INTERRUPT (CPU/IPU)

INTERRUPT PRIORITY LEVEL (BINARY, BITS 9-15)

BITS 9-15 SERVICE INTERRUPT PRIORITY LEVEL (ONES COMPLEMENT)

CHANNEL ADDRESS (BINARY, BITS 16-23)

BIT 16	ALWAYS ZERO
BITS 17-23	CHANNEL PHYSICAL ADDRESS (NOTE: IPU CONSOLE IOP PHYSICAL ADDRESS MUST BE DIVISIBLE BY 2)

DEVICE SUBADDRESS

BITS 24-31 DEVICE SUBADDRESS

NOTE: FOR F-CLASS AND 7-CLASS THE DEVICE SUBADDRESS MUST BE ZERO FILLED.

820723A

Figure 4-3. Scratchpad Interrupt Entry Format

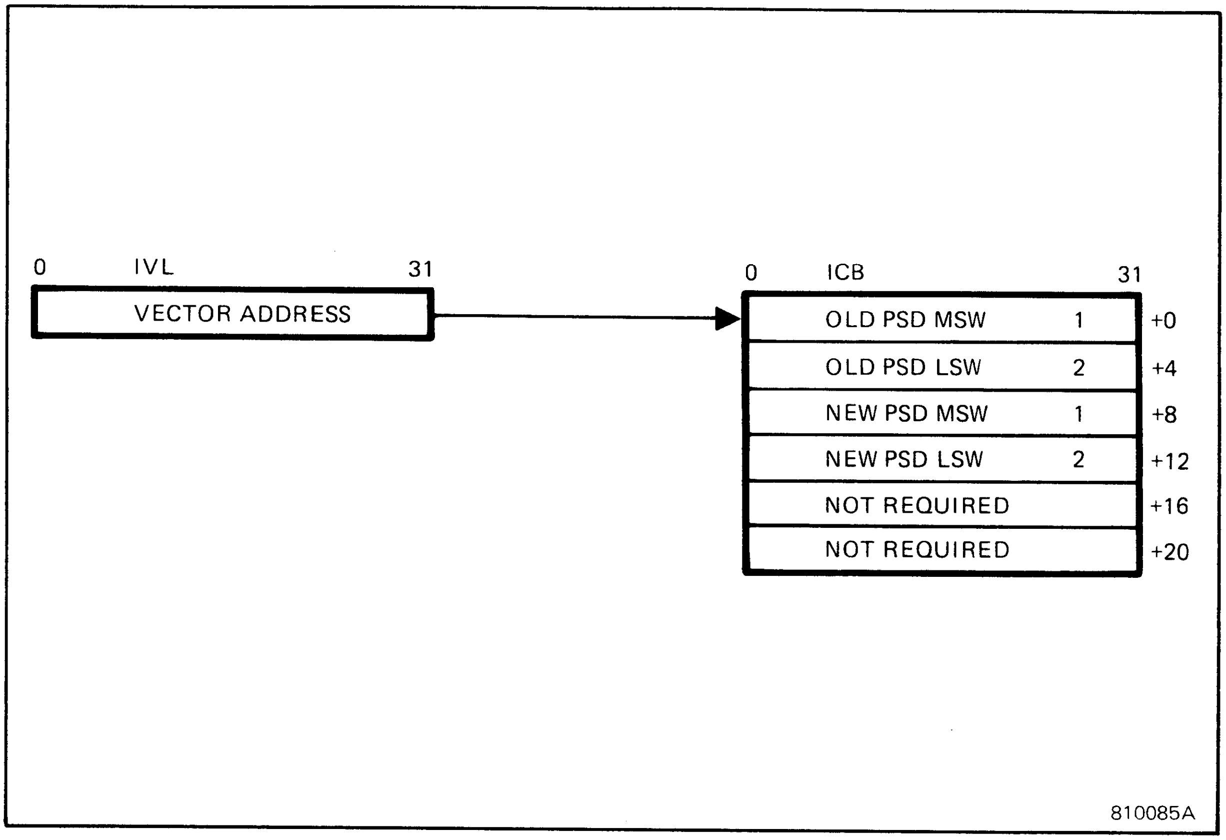


Figure 4-4. Interrupt Context Block Format - External and Nonextended I/O Interrupts

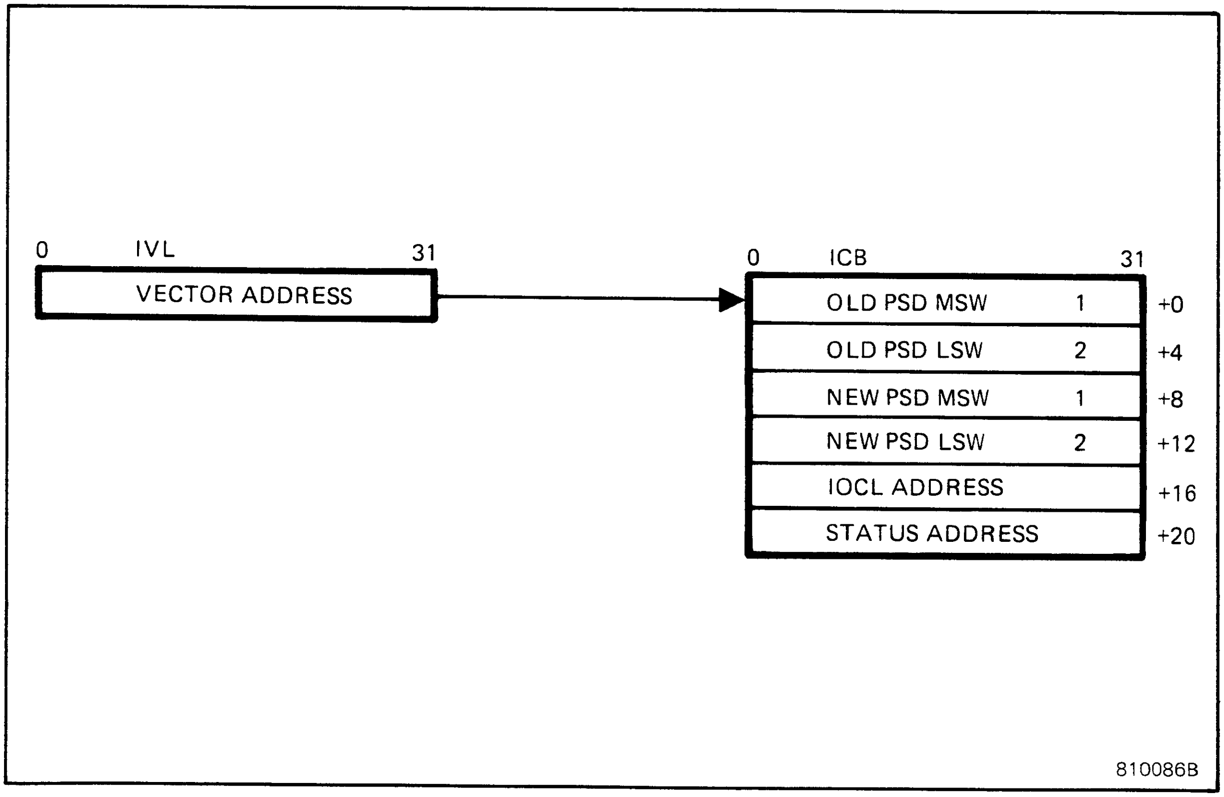


Figure 4-5. Interrupt Context Block Format - Class F (Extended) I/O Interrupts

address, for the associated extended I/O channel. This word must be set up in the ICB by software prior to the execution of either the start I/O (SIO) instruction or the write channel WCS instruction (when applicable). The IOCL address is transmitted to the I/O channel by the CPU during the start I/O or write channel WCS SelBUS sequences.

Also, for extended I/O ICB (refer to Figure 4-5), the sixth word contains the 24-bit real address of the channel status word. Whenever the channel reports status to the CPU software, the channel stores the channel status word in main memory. The CPU then stores the physical memory address of this channel status word in word six of the ICB.

The channel may report status when any one of the following events occurs:

1. An interrupt is acknowledged (a hardware function).
2. A start I/O instruction is executed (SIO)
3. A test I/O instruction is executed (TIO)
4. A halt I/O instruction is executed (HIO)

When status is stored during a start I/O, test I/O, or halt I/O instruction, the channel rejects the instruction, and the CPU condition codes are set to reflect the status stored condition. Under the status stored condition, the channel clears its status pending flags, as well as any interrupt pending flags that are relative to the status reported.

4.5 Traps

Traps are error conditions that are identified and reported by the CPU. All traps have the same priority with the exception of the power fail trap, which overrides all other traps and interrupts. Traps are not deferrable except for the IPU trap and console attention trap, which are deferred while the CPU is in the blocked mode.

4.5.1 Trap Types

Traps are listed below and described in the paragraphs that follow.

1. Power fail trap (power-down)
2. Power-on trap
3. Memory parity trap
4. Nonpresent memory trap
5. Undefined instruction trap
6. Privilege violation trap
7. Supervisor call trap
8. Machine check trap
9. System check trap
10. MAP fault trap
11. IPU undefined instruction trap
12. Signal CPU or Signal IPU trap
13. Address specification trap
14. Console attention trap
15. Privileged mode halt trap
16. Arithmetic exception trap
17. Cache Error Trap (V9 only)
18. Demand page fault trap

4.5.1.1 Power Fail Trap

The power fail trap is caused by a power fail signal from the system power distribution subsystem. This trap is nondeferrable. During the power-down sequence, all I/O channels are master cleared; therefore, they are unusable to software in the power fail trap handler. Main memory will remain operational for a minimum of 500 microseconds following a power fail trap. In addition, the power fail trap is disabled to prevent a second power fail trap from occurring during the power fail trap sequence. The execution of the LPSD, or LPSDCM instruction reenables this trap.

When the battery backup option is installed, the memory locations listed below are loaded and saved:

Locations		Contents
V6	V9	
6DC *	6DC *	CPU Scratchpad Keyword X'ECDAB897'
6D4 *	—	IPU Scratchpad Keyword X'13254768'
6E0	6E0	CPU Configuration Word
6E4	6D8	CPU Status Word
6E8	6E8	IPU Status Word
6EC	—	IPU Configuration Word

* During power down, the CPU and IPU scratchpad keywords are not rolled out to memory.

4.5.1.2 Power-On Trap

In power-on sequences, where auto restart and auto IPL cannot be executed, the CPU executes an automatic trap halt. The power-on trap is generated under two circumstances. If the scratchpad image has memory errors or the scratchpad image does not contain the scratchpad keyword and auto IPL does not occur, the power-on trap halt is generated. If the scratchpad image is correct and auto restart is attempted without the traps being enabled, the power-on trap is generated. The execution of LPSD or LPSDCM enables the power fail trap.

4.5.1.3 Memory Parity Trap

The memory parity trap is caused by memory parity errors or uncorrectable memory errors that are encountered on any of the following types of memory fetches:

1. Instruction fetch.
2. Operand fetch.
3. Indirect fetch.

Memory locations containing errors are not cached. Memory errors detected by prefetching are not reported until execution time.

4.5.1.4 Nonpresent Memory Trap

A nonpresent memory trap is caused by any of the following conditions:

- Instruction fetch from nonpresent memory
- Operand or indirect fetch from nonpresent memory
- Operand write to nonpresent memory
- Memory fetch data return transfer (DRT) timeout (76.8 microseconds for V6 and 38.4 microseconds for V9)
- Instruction fetch memory (DRT) timeout (76.8 microseconds for V6 and 38.4 microseconds for V9)

Memory reads of nonpresent memory are not cached. If nonpresent memory is detected during prefetch, the error condition is not reported until execution time. A write to nonpresent memory is cached and firmware must purge (clear) cache following each write to nonpresent memory.

4.5.1.5 Undefined Instruction Trap

The undefined instruction trap is caused by the following conditions:

1. An undefined instruction operation code.

2. An undefined instruction augment operation code, or subopcode field.

3. A defined fullword instruction operation code that is encountered in a right halfword.

4. Class F (extended I/O) instructions with invalid suboperation code fields. (The suboperation code field of a class F instruction is located in bits 9 through 12 of the instruction word.)

4.5.1.6 Privilege Violation Trap

The privilege violation trap is caused by three different events as follows:

1. If a memory store is directed to a write protected logical memory address. This can only occur in the mapped environment. Memory can be write-protected from either an unprivileged process, or globally, as defined by the Map Image Descriptor (MID) corresponding to the logical address. Bits 1 and 2 of the MID determine when writes are permitted.

2. If an attempt is made to execute a privileged instruction while the CPU is in an unprivileged state.

3. If an unprivileged user attempts to access a mapped location whose P1 bit is reset.

4.5.1.7 Supervisor Call Trap

The supervisor call trap is caused by execution of the supervisor call instruction (SVC).

4.5.1.8 Machine Check Trap

A machine check trap results whenever a firmware sequence is broken by an error condition that would otherwise be reported as a trap if software encountered the equivalent error. It is a hard failure because the firmware cannot guarantee the state of the CPU. The CPU is halted and diagnostics should be run to determine the cause of the problem.

A class of machine check trap errors consists of SelBUS protocol violations during an interrupt sequence. The causes of this type of machine check trap are as follows:

1. Class 3 and E channels during an interrupt sequence

   a. I/O no response (lack of a transfer acknowledge).

   b. Advance and final transfer retry or busy timeout exceeds 1.382 milliseconds.

   c. Ready timeout exceeds 307.2 microseconds.

   d. Final transfer I/O channel busy.

   e. Data return transfer (DRT) timeout exceeds 76.8 microseconds for V6 CPU and 38.4 microseconds for V9 CPU.

2. Class F channel during an interrupt sequence

   a. Advance transfer no response (lack of transfer acknowledge).

   b. Advance and final transfer retry or I/O channel busy timeout exceeds 1.382 milliseconds.

   c. Ready timeout exceeds 38.4 microseconds.

   d. Final transfer no response (lack of a transfer acknowledge).

   e. Data return transfer (DRT) timeout exceeds 76.8 microseconds for V6 CPU and 38.4 microseconds for V9 CPU.

4.5.1.9 System Check Trap

A system check trap is caused if software attempts to force the CPU into an illogical sequence. The specific type of error that caused the trap is described by the trap status word stored in the trap context block. The errors that cause a system check trap are divided into four groups as described in the following paragraphs.

4.5.1.9.1 System Check Trap - Group 1

System check trap - group 1 errors consist of SelBUS protocol violations that occur during class F (extended I/O) bus communication sequences. Class F sequences that pertain to interrupt processing are excluded from group 1 type system check traps (these errors are included in machine check trap).

The causes of class F bus protocol violations that are included in group 1 are the following:

1. Ready timeout exceeds 38.4 microseconds.
2. Final transfer no response.
3. Final transfer retry timeout exceeds 38.4 microseconds.
4. Final transfer I/O channel busy.
5. Data return transfer timeout exceeds 76.8 microseconds for V6 CPU and 38.4 microseconds for V9 CPU.

4.5.1.9.2 System Check Trap - Group 2

System check trap - group 2 includes two types of class F (extended I/O) channel protocol errors, as follows:

1. The execution of a write channel writable control storage instruction (WCWCS) without being preceded by an enable channel writable control storage instruction (ECWCS).

2. The execution of a reset channel instruction (RSCHNL) that results in one of the following conditions:

   a. Receipt of a SelBUS transfer acknowledge after the first bus transfer, but no transfer acknowledge following the second bus transfer.

   b. Receipt of a SelBUS transfer acknowledge after the first bus transfer, but a ready timeout occurs that exceeds 38.4 microseconds.

4.5.1.9.3 System Check Trap - Group 3

System check trap - group 3 errors result from the following causes:

1. An extended I/O instruction directed to a class 3 or class E device.
2. A command device instruction directed to a class F device.

4.5.1.9.4 System Check Trap - Group 4

System check trap - group 4 errors result from the following causes:

1. MAP Write (LMAP) in mapped mode.
2. LPSD error.
3. Load MAP - Operand memory error
4. I/O classification of device entry in scratchpad is incorrect

4.5.1.10 Map Fault Trap

The events which can cause a MAP Fault Trap are the following:

1. The MAP load algorithm does not load any MAP registers (underflow).

2. The MAP load algorithm attempts to load more MAP block entries than there are MAP registers (overflow).

3. Whenever a logical address exceeds the allocated logical address space (MAP limit violation).

4. In the nonbase register mode, if an instruction fetch is attempted above the first 128K words of the logical address space.

5. In the V9 only, in the mapped environment, the CPU causes the map fault trap if a map parity error is detected.

4.5.1.11 IPU Undefined Instruction Trap

In the V6 IPU, this trap will occur when the block external interrupt (BEI) instruction or class 3, E, or F I/O instructions are attempted without the proper class in the scratchpad's device/interrupt entry location.

In the V9 IPU, this trap is caused when the block external interrupt (BEI) instruction or any input/output (I/O) instruction is attempted.

4.5.1.12 Signal CPU/Signal IPU Trap

This trap is caused when the signal IPU (SIPU) instruction is executed. When executed in the CPU the trap is set in the IPU, when executed in the IPU it is set in the CPU. It is deferrable by the CPU/IPU when interrupts are blocked.

4.5.1.13 Address Specification Trap

An address specification error occurs in the CPU if an attempt is made to read a doubleword operand from, or write it to, an odd GPR or an odd word address, or if the data type specified by a memory reference instruction is not legal for that instruction. The CPU executes an address specification trap under the following conditions:

1. The effective F, C0, and C1 bits of a memory reference instruction operand address are 0, 1, and 0, respectively, and bit 29 of the effective operand address is 1. This represents an attempt to reference a doubleword operand on an odd word boundary.

2. An attempt is made to read or write a doubleword operand to/from GPR 1, 3, 5, or 7. Note that this does not preclude using these registers as the source in an MPR, DVR, or NORD instruction, since these treat the source register as a single word operand.

3. A memory reference instruction attempts to use a combination of effective F and C bits which is not included in the following table of permissible data types:

INSTRUCTION	F	C0	C1
ALL BRANCHES	0	X	X
EXM		0	XX
SBM, ZBM, ABM, and TBM	1	X	X
LPSD and LPSDCM	0	0	0
LF, STF, LFBR, STFBR, LBR and STBR	*	0	0
ALL FLOATING POINT	*	X	0
ALL OTHER INSTRUCTIONS	X	X	X

(X = Don't Care)

*These instructions have an implicit direct F bit of 0.

4. In Base Register mode, the byte/halfword/word/doubleword alignment of the effective address must match that of the original instruction word.

5. A multiple-operand instruction (LF, STF, LFBR, STFBR, CALL, CALLM, RETURN, BSUB, and BSUBM) crosses a MAP block boundary. This applies when in either mapped or unmapped mode.

When the CPU hardware calculates the effective address of these instructions, it forces all direct F bits in the indirect address chain to 0 before determining the effective F and C bits. If the effective F and C bits of the modified indirect chain do not match the permissible combinations in the table, an address specification trap will occur.

4.5.1.14 Console Attention Trap

The console attention trap is activated by the attention command from the console. Although it is handled as a trap for servicing, the console attention trap acts much like an interrupt in certain instances. Traps can override a blocking condition and only interrupts are affected; however, the console attention trap is masked when blocking is invoked in the CPU. Also, when interrupts are blocked, such as from a deactivate interrupt instruction, the console attention is masked. When a console attention trap occurs, the trap remains disabled until a LPSD or LPSDCM is executed.

4.5.1.15 Privileged Mode Halt Trap

If a privileged user tries to execute a halt instruction or its equivalent (e.g., the target of an execute instruction contains all zeros) when the privileged mode halt trap is enabled, the CPU will trap. The privileged mode halt trap is enabled or disabled by setting or resetting bit 23 of the CPU status word via the SETCPU instruction. Firmware determines whether this trap is enabled or disabled by looking at the bit 23 of the CPU status.

4.5.1.16 Arithmetic Exception Trap

Whenever an arithmetic or shift operation results in an overflow or underflow condition, an arithmetic exception (AE) is raised. The enable arithmetic exception trap and the disable arithmetic exception trap instructions are used to either set or reset bit 7 of the PSD to enable or disable the AE.

When the CPU has detected an AE, this AE is reported via the condition codes from the current instruction. The CPU will trap if the AE trap is enabled. The program counter contents may be used to identify the instruction that caused the exception.

If the AE trap is disabled and the execution of any floating point instruction results in an overflow or underflow the CPU firmware modifies the destination register. Section 6.2.10.2 (page 324) contains the values placed in the destination register for overflow or underflow in both the positive and negative direction.

4.5.1.17 Cache Fault Trap (V9 Only)

The Cache Fault Trap occurs when one of the following errors is detected.

1. Instruction fetch cache out bus parity error.
2. Operand read cache out bus parity error.
3. Cache index parity error.

4.5.1.18 Demand Page Fault Trap

A demand page fault trap occurs when a memory access references a location in the MAP where the MAP valid bit is not set. Memory access includes instruction fetch, operand fetch, store, indirect, and LEAR instructions.

For instruction fetches, the old PSD points to the logical program counter value which caused the fault. For operand fetches, the old PSD points directly at the instruction which caused the fault.

4.5.2 Trap Halts

The CPU provides for automatic trap halts if the software has not enabled the traps with the enable traps option of the SETCPU instruction. All IPU traps are enabled during the power up sequence or system reset sequence.

The traps that arm the trap halt logic are the following:

1. Memory parity error trap.
2. Nonpresent memory trap.
3. Undefined instruction trap.
4. Privilege violation trap.
5. Machine check trap.
6. System check trap.
7. MAP fault trap.
8. Address specification trap.
9. Power fail trap (power-down trap).
10. Power-on trap.
11. Console attention trap.
12. Halt instruction trap.

13. Cache fault trap.
14. Demand page fault trap.

The traps that do not arm the trap halt logic are the following:

1. Supervisor call trap.
2. Arithmetic exception trap.

Other conditions that arm the automatic trap halt logic are the following:

1. Memory error in power up.
2. I/O or memory error in initial program load.

4.5.3 Trap Halt Implementation

When a trap halt occurs, the following conditions and information exist:

1. The CPU is halted.
2. The INTERRUPT light on the turnkey panel is illuminated.
3. The program counter (PC) portion of the PSD1 contains the dedicated memory address (trap vector location) for the trap causing the halt.
4. Starting at memory location 680 (hexadecimal), the following error information is stored:

Location	Contents
00680	Error PSD1 (CPU)
00684	Error PSD2 (CPU)
00688	CPU trap status word
0068C	Most recently used interrupt entry
00690	Error PSD1 (IPU)
00694	Error PSD2 (IPU)
00698	IPU trap status word

(IPU traps are normally enabled, but software can disable software sensing of IPU traps.)

4.5.4 Trap Related Macroinstructions

The trap related instructions are listed and briefly described below. For a more complete description of each instruction and its format, refer to chapter 6 of this manual.

4.5.4.1 Supervisor Call

The supervisor call activates the supervisor call trap and causes the CPU to vector to the dedicated memory location (trap vector location) for the supervisor call trap.

4.5.4.2 Enable Arithmetic Exception Trap

The enable arithmetic exception trap instruction sets bit 7 of the PSD to enable the arithmetic exception trap.

4.5.4.3 Disable Arithmetic Exception Trap

The disable arithmetic exception trap instruction resets bit 7 of the PSD to disable the arithmetic exception trap.

4.5.4.4 Set CPU Mode

The set CPU mode instruction can enable or disable software handling of all traps. If all traps are disabled, the automatic trap logic is armed, and any subsequent trap will cause a CPU automatic trap halt.

4.5.5 Trap Context Switching

Trap context switching occurs after a trap is detected by the CPU or IPU. The process involves capturing the parameters of the current operating environment (specified in the program status doubleword), saving them, and vectoring to the trap handler.

The following basic elements are used to execute a trap context switch:

1. CPU or IPU scratchpad.
2. Trap vector table (TVT).
3. Trap vector location (TVL).
4. Trap context block (TCB).

The scratchpad is physically located in the CPU and IPU; however, the TVT, TVL, and TCB are located in the main memory. The main memory addresses associated with the TVT, TVL, and TCB are 24-bit addresses. Figure 4-6 shows the interrelationship among these elements.

4.5.5.1 CPU Scratchpad

The scratchpad contains the base address of the trap vector table. This base address is used to calculate the address of the required TVL within the table. The base address of the trap vector table may be assigned by software; if it is not assigned, then the CPU uses the default address of X'80' and the IPU uses X'20' as the TVT base address. Once a software assignment is made, system reset does not reestablish the default address if the scratchpad keyword is present in the scratchpad.

4.5.5.2 Trap Vector Table (TVT)

The trap vector table, whose base address is in the scratchpad, consists of a series of trap vector locations (TVL). Each TVL contains a pointer or vector address and is associated with a particular trap type (refer to Table 4-3). This vector address either points to the trap context block associated with the particular trap that has occurred or, in the case of a supervisor call trap, it provides a basis for calculating a secondary vector address. This secondary vector address points to the appropriate trap context block and applies to the supervisor call traps only (refer to Figure 4-6).

Table 4-3
Default Trap Vector Locations

Trap Number	Default Trap Vector Location (TVL)-CPU	 (TVL)-IPU	Trap Condition
00	80	20	Power fail trap
01	84	24	Power on trap
02	88	28	Memory parity trap
03	8C	2C	Nonpresent memory trap
04	90	30	Undefined instruction trap
05	94	34	Privilege violation trap
06	98	38	Supervisor call trap
07	9C	3C	Machine check trap
08	A0	40	System check trap
09	A4	44	Map fault trap
0A	A8	48	Undefined IPU instruction trap
0B	AC	4C	Signal CPU or Signal IPU trap
0C	B0	50	Address specification trap
0D	B4	54	Console attention
0E	B8	58	Privilege mode halt trap
0F	BC	5C	Arithmetic exception
10	C0	60	Cache fault trap (V9 only)
11	C4	64	Demand page fault trap

4.5.5.3 Trap Context Block (TCB)

Trap context block formats are of three different types: supervisor call, demand page fault, and all other traps (refer to figures 4-7 through 4-9).

Words one through four are the same for all three formats. The first two words (old PSD) contain the CPU operating parameters that existed when the trap occurred. Words three and four (new PSD) establish the operating environment for the trap handler and supply the address (program count) of the first instruction in that handler.

For a supervisor call TCB, word five (bits 20 through 31) is used to store the call number of the supervisor call instruction which invoked the trap. For all TCBs not associated with supervisor call, word five is used to store the trap status word (see tables 4-4 and 4-5). This word is stored in the TCB after the trap is detected by the CPU. The trap status word contains additional descriptor bits for defining the error condition.

Word six, the page fault word, is used only by the demand page fault TCB (see figure 4-10). This word (bits 21 through 31) contains the map register number (logical map block number) of the faulting block. The fault page word (bit 0) also indicates whether the trap was caused by an instruction fetch or an operand access. For instruction fetches, the old PSD in words one and two points to the logical program counter value which caused the fault. For operand fetches, the old PSD points directly at the instruction which caused the fault.

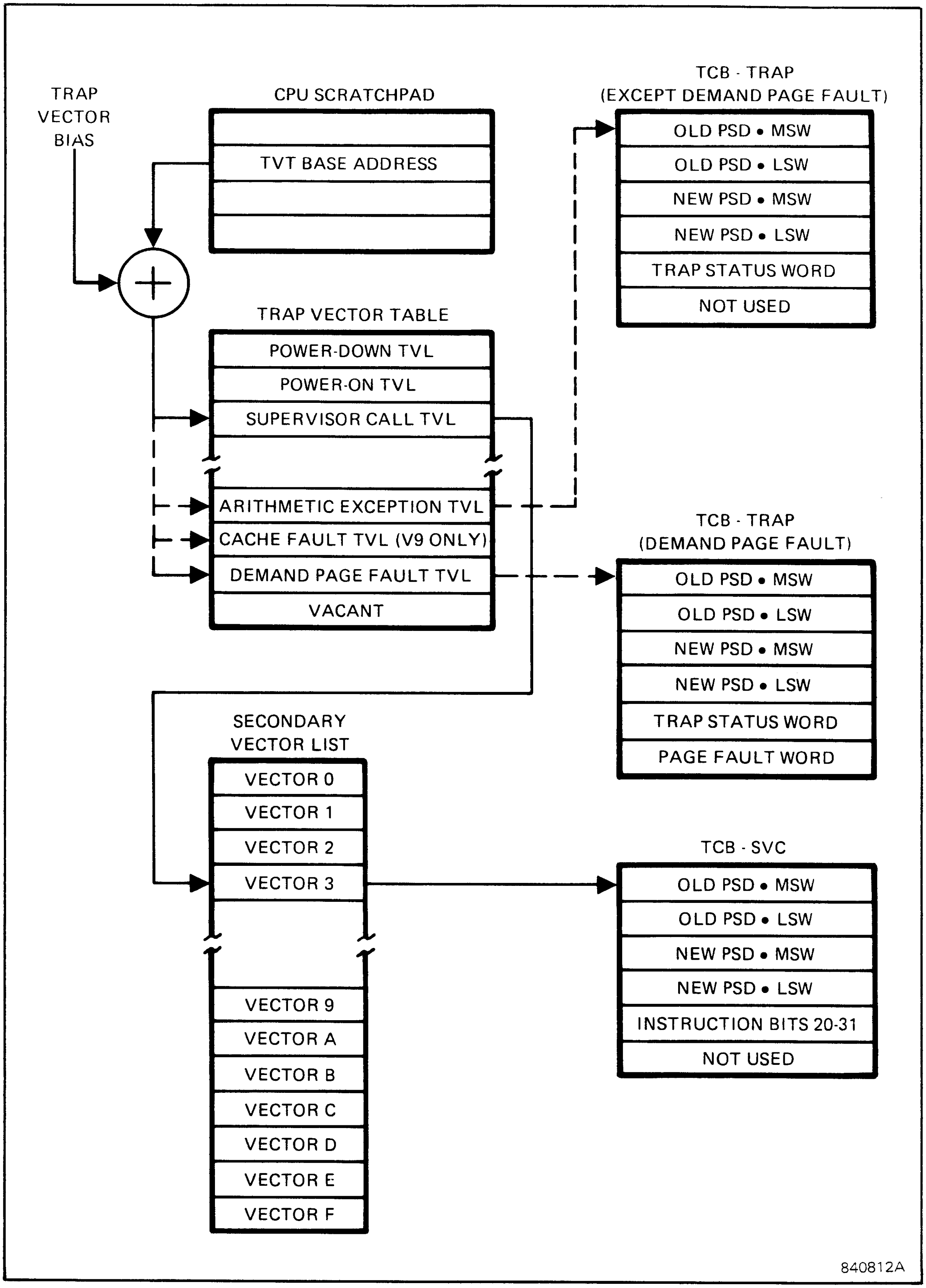


Figure 4-6. Trap Structure

4.5.6 ICB/TCB Formats

Interrupt and trap context block formats consist of six words; however, for most traps or interrupts, only four or five of the words are used. Figures 4-4, 4-5, 4-7, 4-8, and 4-9 illustrate the ICB/TCB formats:

1. External, nonextended I/O format.
2. Class F (extended) I/O format.
3. Supervisor call format.
4. Trap format (other than demand page fault)
5. Trap format (demand page fault).

4.5.6.1 Old and New PSD

The first four words of all context block formats are identical in that they contain the old PSD followed by the new PSD.

The old PSD is stored in the context block whenever an interrupt or trap is asserted by the CPU. The old PSD provides CPU context information current at the time a particular trap or interrupt occurred. The program count points to the interrupted instruction plus one.

In the case of traps, PSD bit 31 is set to indicate that the last instruction executed was a right halfword instruction and bit 30 (also applicable to interrupts) is set to indicate that the next instruction to be executed is a right halfword instruction.

The new PSD contains the necessary information to set the hardware and software in the appropriate context for serving the interrupt.

4.5.6.2 External and Nonextended Format

The external interrupts and nonextended I/O interrupts ICB format is used with all RTOM interrupts, CD, and TD I/O interrupts. RTOM interrupts include the interval timer and the real time clock interrupt (refer to Figure 4-2).

4.5.6.3 Trap Format

The fifth word of the TCB format contains the trap status word. This word is stored in the TCB at the time a trap occurs. The status word may provide additional descriptor bits for defining the error condition (refer to Table 4-4 for V6 and Table 4-5 for V9).

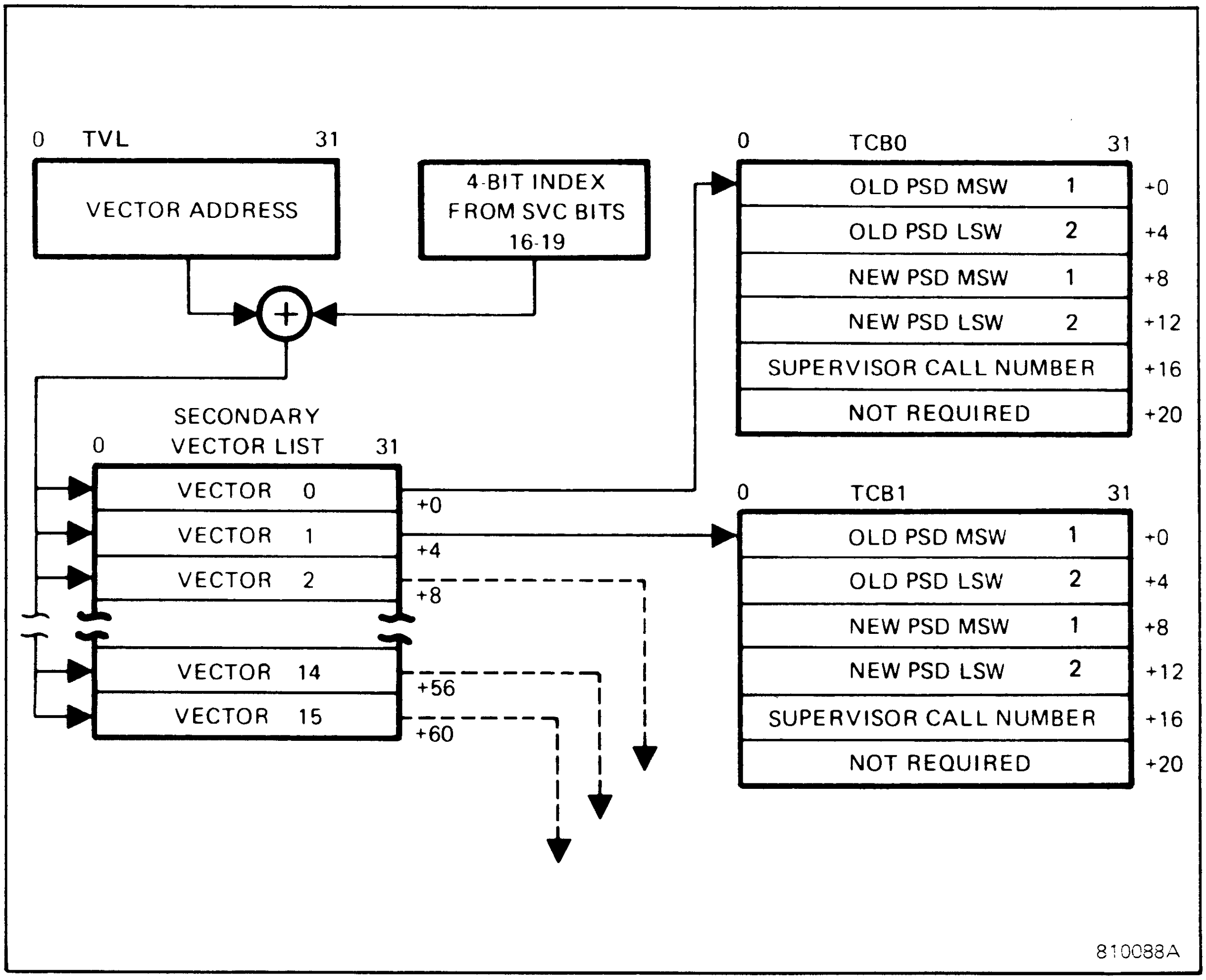


Figure 4-7. Trap Context Block Format - Supervisor Call (SVC)

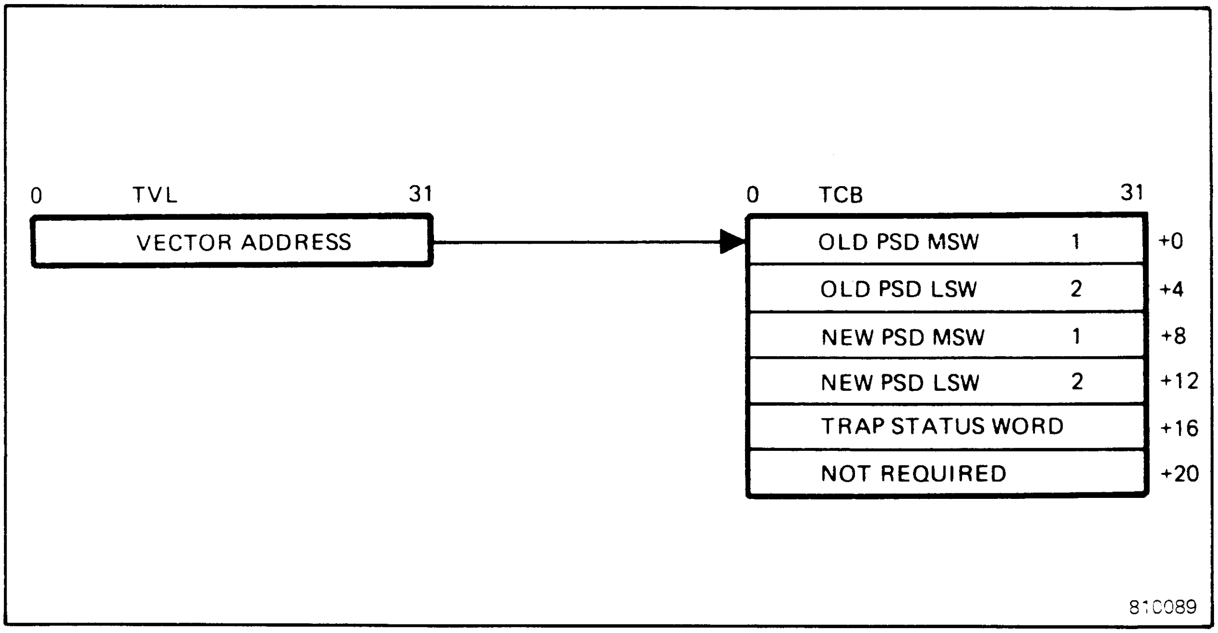


Figure 4-8. Trap Context Block Format (Except Demand Page Fault)

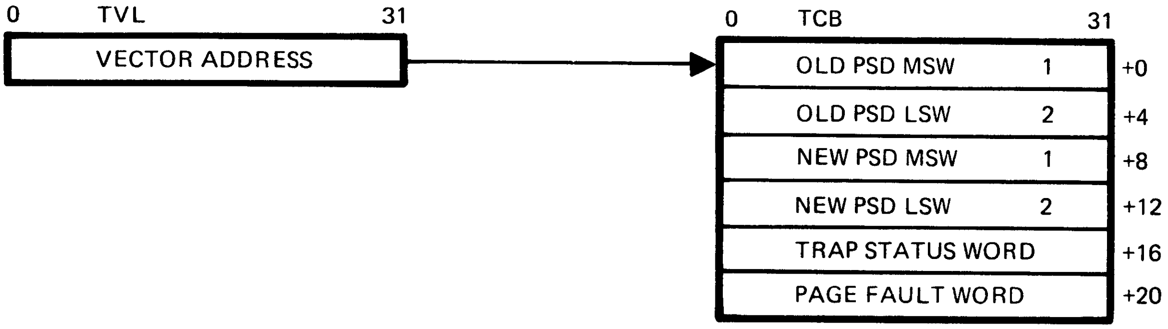
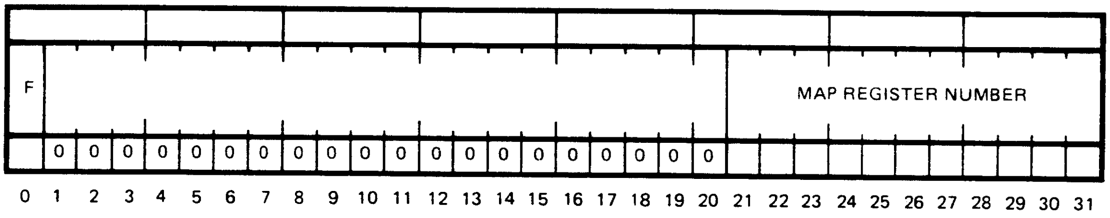


Figure 4-9. Trap Context Block Format (Demand Page Fault)



BIT 0 = 0 THE MAP FAULT WAS CAUSED BY AN INSTRUCTION FETCH
= 1 THE MAP FAULT WAS CAUSED BY AN OPERAND ACCESS

BITS 1-20 ALWAYS ZERO

BITS 21-31 MAP REGISTER NUMBER (LOGICAL MAP BLOCK NUMBER)

831512

Figure 4-10. Page Fault Word Format

4.5.6.4 Class F I/O Format

The fifth word of the ICB provides the input/output command list (IOCL) address for the associated Class F I/O channel. This word is set up in the ICB by software prior to the execution of either a start I/O or write channel WCS instruction. The IOCL address is transmitted to the I/O channel by the CPU during the start I/O or write channel WCS SelBUS sequences.

The sixth word of the class F I/O ICB contains the 24-bit real address of the channel status word. Whenever the channel reports status to the CPU (and software), the channel stores the channel status word in memory. The CPU then stores the memory address of the channel status word into the word six of the ICB.

The channel may report status when any one of the following events occurs:

1. An interrupt is acknowledged (a hardware event).
2. A start I/O instruction is executed.
3. A test I/O instruction is executed.
4. A halt I/O instruction is executed.

When a status is stored during a start I/O, test I/O, or halt I/O instruction, the channel rejects the instruction, and the CPU condition codes are set to reflect the status stored condition. Under the status stored condition, the channel clears its status pending flags, as well as any interrupt pending flags that are relative to the status just reported (refer to Figure 4-3).

4.5.6.5 Supervisor Call Format

The supervisor call (SVC) trap may have up to 16 TCBs.

The address of a specific TCB is obtained by adding a 4-bit index value from bits 16 through 19 of the SVC instruction to the 24-bit address that is in the SVC trap vector location (TVL). The sum of these values provides a 24-bit real address of a secondary vector location. The contents of the secondary vector location is the 24-bit real address of the appropriate supervisor call TCB.

Words one through four of a supervisor call TCB are provided for the old and new PSDs. Word five of the SVC TCB contains the SVC call number. Bits 20 through 31 of the SVC instruction are used by the CPU to set up word five of the SVC TCB.

Table 4-4
V6 CPU Trap Status Word

Note: The trap status word reports trap error status.

Bit	Description
0	=0, Class E I/O error; =1, Class F (extended I/O) error
1	=0, I/O processing error; =1, Interrupt/trap processing error
2	Final SelBUS transfer error
3	SelBUS no response error (no transfer acknowledge)
4	I/O Channel busy or busy status bit error
5	Ready timeout error
6	I/O DRT timeout error
7	Retry count exhausted error
8	Operand fetch memory parity error
9	Instruction fetch memory parity error
10	Operand nonpresent memory error
11	Instruction nonpresent memory error
12	Map memory protect violation (Read=0, Write=1)
13	Memory fetch DRT timeout error
14	Reset channel error
15	Channel WCS not enabled error
16	Map register load underflow/overflow
17	Unexplained memory error
18	LPSD, LPSDCM instruction error or map miss error
19	Privilege violation error
20	Map operand invalid access or wait instruction with interrupts blocked error
21	Scratchpad formatting error
22	Map instruction invalid access error
23	CPU is in the halt mode (for power-down trap)
24	Trap condition related to current process
25	CPU traps are software enabled
26	Imprecise No Transfer Acknowledge (store to non-present Memory)
27	0 = CPU; 1 = IPU
28	Error during MAP load

Bit 29	Bit 30	Bit 31	Description
0	0	0	32/55 CPU
0	0	1	32/75 CPU
0	1	0	32/27 CPU
0	1	1	32/67 CPU
1	0	0	32/87 CPU
1	0	1	32/97 CPU
1	1	0	V6 CPU
1	1	1	V9 CPU

Table 4-5
V9 Trap Status Word by Trap Class (Sheet 1 of 2)

Trap Class	Trap Number	Status Bit	Meaning
Power Fail	00	—	N/A
Power On	01	—	N/A
Memory Parity Error	02	00	Instruction Fetch Parity Error
		01	Operand Fetch Parity Error
Non-Present Memory	03	00	Instruction Fetch Non-present Mem.
		01	Operand Fetch Non-present Mem.
		02	Operand Write Non-present Mem.
		03	Operand Fetch DRT Timeout
		04	Instruction Fetch Memory DRT Timeout
Undefined Instruction	04	00	Undefined Instruction
		01	Invalid Intr Level (Intr CTRL INS)
Privileged Violation	05	00	Privileged Violation
		01	MAP Protect Violation on Memory Write
		02	Illegal Map Access (read/write protect violation)
SVC	06	20-31	SVC Call Number
Machine Check	07	00	Interrupt or Trap Sequence Error
		01	Micro Control Store Parity Error
		02	Map Parity Error (Instruction Fetch)
		03	Map Parity Error (Operand Read)
		04	Map Parity Error (Operand Write)
		05	Map Registers Overflow
		06	Map Doubleword Boundary Error
		07	LD Map Operand Memory Error
		08	Polling Interrupt Level Invalid
		09	'LINTR' Malfunction
		10	Old PSD Store/New PSD Fetch Error
		11	Unexplained Hardware Failure
		12	Machine Check Processing Fails
		16	I/O No Response
		17	Retry Count Exhausted
		18	Wait For Ready Timeout
		19	Final Transfer Error
		20	I/O Busy
		21	Wait For I/O DRT Timeout
		22	Micro Store Branch Error
System Check	08	00	Extended I/O Error
		01	N.U.
		02	Final Transfer Error
		03	I/O No Response Error

Table 4-5
V9 Trap Status Word by Trap Class (Sheet 2 of 2)

Trap Class	Trap Number	Status Bit	Meaning
System Check (Cont.)		04	I/O Busy
		05	Wait for Ready Timeout
		06	Wait for I/O DRT Timeout
		07	Retry Count Exhausted
		08	Reset Channel Error
		09	Channel WCS not Enabled
		10	N.U.
		11	N.U.
		12	Wait Instruction in Blocked Mode Error
		13	LPSD Error
		23	Scratch Pad Formatting Error
Map Fault	09	00	Map Invalid (Instruction Fetch)
		01	Map Invalid (Operand Read)
		02	Map Invalid (Operand Write)
		03	Attempt to Load '0' Map Registers
		05	Map Registers Overflow
		06	Map Doubleword Boundary Error
		07	LD Map Operand Memory Error
		08	LMAP in Mapped Mode
		09	L-Board Map Update Error (Operand)
		10	L-Board Map Update Error (Instruction)
		11	Memory Fetch DRT Timeout
		12	Memory Fetch Nonpresent
		13	Memory Parity Error
		14	Map In Bus Parity Error
IPU Undefined	0A	01	Undefined instruction to IPU
IPU	0B	—	Start IPU or IPU finished
Address Specification	0C	—	Address Specification Error
Console Attention	0D	—	Console Attention
Halt Trap	0E	—	Privileged Mode Halt Trap
Arithmetic Exception	0F	—	Arithmetic Exception
Cache Fault	10	00	Cache Out Bus Parity Error (Instruction Fetch)
		01	Cache Out Bus Parity Error (Operand Read)
		02	Cache Index Parity Error
Demand Page	11	—	Demand Page

NOTE: Bits 28-31 of the CPU Configuration Word are merged with these Trap Status Bits to show CPU Model Indicator.

CHAPTER 5

INPUT/OUTPUT SYSTEM

5.1 Introduction

This chapter provides a general description of the input/output (I/O) operations used by the CPU. I/O operations consist of transferring data in blocks of bytes, halfwords, or words between peripheral devices and the main memory. Once initiated, such transfers occur automatically, leaving the CPU free for other tasks.

This chapter also provides an overview of the I/O organization, including definitions of the major elements with an explanation of their functions. Particular attention is given to descriptions of the specific classes of I/O protocols. Formats are defined and illustrated for the types of controlling information used in conjunction with the I/O classes. Details of unique device-dependent features of I/O devices are more aptly detailed in the specific publication applicable to the device.

5.2 I/O Organization

Figure 5-1 depicts the I/O organization and the major components which participate in the I/O operations. The CPU communicates to all other modules of the I/O structure by the SelBUS. The IOP (input/output processor) as shown in Figure 5-1 is a specially designed logic board that provides RTOM capability, an interval timer, and a real-time clock. The IOP also serves as an interface between the CPU and IOP device controllers. The general term, I/O processor, used in this chapter refers to a channel, a controller, or a combination of channel and controller that is responsive to the corresponding I/O protocol class.

A channel serves as an intermediary between I/O device controllers and CPU/memory to handle the flow of information between I/O devices and memory. The channel receives requests over the SelBUS from the CPU as a result of I/O instructions that are executed.

I/O controllers contain the necessary electronics to interface with the channel and the I/O device, whether connected individually or via a common bus. A controller receives control information from the channel, provides the control buffering capabilities required to regulate the I/O devices, validates the command, and converts the I/O command to a form acceptable to the I/O devices. Often one controller is shared by several devices. Multiplexed controllers operate multiple devices in parallel, while others operate one device at a time.

Input/output devices provide a means of communication between the computer system and an external media.

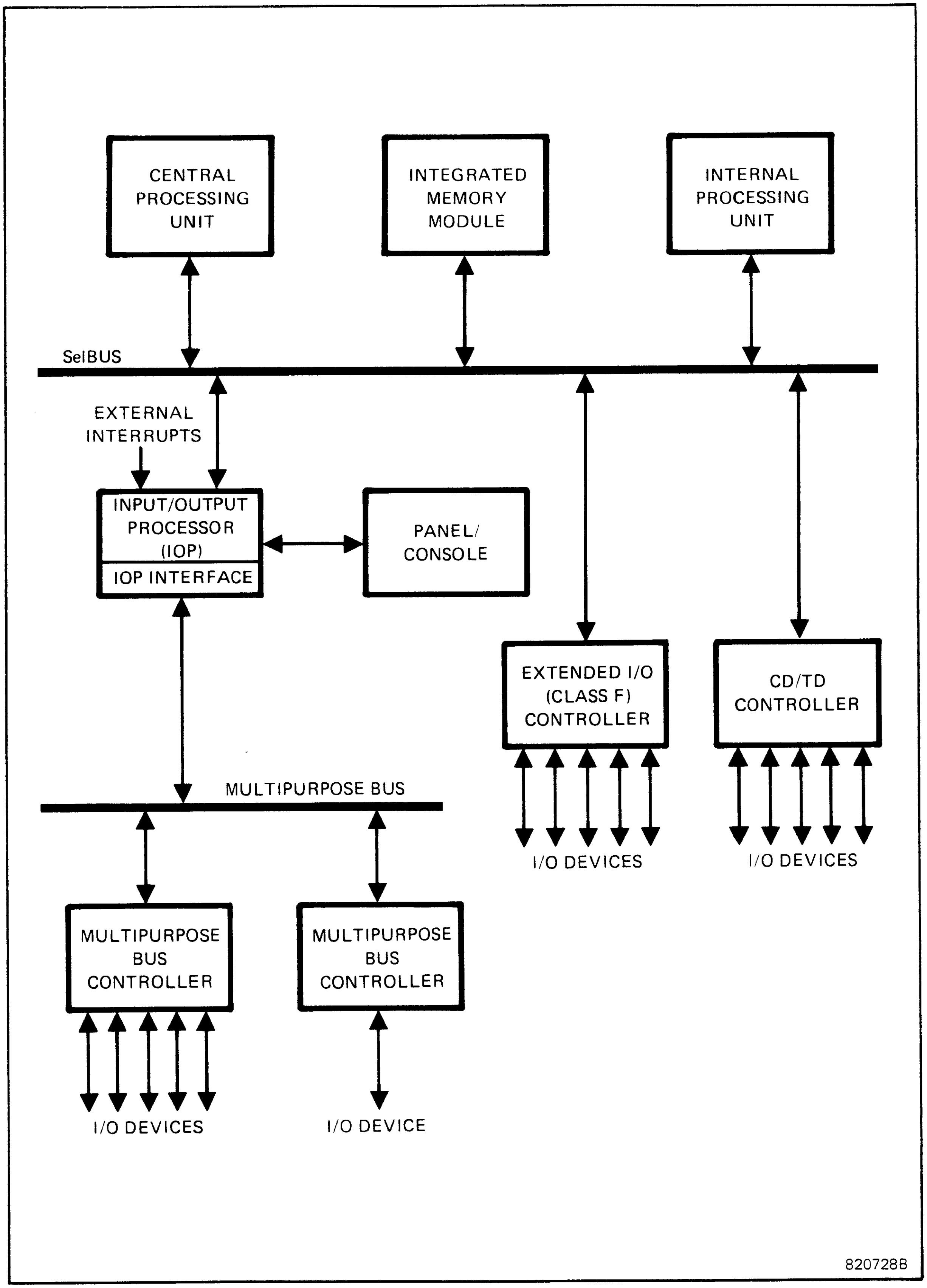


Figure 5-1. Major Elements of the I/O Organization

5.3 I/O Classifications

The V6 CPU supports class 3, E, and F I/O operations. The V6 IPU scratchpad device and interrupt entries use I/O classes that are subsets of the V6 CPU I/O and interrupt classes. The subset I/O class identification value is obtained by taking the standard class field from the device and interrupt scratchpad entries and ones-complementing the most significant bit of the four-bit class field. Therefore, class 7 is the subset of class F; class B is the subset of class 3, and class 6 is the subset of class E. In all cases, the subset I/O classes only relate to those I/O devices and interrupts that are controlled by the V6 IPU Console IOP.

The V9 is similar to the V6, except that the V9 IPU does not support any I/O.

Table 5-1 lists the classes along with the corresponding device types and the applicable instruction sets used. For class 3 or B protocol, the I/O processor can be either the IOP or the real time option module. Processors of class 3 or B are controlled by the command device instructions. The devices in class E or 6 are usually controlled by an input/output microprogrammable (IOM) processor. The I/O processors for this class are controlled by the command device and test device instructions. Class F or 7 I/O processors respond to the extended I/O instructions and have the capability of addressing memory throughout a 4M word (16M byte) range; in some cases class F or 7 I/O processors support an optional writable control storage (WCS) unit.

NOTE

> The device class is specified by the user during system generation (SYSGEN) and subsequently loaded into the CPU scratchpad. All device entries that are designated as class D will be converted to class E by software before the CPU scratchpad is loaded. Thus, the CPU hardware/firmware will handle all I/O operations that involve class D devices just as if they were class E devices.

Table 5-1
I/O Protocol Classes

V6/V9 CPU Protocol Class	V6 IPU Protocol Class	V9 IPU Protocol Class	Device Type	Instruction Set
3	B	NONE	IOP or RTOM Interval Timer	CD
E	NONE	NONE	HSD, etc.	CD, TD
F	7	NONE	IOP/RPU based designs, new channels	Extended I/O
NONE	6	NONE	IOP/RTOM external (non I/O interrupts)	EI, DI, etc.

5.3.1 Operation of Class 3 or B I/O Devices

The CPU interval timer contained in either the IOP or the RTOM module uses class 3 protocol. The class B I/O protocol is used by the V6 IPU Console IOP interval timer and its associated interrupt level. Class B is a subset of class 3, and obeys class 3 (CD) protocol rules. Class B must be used in the device and interrupt scratchpad entries for the V6 IPU Console IOP interval timer. The V9 IPU does not support I/O in any manner, nor does it have an IPU Console IOP.

5.3.1.1 Interrupt Level

Each class 3 or B I/O processor has a unique interrupt level number assigned to it. The interrupt level may be any one of the levels supported by either the V6 or V9 CPU, or the V6 IPU.

5.3.1.2 Subaddress

A class 3 or B I/O processor can have only one subaddress.

5.3.1.3 Interval Timer

The interval timer can be programmed by the CD instruction as follows:

1. Select one of four counting rates.
2. Select either single or multiple interrupts for a single count value.
3. Enable or disable the interval timer.
4. Write the initial count value from the V6 or V9 CPU or the V6 IPU general purpose register zero (GPR0) to the interval timer, or read the contents of the interval timer into V6 or V9 CPU or V6 IPU GPR0.

5.3.1.4 Command Device Instruction

The interval timer is controlled by the software command device (CD) instruction using a V6 or V9 CPU (or V6 IPU) register to/from I/O concept. The interval timer or CPU does not need to access memory during a CD instruction (other than normal instruction fetch); therefore, the execution time of the CD instruction is reduced. During a CD instruction execution, GPR0 is used to send data (initial count) to or receive status (current count) from the interval timer. Figure 5-2 illustrates the format of the CD instruction used to control the interval timer.

The interval timer does not respond to the following types of CD instructions:

1. CD terminate
2. CD transfer current word address
3. CD start I/O (initialize data transfer)

5.3.1.5 Test Device Instruction

The test device (TD) instruction cannot be used with the interval timer. A TD instruction executed to the interval timer at levels TD8000 or TD4000 will cause all condition codes to be set to zero. A TD instruction to the interval timer at the TD2000 level will indicate status transfer not performed (CC2).

5.3.1.6 Read the Interval Timer

To read the value in the interval timer, software executes a command device instruction to the interval timer. The current count will be transferred to GPR0.

5.3.1.7 Program the Interval Timer

To program the interval timer, software executes a CD instruction, and the contents of GPR0 are transferred to the interval timer. The interval timer can be loaded, under program control, with a 32-bit count value and a rate selection code. The rate selection code, bits 30 and 31 of the interval timer CD instruction, designates whether the programmed frequency, 120 Hertz, or the external clock is selected. If the interval timer is programmed to generate a single interrupt, the counter counts to zero, generates the interrupt, and continues to count negative. To determine the time elapsed after the interrupt was generated, a command device read interval timer instruction is executed.

5.3.2 Operation of Class E or 6 Devices

The class E I/O devices are usually controlled by an I/O controller that obeys or responds to CD and TD instructions. Class 6 describes the V6 IPU Console IOP real-time clock interrupts and external interrupts. Class 6 I/O identifies those interrupt levels that obey class E interrupt protocol (EI, DI, etc.) but are dedicated to operate with the V6 IPU Console IOP. The real time clock and external interrupts are classified as non-I/O interrupts in the V6 CPU, the V9 CPU, and the V6 IPU.

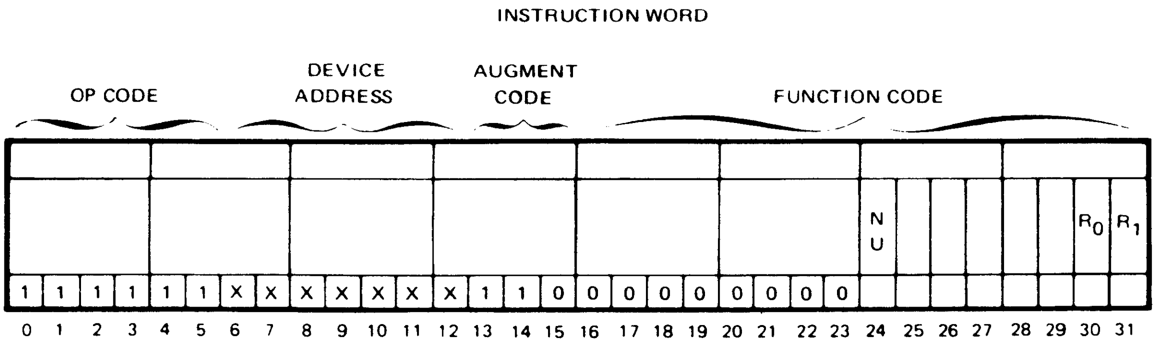
In the CPU scratchpad, non-I/O interrupt entries have a class field of zero. However, the class field is not verified. In the V6 IPU, non-I/O interrupts must have a class field equal to 6 indicating validation for IPU operation. The external interrupts (non-I/O) have only scratchpad interrupt entries (no device entries). The entry is defined as non-I/O when bit 8 is equal to a one.

5.3.2.1 Interrupt Level

Each class E or 6 I/O processor has a unique interrupt level number assigned to it. The interrupt level number must be a hexadecimal number between 04 and 13.

5.3.2.2 Subaddress

A class E or 6 I/O processor may have up to 16 subaddresses. Each subaddress may be associated with a software device address. However, for class E this is applicable only to the V6 IPU external interrupts because no I/O operations are permitted.



BIT 24 = 1 NOT USED

BIT 25 = 1 SPECIFIES READ TIMER CAUSING THE 32-BIT CONTENTS OF THE TIMER TO BE LOADED INTO GPR0

BIT 26 = 1 SPECIFIES PROGRAM INTERVAL TIMER AND THAT BITS 27-31 ARE VALID.

BIT 27 = 1 ENABLES (START) INTERVAL TIMER

= 0 DISABLES (STOP) INTERVAL TIMER

BIT 28 = 1 LOADS BITS 00-31 FROM THE GPR0 INTO THE INTERVAL TIMER BITS 00-31.

= 0 DOES NOT ALTER STORE COUNT.

BIT 29 = 1 GENERATE MULTIPLE INTERRUPT WHEN COUNT ZERO IS REACHED. THEN GENERATE INTERRUPT, RELOAD INITIAL COUNT, AND CONTINUE COUNTING.

= 0 GENERATE SINGLE INTERRUPT WHEN COUNT ZERO IS REACHED. THEN CONTINUE TO COUNT NEGATIVE.

BIT 30	BIT 31	SELECT COUNT RATE
0	0	SELECT HIGH FREQUENCY } APPLICABLE TO RTOM ONLY
0	1	SELECT LOW FREQUENCY } APPLICABLE TO RTOM ONLY
1	0	SELECT 120 HERTZ
1	1	SELECT EXTERNAL CLOCK

841479

Figure 5-2. Interval Timer Command Device Instruction Format

5.3.2.3 Command Device Instruction

The initiation of data or nondata transfers and the termination of I/O operations can occur as the result of the execution of a command device instruction in the CPU. The CD instruction, illustrated in Figure 5-3, specifies the device, the direction of transfer, and other controlling bits, especially the command code, required to condition the device for generating or accepting a transfer. When a class E I/O processor accepts the CD from the CPU, the control bits and command code are routed to the device addressed in the instruction.

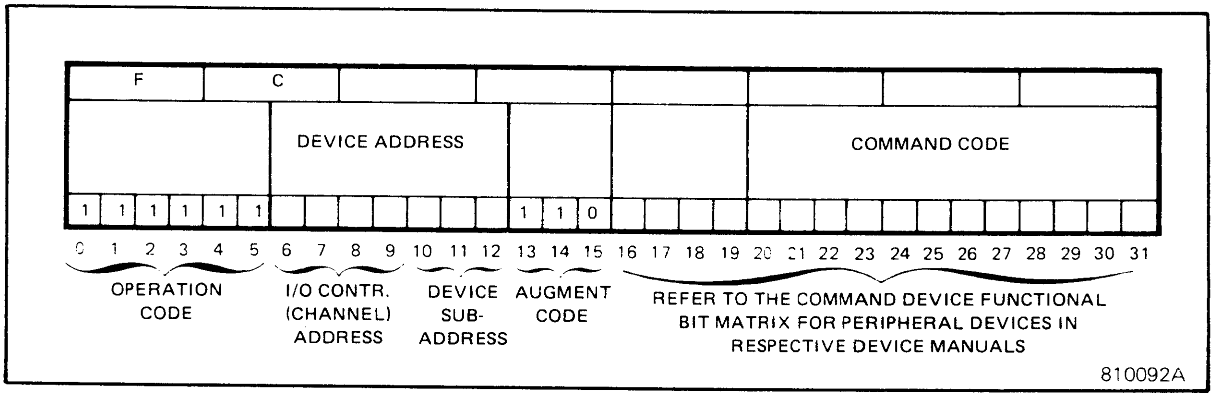


Figure 5-3. Class E Command Device Instruction Format

5.3.2.4 Transfer Control Word

For data transfers, a transfer control word (TCW) is used with the CD instruction to initialize a block of transfers. The TCW address must be set up by software before a CD instruction is executed. The TCW, illustrated in Figure 5-4, contains a 19-bit memory word address which defines where the block of transfers begins, and a 12-bit transfer count which defines the number of bytes, halfwords, or words to be transferred.

The format code (F and C bits) in the TCW defines the meaning of the TC field as summarized in Figure 5-4. The format code is designed such that when F is equal to one in a given TCW, the address is incremented in bit position 31 each time a transfer occurs. Therefore, each transfer is stored in, or read from, a consecutive byte in memory in this order:

Word N Word N+1

---Byte 0, Byte 1, Byte 2, Byte 3......Byte 0, Byte 1, Byte 2, Byte 3---

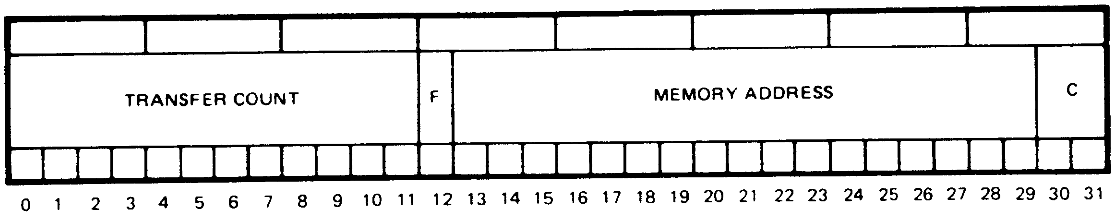
The proper binary value of the format code for accessing consecutive halfwords in memory is F equal to 0, and C equal to Y1; where Y equal to zero designates the left halfword and Y equal to one designates the right halfword. With this value of format code, the address is incremented in bit position 30 each time a transfer is made. This results in the desired accessing of consecutive halfwords.

The proper value of format code for consecutive word accessing is F equal to 0 and C equal to 00. When this value is present in a given TCW, the I/O controller increments the TCW in bit position 29 each time a transfer occurs.

Each time the address incremented, the transfer count is decremented. Therefore, the block length is always defined by the number of memory accesses and not by the number of words transferred.

Doubleword transfers are not supported.

CLASS E



BITS 0-11 DESIGNATE THE NUMBER OF TRANSFERS TO BE MADE BETWEEN MEMORY AND THE DEVICE CONTROLLER CHANNEL. THE TRANSFER COUNT IS WORDS, HALFWORDS, OR BYTES AS SPECIFIED BY THE 'F' AND 'C' BITS.

BITS 13-29 DESIGNATE THE MEMORY LOCATION FOR EACH TRANSFER. THE MEMORY ADDRESS IS EITHER A WORD, HALFWORD, OR BYTE ADDRESS AS SPECIFIED BY THE 'F' AND 'C' BITS.

BITS 12, 30, 31 (F AND C BITS) SPECIFY THE FORMAT CODE FOR THE TRANSFER.

FORMAT BITS			TRANSFER TYPE
F	C		
BIT 12	BIT 30	BIT 31	
0	0	0	WORD TRANSFER
0	Y	1	HALFWORD TRANSFER
1	X	X	BYTE TRANSFER

NOTES:

Y = 0 SPECIFIES LEFT HALFWORD
Y = 1 SPECIFIES RIGHT HALFWORD
XX = BYTE NUMBER AS FOLLOWS:

00 = BYTE 0
01 = BYTE 1
10 = BYTE 2
11 = BYTE 3

NOTE: 1. FOR TEST DEVICE LEVEL 2000 INSTRUCTIONS, THE TCW MUST SPECIFY A HALFWORD TRANSFER TO EITHER THE LEFT OR RIGHT HALFWORD.

2. SOME E-CLASS CHANNELS USE DIFFERENT INTERPRETATIONS OF THE TCW i.e. GPMC, HSD, ADI (ALL D-CLASS)

820727

Figure 5-4. Transfer Control Word Format

5.3.2.5 Input/Output Command Doubleword (IOCD)

The CPU firmware formats the first word of the IOCD. The second word of the IOCD is formatted by software and contains the TCW address. The doubleword is stored in a memory location that is dedicated to the I/O controller being operated by the command device instruction.

The specific IOCD format is a function of the type of device or controller being operated by the CD instruction and the type of I/O operation being initiated.

Figure 5-5 illustrates the IOCD format used with class E devices. Command device instruction bits 16 through 31, the command code, are formatted into the first IOCD word by the firmware, and the TCW dedicated memory address is formatted into the second IOCD word by the software. The class E I/O processor firmware obtains the contents of the TCW from memory for data transfer instructions.

Figure 5-6 illustrates the IOCD format for an initial program load (IPL) initiated by the IPL function of the class E I/O processor. The specific IOCD format is only used for the first read from the IPL input device.

5.3.2.6 Addresses of the IOCD and TCW

The address of the IOCD for a class E channel is determined by subtracting 4 from the interrupt level number of that channel, and then multiplying the result by 8 to form an index into a doubleword per entry table. The resulting address points to the first word of the IOCD for the I/O operation. The second word of the IOCD contains the address of the TCW for the operation. The TCW address is stored in the second word of the IOCD by software at some point in time before the I/O operation is requested. Table 5-2 provides the class E I/O default address for the IOCD of each channel interrupt. The base address of the E-Class IOCD provided by the scratchpad is located at address $F2_{16}$. The scratchpad default IOCD table is located at address 700_{16}.

5.3.2.7 Test Device Instruction

The test device (TD) instruction does not initiate any action in the I/O operation, but may be used to obtain status information from the peripheral device(s). It can be programmed in one of three descriptive levels of test: the TD8000, TD4000, or TD2000 level. The status information is recorded in the condition code. Figure 5-7 illustrates the format for the TD instruction and lists the condition code responses for each of the three levels of test.

The TD8000 level of test presents the basic status of the addressed device and the associated I/O processor. The TD4000 level of test reveals more specific status definition than the TD8000 level. In the TD2000 level of test, detailed status information is reflected in a 16-bit halfword and four condition code bits.

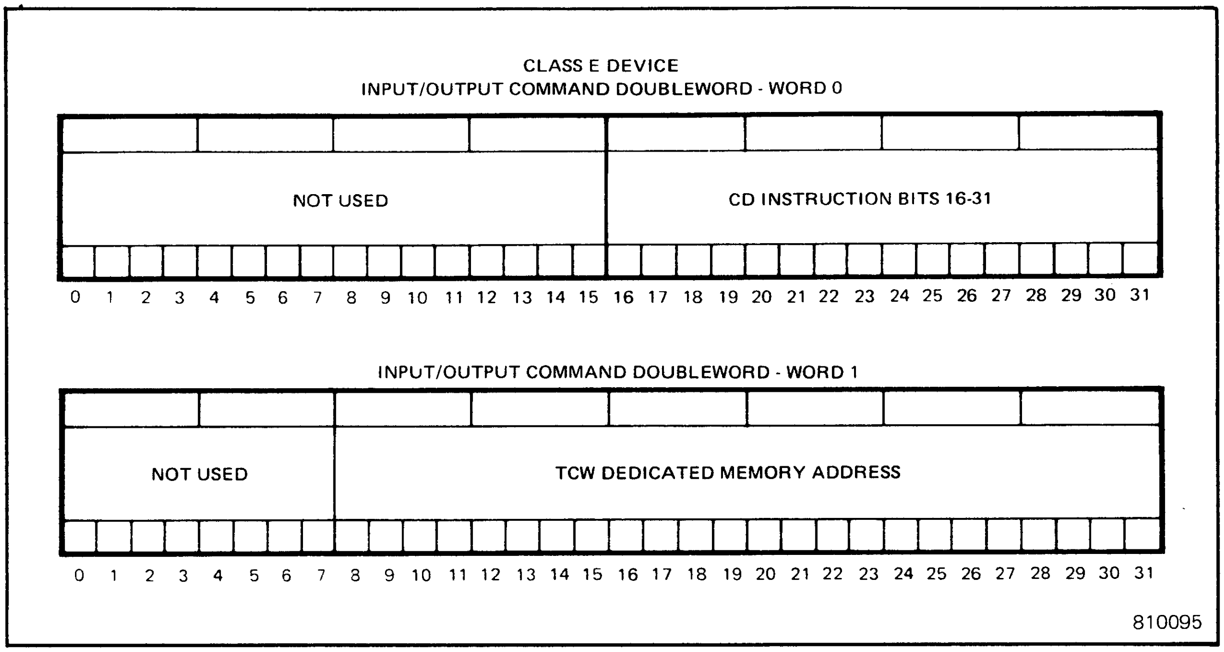
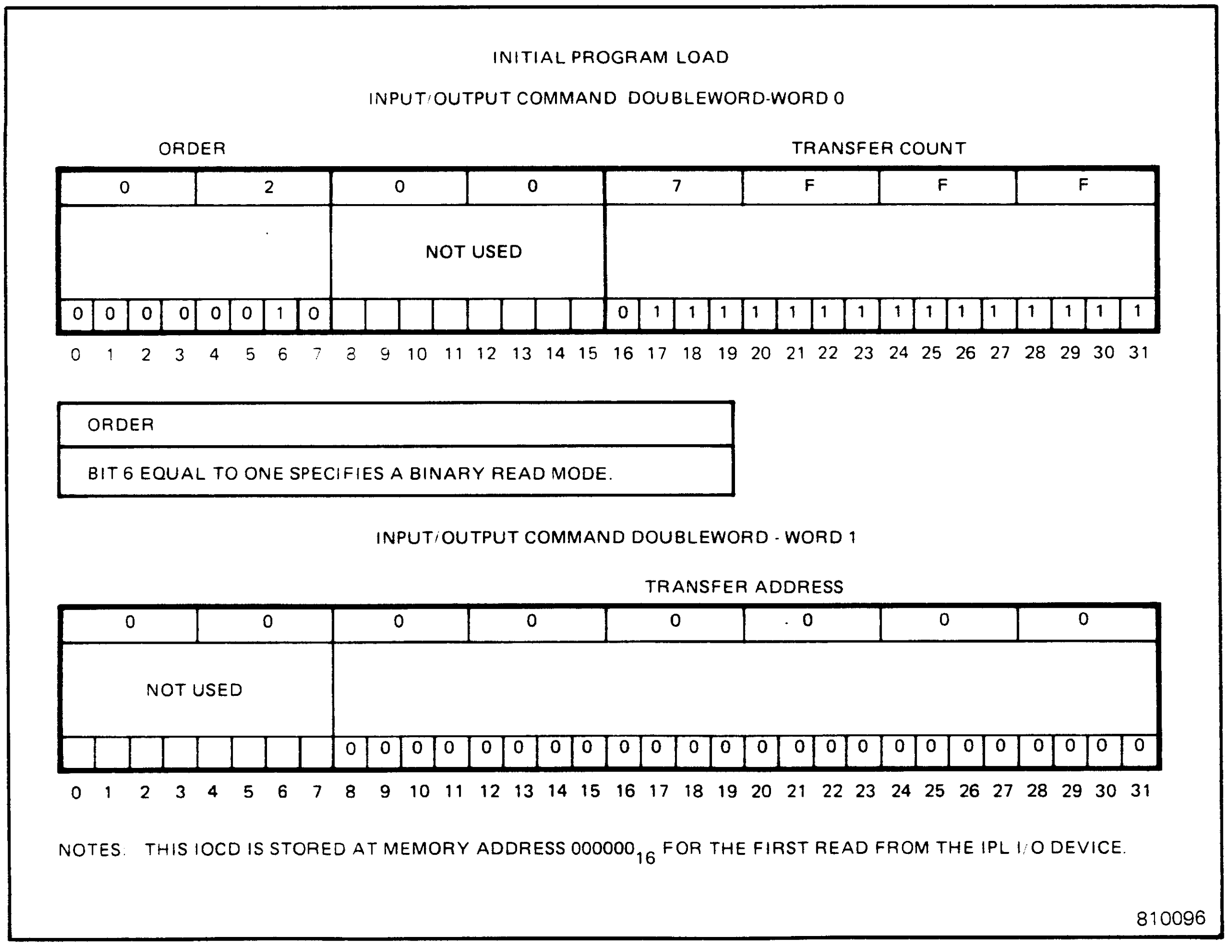


Figure 5-5. Class E Devices, IOCD Format



BIT 6 EQUAL TO ONE SPECIFIES A BINARY READ MODE.

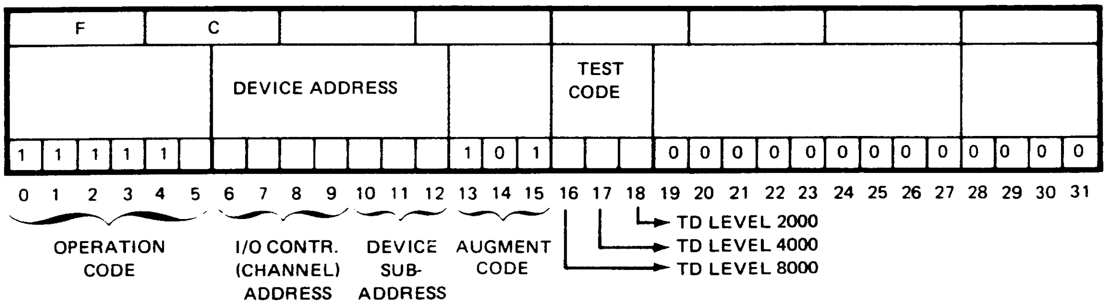
NOTES. THIS IOCD IS STORED AT MEMORY ADDRESS 000000_{16} FOR THE FIRST READ FROM THE IPL I/O DEVICE.

Figure 5-6. Initial Program Load, IOCD Format

Table 5-2
Class E I/O Default Addresses for IOCD and TCW

Default Interrupt Vector Location (IVL)	Interrupt Condition	*Default IOCD Address	*Default TCW Address
100	External/software interrupt 0		
104	External/software interrupt 1		
108	External/software interrupt 2		
10C	External/software interrupt 3		
110	I/O channel 0, interrupt 4	700	704
114	I/O channel 1, interrupt 5	708	70C
118	I/O channel 2, interrupt 6	710	714
11C	I/O channel 3, interrupt 7	718	71C
120	I/O channel 4, interrupt 8	720	724
124	I/O channel 5, interrupt 9	728	72C
128	I/O channel 6, interrupt A	730	734
12C	I/O channel 7, interrupt B	738	73C
130	I/O channel 8, interrupt C	740	744
134	I/O channel 9, interrupt D	748	74C
138	I/O channel A, interrupt E	750	754
13C	I/O channel B, interrupt F	758	75C
140	I/O channel C, interrupt 10	760	764
144	I/O channel D, interrupt 11	768	76C
148	I/O channel E, interrupt 12	770	774
14C	I/O channel F, interrupt 13	778	77C
150	External/software interrupt 14		
154	External/software interrupt 15		
158	External/software interrupt 16		
15C	External/software interrupt 17		
160	Real-time clock interrupt 18		
164	External/software interrupts 19		
↓	↓ ↓		
2B8	External/software interrupts 6E		
2BC	Interval timer interrupt 6F		

*Applicable to non-class F only.



TD LEVEL		CONDITION CODE RESPONSE		
	CC1	CC2	CC3	CC4
TD 8000	UNDEFINED	I/O CHANNEL ACTIVE (BUSY)	CLASS E I/O PROCESSOR ERROR	DEVICE STATUS PRESENT
TD 4000	INVALID MEMORY ACCESS	MEMORY PARITY ERROR	PROGRAM VIOLATION	DATA UNDERFLOW OR OVERFLOW
TD 2000	CAUSES A TRANSFER OF 16 BITS OF DEVICE STATUS INFORMATION TO THE MEMORY LOCATION SPECIFIED IN THE TCW DEDICATED LOCATION. THE MEANING OF EACH BIT IN THE 16-BIT STATUS HALFWORD DIFFERS ACCORDING TO DEVICE TYPE.			

NOTE: 1.	CC2 = 0 STATUS TRANSFER WAS PERFORMED
	CC2 = 1 STATUS TRANSFER WAS NOT PERFORMED
	CC4 = 1 CONTROLLER IS ABSENT OR POWERED OFF			

2.	IF ALL CONDITION CODES AT ANY TD LEVEL ARE TRUE (i.e., CC1-4=F), THE CONTROLLER IS NOT PRESENT OR IS TOTALLY INOPERABLE.			

Figure 5-7. Test Device Instruction Format

The status halfword is stored into the memory location specified by the contents of the transfer control word that corresponds to the I/O device addressed by the TD instruction. The status halfword is placed in either the right or left halfword position, depending on bits 30 and 31 of the TCW address. A TCW used with a TD2000 level instruction should always specify the halfword memory addressing.

5.3.3 Operation of Class F or 7 I/O Processors

The CPU supports class F I/O processors; that is, the Disc Processor II, the High Speed Tape Processor and the IOP. One class F I/O processor consists of the IOP (the channel), the multipurpose bus (MPB), and MPB controllers (see Figure 5-1).

The V6 IPU supports class 7 I/O which services the V6 IPU Console IOP channel devices. Each of these channels and devices obey class F (extended I/O) protocol rules. Class 7 is the subset of class F. Class 7 is defined only in the V6 IPU, and must be used in scratchpad channel (device) and interrupt entries to specify the V6 IPU Console IOP channel and devices (console, floppy disc, etc.).

The IOP serves as the interface between the MPB controllers and the CPU/memory by way of the SelBUS. The IOP receives requests over the SelBUS from the CPU as a result of I/O instructions. It executes the IOP channel-type programs and initiates CPU interrupt/status transfers to indicate I/O completion or exceptional conditions. The IOP also schedules requests for main memory from the controllers.

A MPB controller receives control information from the IOP, controls timing, provides data buffers, validates the command, and converts the I/O command to a form acceptable to the I/O device.

There are three types of MPB controllers:

1. A single device controller is dedicated to a single device only.
2. A multiplexing controller services several devices while maintaining completely concurrent operation of all its devices.
3. A multidevice controller (MDC) also services several devices but can service only one at a time. Such a controller will appear busy when service is simultaneously requested of a second device.

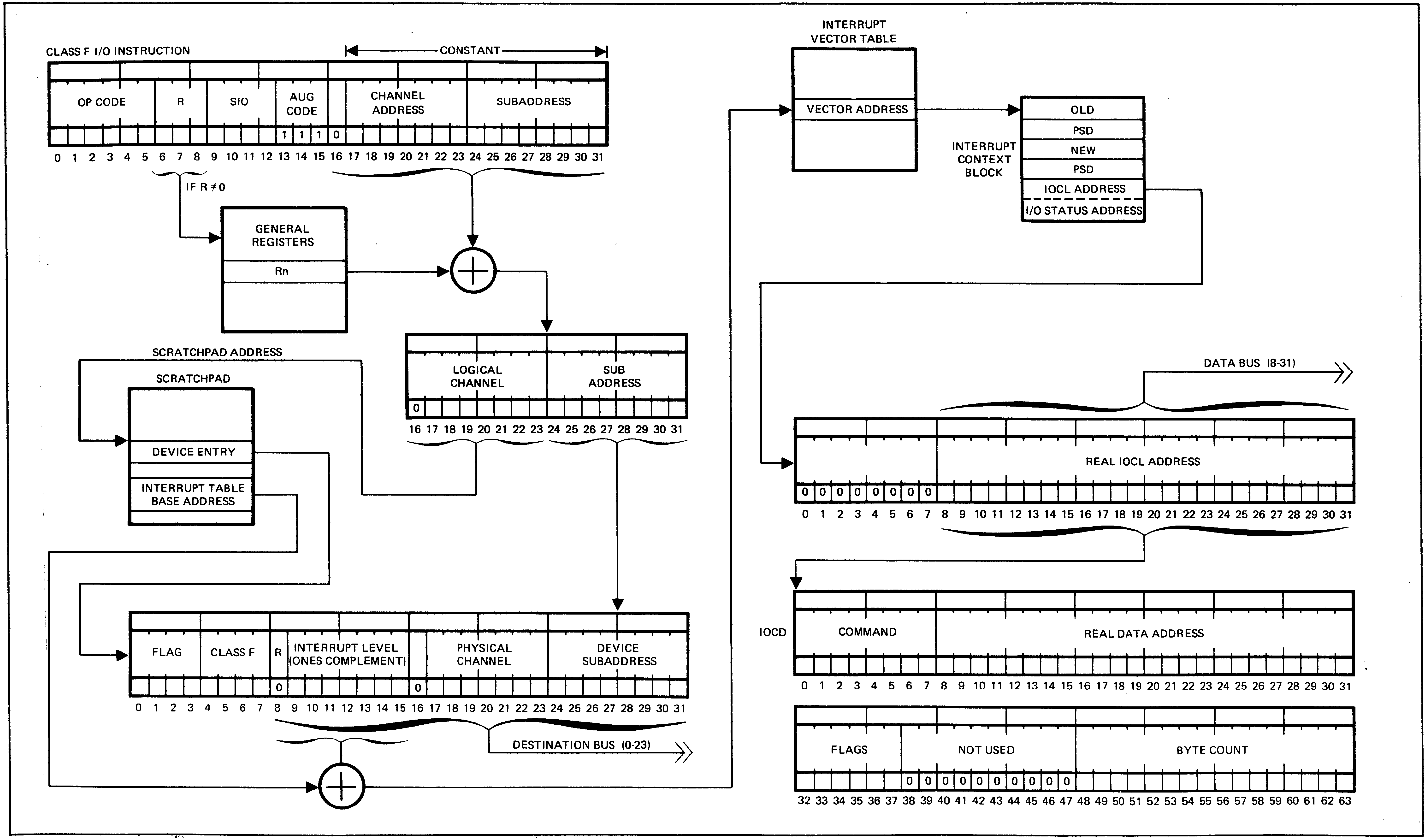
The IOP (channel) can support up to 16 I/O controllers. Each I/O controller may in turn support as many as 16 device addresses, but there is a maximum of 128 separately addressed devices that may be connected to the IOP at any one time.

Each I/O processor is assigned main memory locations to transmit or receive control information required to initiate or terminate an I/O operation. The control information consists of:

1. Service interrupt vector address.
2. Input/output command list address.
3. Status address.
4. New program status doubleword (new PSD).
5. Old program status doubleword (old PSD).

A graphic representation of the I/O control words is shown in Figure 5-8.

This page intentionally left blank



I/O Control Words (Class F)

Figure 5-8

Writable control storage is an option that provides a source of write/read memory for the channel. The writable control storage allows the I/O processor to be customized for special uses. The writable control storage is loaded by special instruction and may contain any program the user requires.

5.3.3.1 Interrupt Level

Each class F I/O processor has a unique interrupt level number assigned to it. The interrupt level may be any one of the levels supported by the CPU implementation.

5.3.3.2 Subaddresses

Each IOP can support a total of 16 MPB controllers. Each MPB controller may in turn support as many as 16 device addresses, but a total of 128 subaddresses maximum. Four of these 128 subaddresses (FC through FF) are used for operator console functions. However, the number of separately addressed devices that can be connected to the IOP at any one time is determined by the generic capability of the MPB controllers.

5.3.3.3 Input/Output Instructions

Class F I/O operations provide for extended addressing capabilities and are used with all standard I/O devices. Class F I/O includes the implementation of a set of special instructions that provide extended software control. As with all I/O instructions, class F instructions can only be executed when the CPU is in the privileged mode. The following is a list of the input/output instructions:

1. Start I/O (SIO)
2. Test I/O (TIO)
3. Halt I/O (HIO)
4. Enable write channel WCS (ECWCS)
5. Write channel WCS (WCWCS)
6. Enable channel interrupt (ECI)
7. Disable channel interrupt (DCI)
8. Activate channel interrupt (ACI)
9. Deactivate channel interrupt (DACI)
10. Reset channel (RSCHNL)
11. Stop I/O (STPIO)
12. Reset controller (RSCTL)
13. Grab controller (GRIO)

As shown in Figure 5-8, for all class F I/O instructions, bits 16 through 31 contain the channel and subaddress fields and bits 6 through 8 designate the R field. If the R field is nonzero, then bits 6 through 8 specify the general register whose contents will be added to the channel and subaddress field to form the logical channel and device subaddress. If R is specified as zero, then only the channel and subaddress fields will be used. Also, the IOP will ignore the subaddress for operations that pertain only to the channel.

The start I/O (SIO) instruction initiates an I/O operation or is used to return condition codes if an I/O execution could not be executed and may clear pending interrupt (SIO rejected).

The test I/O (TIO) instruction interrogates the current state of the channel, subchannel, controller, and device, and may be used to clear pending interrupt conditions.

The halt I/O (HIO) instruction terminates a channel, controller, and/or device operation immediately and may clear pending interrupt (HIO Rejected).

The enable write channel WCS (ECWCS) instruction sets an interlock in the CPU and conditions the channel for loading WCS. The ECWCS must be executed prior to actually writing the control store with a write channel WCS command.

The write channel WCS (WCWCS) instruction is the second part of a two-instruction sequence, the first being the ECWCS, for loading the specified channel WCS. There must be no intervening I/O instructions to the class F I/O controller to be loaded.

The enable channel interrupt (ECI) instruction allows the channel to request interrupts from the CPU.

The disable channel interrupt (DCI) instruction prohibits the channel from requesting an interrupt. Pending status conditions can only be cleared by the execution of a start I/O, test I/O, or halt I/O if the channel is disabled. The DCI instruction does not clear pending requests.

The activate channel interrupt (ACI) instruction causes the channel to actively contend for interrupt priority except that the channel never requests an interrupt. This instruction affects pending status conditions because interrupt requests from this level and all lower levels are inhibited, and no status will be posted except by issuing a Start I/O (SIO), Test I/O (TIO), or Halt I/O (HIO).

The deactivate channel interrupt (DACI) instruction causes the channel to suspend contention for interrupt priority. If an interrupt request is queued, the channel may now request an interrupt. The instruction following a DACI is uninterruptible.

The reset channel (RSCHNL) instruction resets all interrupt and I/O activity in the channel. All requesting and pending conditions will be cleared. The channel work buffer in main memory will be deallocated. The channel may remain busy for extended periods of time following RSCHNL.

The stop I/O (STPIO) instruction terminates the operation in the controller after the completion of the current IOCD. The termination is orderly. The channel will suppress command and data chaining.

The reset controller (RSCTL) instruction resets a specific controller regardless of its previous condition. The subchannel and all pending and generated status conditions are cleared. The reset is immediate.

The grab controller (GRIO) instruction takes away control of a controller which is reserved to another channel. The grabbing channel is assigned as the reserving channel.

5.3.3.4 Input/Output Initiation

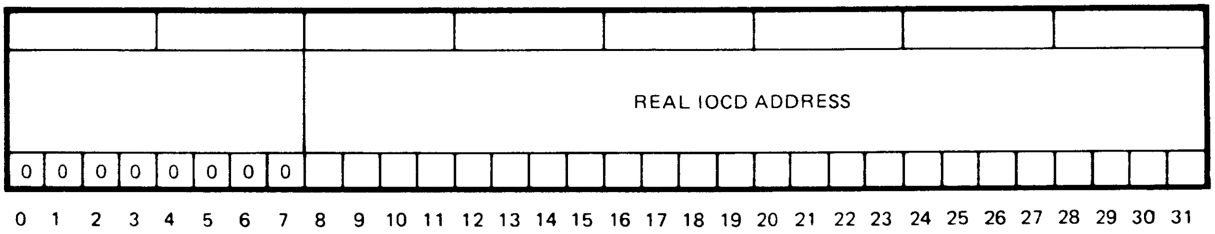
I/O operations are initiated by the start I/O instruction. If the specified channel/subchannel is present and not busy, the SIO is accepted, and the CPU continues to the next sequential instruction. The channel/controller independently governs the I/O device specified by the instruction.

Prior to the execution of the I/O instruction, the software stores the address of the first input/output command doubleword (IOCD) to be executed into the fifth word of the interrupt context block associated with the channel.

5.3.3.5 Input/Output Command List Address (IOCLA)

The class F IOCD is a 64-bit value that describes a step in an I/O operation. One or more IOCDs in a series comprise an input/output command list (IOCL). The input/output command list address (IOCLA) indicates a word address of the first IOCD of a series to be executed.

Successful execution of a SIO or a WCWCS instruction causes the CPU to transmit the IOCLA to the channel/controller. The IOCLA is stationed in main memory at a location specified by the service interrupt vector, plus 16 (decimal). Each of the I/O channels has a corresponding service interrupt vector. Below is the format for the IOCLA indicated by the contents of the service interrupt vector plus 16.



830557

5.3.3.6 Input/Output Memory Addressing

The memory addressing method used for class F I/O is real addressing. Real addressing is the capability to directly address any memory location within the 16MB (4MW) maximum capacity of the main memory without any address translation. This addressing method differs from the addressing method normally used by the software, which relies on hardware address conversion to transform the logical address to a real (physical) address.

Memory addresses are transferred to the channel when a start I/O or write channel WCS instruction is executed by the CPU. Prior to the execution of the I/O instruction, the software stores the address of the first input/output command doubleword (IOCD) to be executed in the fifth word of the interrupt context block associated with the channel. The word indicated is referred to as the input/output command list address (IOCLA).

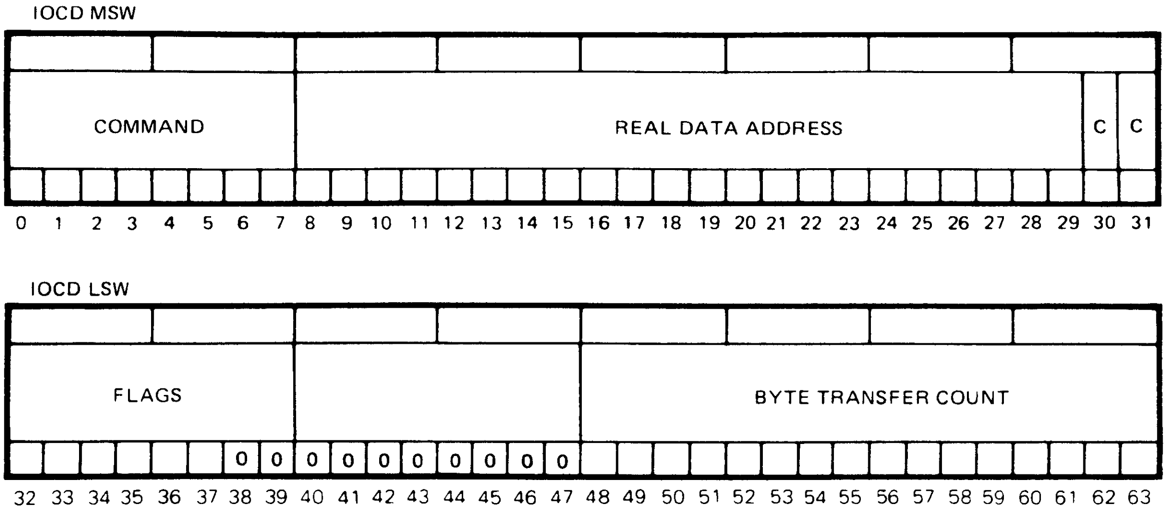
5.3.3.7 Input/Output Command Doubleword Format

The real IOCL address is passed to the channel/controller on the data bus.

The start I/O instruction is the only instruction that is able to cause the channel/controller to fetch an IOCD. One or more IOCDs create an input/output command list (IOCL).

The address indicated in the IOCLA specifies the word address of the first IOCD to be executed by the channel.

The IOCD format is shown in Figure 5-9 and described in subsequent paragraphs.



BIT ASSIGNMENTS IN THE COMMAND ARE:

X	X	X	X	0	0	0	0	CHANNEL CONTROL
M	M	M	M	0	1	0	0	SENSE
X	X	X	X	1	0	0	0	TRANSFER IN CHANNEL
M	M	M	M	1	1	0	0	BACKWARD READ
M	M	M	M	M	M	0	1	WRITE
M	M	M	M	M	M	1	0	READ
M	M	M	M	M	M	1	1	CONTROL

M = MODIFIER BITS; THESE BITS ARE USED FOR THE BASIC COMMAND.

C-BIT ASSIGNMENTS ARE:

BIT 30	BIT 31	
0	0	BYTE 0
0	1	BYTE 1
1	0	BYTE 2
1	1	BYTE 3

FLAG BIT ASSIGNMENTS ARE:

BIT 32	DATA CHAIN (HOLDS OFF TERMINATION WHEN TRANSFER COUNT = 0)
BIT 33	COMMAND CHAIN
BIT 34	SUPPRESS INCORRECT LENGTH
BIT 35	SKIP
BIT 36	PROGRAM CONTROLLED INTERRUPT
BIT 37	REAL-TIME OPTION
BITS 38 AND 39	RESERVED (MUST BE ZERO)

Figure 5-9. Class F Devices IOCD Format

The IOCD command field specifies one of the following seven commands:

Write
Read
Read backward
Control
Sense
Transfer in channel
Channel control

If more than one IOCD is specified, the IOCDs are fetched sequentially except when transfer in channel (TIC) is specified. Search (compare) commands can cause the skipping of the next sequential IOCD if the condition becomes true (i.e., search equal, search low, or search high). The channel or controller will then increment by 16 rather than by 8.

The real data address (IOCD MSW bits 8 through 29) specifies the starting address of the data area. The data address will be a byte address, and the channel will internally align the information transferred to or from main memory.

The byte transfer count (IOCD LSW bits 16 through 31) specifies the number of bytes to be transferred by the channel to or from main memory. The actual number of memory transfers performed by the channel is dependent upon the channel implementation.

5.3.3.8 Input/Output Commands

The write command causes a write (output) operation to the selected I/O device from the specified main memory address.

The read command causes a read (input) operation from the selected I/O device to the specified main memory address.

The read backward command causes a read (input) operation in the reverse direction from the selected I/O tape device to the specified main memory address in descending order.

The control command causes control information to be passed to the selected I/O device. The control command may provide a data address and byte count for additional control information that may be stored in main memory. Control information is device dependent. For example, it may instruct a magnetic tape to rewind, a printer to space a certain number of lines, or a disc to perform a seek operation.

The sense command causes the storing of controller/device information to the specified location of main memory. One or more bytes of information are transferred, depending upon the device, up to the byte count specified in the operation. The sense information provides additional device-dependent information not provided in the status flags.

The transfer in channel (TIC) command specifies the address of the next IOCD to be executed. The TIC command allows the programmer to change the sequence of IOCD execution. The IOCLA cannot specify a TIC command as the first IOCD in a command list, nor can one TIC specify another TIC command.

The channel control command causes the transfer of information from a specific location in main memory. One or more bytes of information is transmitted to the channel. The channel control command provides for the passing of information required to initialize all channels.

5.3.3.9 Input/Output Termination

An I/O operation terminates when the channel, controller, and/or device indicates the end of an operation. All I/O operations accepted by the channel will always terminate with at least one termination status being presented to software.

Channel end is a termination condition that indicates that all information associated with the I/O operation has been received or provided, and the channel or the integrated channel/controller is no longer needed.

Controller end is a termination condition that is reported after a channel end was reported in which the controller was busy. It indicates that the controller is no longer active and is available to initiate another command.

Device end is an indication from the controller to the channel that an I/O device has terminated execution of its operation.

The combination of channel end, controller end, and device end indicates that the I/O operation is complete. The termination indications are hierarchical; a device end cannot happen before a controller end, which cannot happen before a channel end.

An I/O operation can also fail to be accepted by the channel during I/O initiation. Conditions that prevent I/O initiation are:

1. Channel or subchannel busy
2. Channel not operational or nonexistent
3. Pending termination (SIO, TIO, and HIO only) status from a previously initiated I/O operation.

I/O initiation failures are reported to software by the setting of condition codes on the start I/O instruction and, where applicable, the storing of status.

5.3.3.10 Input/Output Status Words

The status word is maintained and stored by the channel. When the CPU acknowledges an interrupt, the channel stores the status words in the CPU memory and transmits the address where they are stored to the CPU. This address of the status words is then stored in the sixth word of the extended I/O interrupt context block (EXT I/O ICB). The status words contain information relating to the execution of the last IOCD or from any asynchronous condition requiring software notification (i.e., tape loaded, disc pack mounted, etc.). Figure 5-10 provides the formats and definitions of contents in the I/O status words.

5.4 Input/Output Interrupts

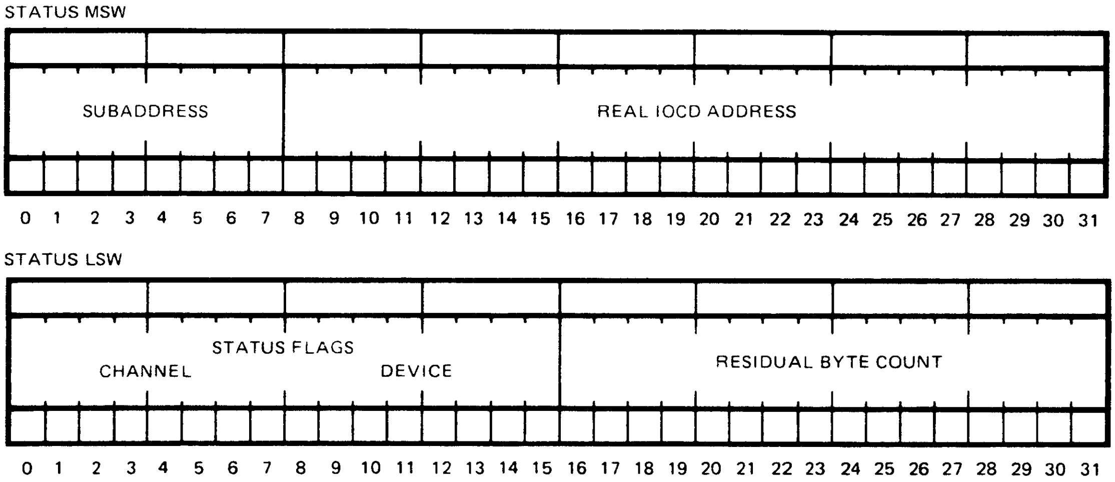
Input/output interrupts can be caused either by termination of an I/O operation, by operator intervention at the I/O device, or when a program-controlled interrupt is requested by an IOCD. An I/O interrupt causes the current PSD to be stored in the ICB (Figure 5-9) associated with the interrupt level. The new PSD is loaded from the ICB. For F class I/O operations, the status address is updated in the ICB by the CPU during the interrupt processing.

An interrupt can be caused by the device, controller, or channel. If a channel or controller has multiple I/O interrupt requests pending, it establishes a priority sequence for them before initiating an I/O interrupt request to the CPU. This priority sequence is maintained when the channel stores the status and reports the status address to the CPU.

The mode in which the channel operates during the software interrupt processing is determined by the mode setting of the channel and the implementation of the channel. The software may use bits 48 and 49 of the new PSD to select one of two options: unblocked or blocked operation.

Unblocked operation specifies that the CPU, upon receipt of an interrupt, causes the channel to go active and block all interrupts of a lower priority. The channel services the interrupt, and the software in turn issues a DACI command to restore the interrupt processing.

Blocking specifies that the CPU, upon receipt of an interrupt, causes the channel to deactivate. The CPU blocks all incoming interrupts and services the pending interrupt. The software, in turn, issues a UEI command or a LPSD, or LPSDCM to the CPU, thereby restoring interrupt processing. The target PSD of the LPSD or LPSDCM instruction should specify an unblocked operation in bits 48 and 49.



MOST-SIGNIFICANT WORD

BITS 0-7 SUBADDRESS SPECIFIES THE COMMUNICATIONS SUBCHANNEL PRESENTING THE STATUS.

BITS 8-31 REAL IOCD ADDRESS SPECIFIES THE ADDRESS OF THE LAST IOCD EXECUTED, PLUS 8

LEAST-SIGNIFICANT WORD

STATUS FLAGS

BIT 0 ECHO
1 POST-PROGRAMMED CONTROLLED INTERRUPT
2 INCORRECT LENGTH
3 CHANNEL PROGRAM CHECK
4 CHANNEL DATA CHECK
5 CHANNEL CONTROL CHECK
6 INTERFACE CHECK
7 CHAINING CHECK
8 DEVICE BUSY
9 STATUS MODIFIER
10 CONTROLLER END (N.U.)
11 ATTENTION (N.U.)
12 CHANNEL END
13 DEVICE END
14 UNIT CHECK
15 UNIT EXCEPTION
BITS 16-31 RESIDUAL BYTE COUNT FIELD SPECIFIES THE RESIDUAL BYTE COUNT OF THE LAST IOCD USED.

Figure 5-10. Input/Output Status Words Format

CHAPTER 6

INSTRUCTION REPERTOIRE

6.1 Introduction

This chapter describes each instruction that can be executed by the CPU. Instructions are grouped according to their purpose and function. Preceding each group is a text portion that explains the formats used in each group and other noteworthy features of that group of instructions.

The appendixes in this manual provide easy access to find specific instructions. The key listing for each appendix is along the right-hand margin for reference as an index. Appendix A contains all instructions functionally grouped in order as presented in the instruction set. Appendix B lists the instruction by mnemonic in alphabetical order. The single case execution time for each instruction is also included. Appendix C lists all the instructions by Op Code in hexadecimal order. Appendix D provides three special lists of instructions, which are:

. Instructions used in the 32 SERIES and the CONCEPT 32, but not recognized by this CPU.

. Instructions that are undefined when the computer operates in the base register mode.

. Instructions used in the base register mode only.

List of functional groupings:

Load/Store Instructions
Register Transfer Instructions
Memory Management Instructions
Branch Instructions
Compare Instructions
Logical Instructions
Shift Operation Instructions
Bit Manipulation Instructions
Fixed-Point Arithmetic Instructions
Floating-Point Arithmetic Instructions
Floating-Point Conversion Instructions
Control Instructions
Interrupt Instructions
Input/Output Instructions
Class F I/O Instructions
Class F I/O Writable Control Storage (WCS) Instructions
Alterable Control Storage/Writable Control Storage Instructions (V6 only)

The detailed information contained in each instruction is discussed in the following paragraphs.

NOTE

The V6 CPU and V9 CPU are designed to run in the Base Register mode of operation ONLY. Non-Base register mode of operation can run, but this mode is not fully supported. Indirect addressing in mapped environment may not function.

6.1.1 Mnemonic

The mnemonic for each instruction is a two-to six-letter symbol, in uppercase letters, that represents the instruction name and is accepted by the assembler.

The CPU instruction mnemonics follow a simple format. The basic types are as follows:

L	Load	or	LM	Load masked
ST	Store	or	STM	Store masked
AD	Add			
ADM	Add memory to register			
ARM	Add register to memory			
SU	Subtract			
SUM	Subtract memory from register			
MP	Multiply			
DV	Divide			
ADF	Floating-point arithmetic			
SUF				
MPF				
DVF				
B	Branch			
AN	AND			
OR	Logical OR			
EO	Exclusive OR			
C	Compare			

These basic mnemonics are augmented to define the operand data type. (A special set of instructions is provided for bit manipulation). The five basic data types are as follows:

B	Byte	(8 bits)
H	Halfword	(16 bits)
W	Word	(32 bits)
D	Doubleword	(64 bits)
I	Immediate	(16 bits)

Therefore, the resulting instruction mnemonics have forms such as:

LB	Load byte
LMH	Load masked halfword
STMW	Store masked word
ADI	Add immediate to register
SUMD	Subtract memory doubleword

6.1.2 Formats

A 16-bit or 32-bit machine-language representation shows the acceptable format(s) for each instruction. Several of the memory reference instructions display two formats, namely to distinguish between the nonbase register and the base register modes. Both formats for the memory reference instructions are shown in Figure 6-1. Figures 6-2 and 6-3 contain the other standard formats for Immediate and Interregister instructions, respectively.

6.1.3 Definition

The operation that is performed when the instruction is executed is briefly described. Registers or memory locations which are modified are defined. Table 6-1 includes the abbreviations used in the definitions.

6.1.4 Notes

Special considerations are given in notes following the basic operational description.

6.1.5 Summary Expressions

The summary expression is a symbolic expression, based on the definition, that shows the operation performed by the instruction. Table 6-1 includes the symbols and abbreviations used in summary expressions.

The following are examples of summary expressions used in the instructions:

$(R_S) \rightarrow (R_D)$

means that the contents of general purpose register (GPR) S replace the contents of general purpose register (GPR) D.

Zeros_{0-23}, byte operand → R

means that the byte operand is appended with zeros in positions 0 through 23 and the resulting word replaces the contents of the GPR specified by R.

(R), (R+1)

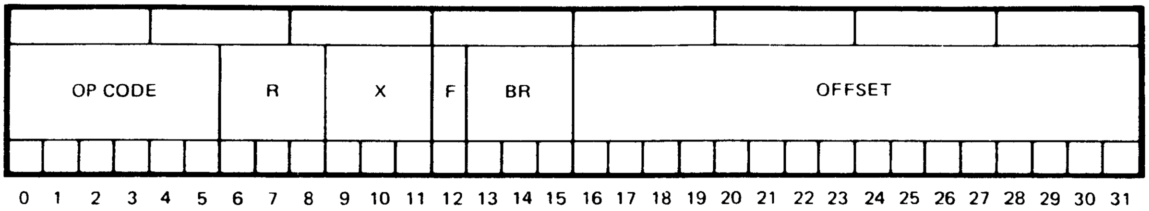
is an even/odd pair of registers.

(EWL), (EWL+1)

is an effective doubleword memory location.

6.1.6 Operation Code

The operation code (or op code) for each instruction is given in a four-digit, left-justified hexadecimal format. This format represents the 16 most-significant bits of the instruction word, which includes the implicit bits which identify the general purpose register, indirect addressing, indexing, and byte addressing. These additional bits, when set, are reflected in the true operation code.



BASE REGISTER FORMAT

BITS 0-5 DEFINE THE OPERATION CODE.

BITS 6-8 DESIGNATE A GENERAL PURPOSE REGISTER ADDRESS (0-7).

BITS 9-11 INDEX REGISTER FIELD ALLOWS GPR 1-7 TO BE USED AS INDEX REGISTERS IN THE ADDRESSING CALCULATION.

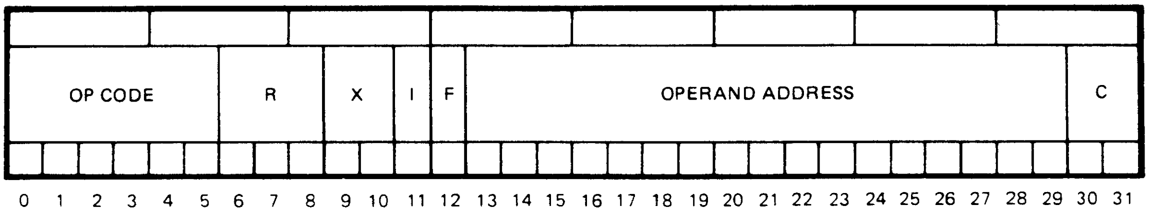
BIT 12 THE F BIT IS EFFECTIVELY PART OF THE OPERATION CODE.

BITS 13-15 THE BASE REGISTER FIELD IDENTIFIES ONE OF SEVEN REGISTERS (1-7) THAT CONTAINS A REFERENCE ADDRESS WITHIN THE PROGRAM ADDRESSING SPACE. IF THIS FIELD CONTAINS ALL ZEROS A 24-BIT VALUE OF ZERO IS USED AS THE BASE ADDRESS IN THE EFFECTIVE ADDRESS CALCULATION.

BITS 16-31 OFFSET FIELD PROVIDES THE POSITIVE DISPLACEMENT VALUE THAT IS ADDED TO THE RESULT OF THE BASE REGISTER FIELD TO FORM THE ADDRESS OF THE OPERAND. THE CONTENTS OF THE INDEX REGISTER IS ADDED IF X IS NON-ZERO.

BASE REGISTER EFFECTIVE ADDRESS CALCULATION.

EA = (0 if R_x = 0 ELSE contents of RX) + (O if BR = 0 ELSE contents of BR) + DISPLACEMENT



NONBASE REGISTER FORMAT

BITS 0-5 DEFINE THE OPERATION CODE.

BITS 6-8 DESIGNATE A GENERAL PURPOSE REGISTER ADDRESS (0-7).

BITS 9-10 DESIGNATE ONE OF THREE GENERAL PURPOSE REGISTERS TO BE USED AS AN INDEX REGISTER.
X = 00 DESIGNATES THAT NO INDEXING OPERATION IS TO BE PERFORMED.
X = 01 DESIGNATES THE USE OF R1 FOR INDEXING.
X = 10 DESIGNATES THE USE OF R2 FOR INDEXING.
X = 11 DESIGNATES THE USE OF R3 FOR INDEXING.

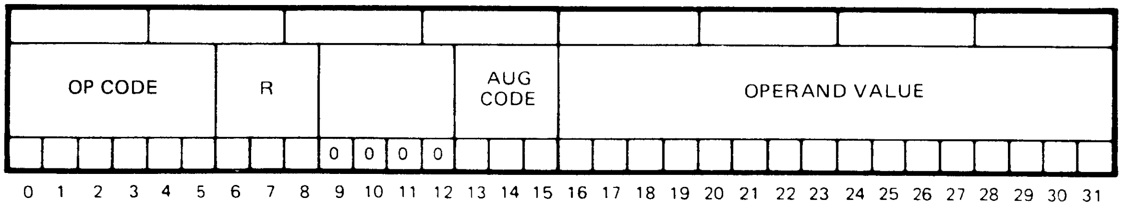
BIT 11 DESIGNATES WHETHER AN INDIRECT ADDRESSING OPERATION IS TO BE PERFORMED.
I = 0 DESIGNATES THAT NO INDIRECT ADDRESSING OPERATION IS TO BE PERFORMED.
I = 1 DESIGNATES THAT AN INDIRECT ADDRESSING OPERATION IS TO BE PERFORMED.

BITS 12-31 SPECIFY THE ADDRESS OF THE OPERAND WHEN THE X AND I FIELDS ARE EQUAL TO ZERO.

810344

Figure 6-1. Memory Reference Instruction Format

IN IMMEDIATE OPERAND INSTRUCTIONS, THE RIGHT HALFWORD OF THE INSTRUCTION CONTAINS THE 16-BIT OPERAND VALUE. THE FORMAT FOR THESE INSTRUCTIONS IS GIVEN BELOW.



BITS 0-5 DEFINE THE OPERATION CODE.

BITS 6-8 DESIGNATE A GENERAL PURPOSE REGISTER ADDRESS (0-7).

BITS 9-12 UNASSIGNED.

BITS 13-15 DEFINE AUGMENTING OPERATION CODE.

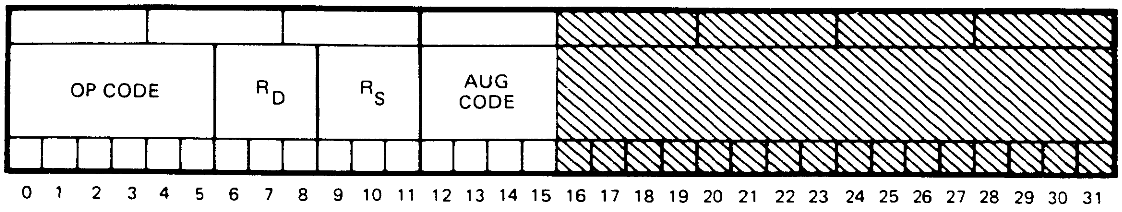
BITS 16-31 CONTAIN THE 16-BIT OPERAND VALUE.

ARITHMETIC OPERANDS ARE REPRESENTED IN TWOS COMPLEMENT FORM WITH SIGN IN BIT 16.

810346

Figure 6-2. Immediate Instruction Format

INTERREGISTER INSTRUCTIONS ARE HALFWORD INSTRUCTIONS AND AS SUCH MAY BE STORED IN EITHER THE LEFT OR RIGHT HALF OF A MEMORY WORD. THE FORMAT FOR INTERREGISTER INSTRUCTIONS IS GIVEN BELOW.



BITS 0-5 DEFINE THE OPERATION CODE.

BITS 6-8 DESIGNATE THE REGISTER TO CONTAIN THE RESULT OF THE OPERATION.

BITS 9-11 DESIGNATE THE REGISTER WHICH CONTAINS THE SOURCE OPERAND.

BITS 12-15 DEFINE THE AUGMENTING OPERATION CODE.

810345

Figure 6-3. Interregister Instruction Format

Table 6-1
Symbol Definitions (Sheet 1 of 3)

Symbol	Definition
BR	Base Register 0-7 (BR0-BR7)
CCn	Condition code bit n
CMCR	Cache Memory Control Register
CS	Control Switches
D	Specified branch condition
EA	Effective Address of an operand or instruction
EBA	Effective Byte Address
EBL	Effective Byte Location specified by EBA
EDA	Effective Doubleword Address
EDL	Effective Doubleword Location consisting of an even numbered word location and the next higher word location specified by the EDA
EHA	Effective Halfword Address
EHL	Effective Halfword Location specified by the EHA
EWA	Effective Word Address
EWL	Effective Word Location specified by the EWA
F	Format bit
FIX	Conversion of floating-point to fixed-point form
FLT	Conversion of fixed-point to floating-point form
GPR	General Purpose Register 0-7 (GPR0-GPR7)
I	Indirect address bit
IW	Instruction Word
MA	Real Memory Address
MIDL	Memory Image Descriptor List
PC	Program Count
PSD	Program Status Doubleword

Table 6-1
Symbol Definitions (Sheet 2 of 3)

Symbol	Definition
PSD1	The first word of the Program Status Doubleword
PSD2	The second word of the Program Status Doubleword
R	Register 0-7 (R0-R7)
R_B	Base register
R_D	Destination register
RD_B	Destination base register
R_S	Source register
RS_B	Source base register
Rm-n	Bits m through n of general purpose register R
Rn	Bit of general purpose register R
SBL	Specified Bit Location within a byte
SCC	Set condition code bits
SE	Sign Extended
X	Index register: X Value — GPR Used for Indexing 00 — None 01 — R1 10 — R2 11 — R3
-Y	Twos complement of Y
$\overline{Y}$	Ones complement of Y, or logical NOT function for Y
Zeros	Zero fill as specified
&	Logical AND
v	Logical OR
⊕	Exclusive OR
→	Replace data from left symbol to right symbol (e.g., (R) → (R1) means the contents of R replaces the contents of R1.
1	Indicates referenced bit locations which are set

Table 6-1
Symbol Definitions (Sheet 3 of 3)

Symbol	Definition
+1	The register or memory address is incremented by one (e.g., R, R+1 indicates a register even/odd pair consisting of (R) and (R+1) respectively
+n	The memory address or register incremented 'n' times
( )	Contents of
[ ]	The combined contents of
<	Less
≤	Less or equal
=	Equal
≥	Greater or equal
>	Greater
≠	Not Equal
+	Algebraic addition
-	Algebraic subtraction
x	(or no symbol) Algebraic multiplication
/	Algebraic division
:	Comparison symbol

6.1.7 Assembly Coding Conventions

The basic assembler coding format for memory reference instructions is:

XXXXXX $\overset{s}{d}$,*m,x

which translates to

XXXXXX	Instruction mnemonic
s d	Source or destination general purpose register
*	Indirect addressing (optional)
m	Memory operand
x	Indexed by register number x

Nonmemory reference instruction coding is similar to the memory reference format. Table 6-2 lists all codes used in defining the Assembler coding formats.

Table 6-2
Assembler Coding Symbols

Code	Description
Uppercase letters	Instruction mnemonic
b	Bit number (0-31) in a general purpose register
c	Bit number (0-7) within a byte
d	Destination general purpose register number (0-7)
f	Function
m	Operand memory address
n	Device address
s	Source general purpose register number (0-7)
v	Value for immediate operands, number of shifts, etc.
x	Index register number 1, 2, or 3 (optional)
*	Indirect addressing (optional)
,	Assembler syntax

6.2 Instruction Set

6.2.1 Load/Store Instructions

The load/store instruction group manipulates data between memory and the general purpose or base registers. In general, load instructions transfer operands from specified memory locations to registers; store instructions transfer data from registers to specified memory locations. Provisions have been made to mask or clear the contents of registers, memory bytes, halfwords, words, or doublewords during instruction execution.

6.2.1.1 Instruction Format

The load/store instructions use the standard memory reference, immediate, and interregister formats.

6.2.1.2 Condition Code

A condition code is set during the execution of most load instructions to indicate whether the operand being transferred is greater than, less than, or equal to zero. Arithmetic exceptions are also reflected by the condition code results. All store instructions leave the condition code unchanged.

6.2.1.3 Memory to Register Transfer

Figure 6-4 depicts the positioning of information for transfer from memory to any general purpose or base register in the computer.

This page intentionally left blank

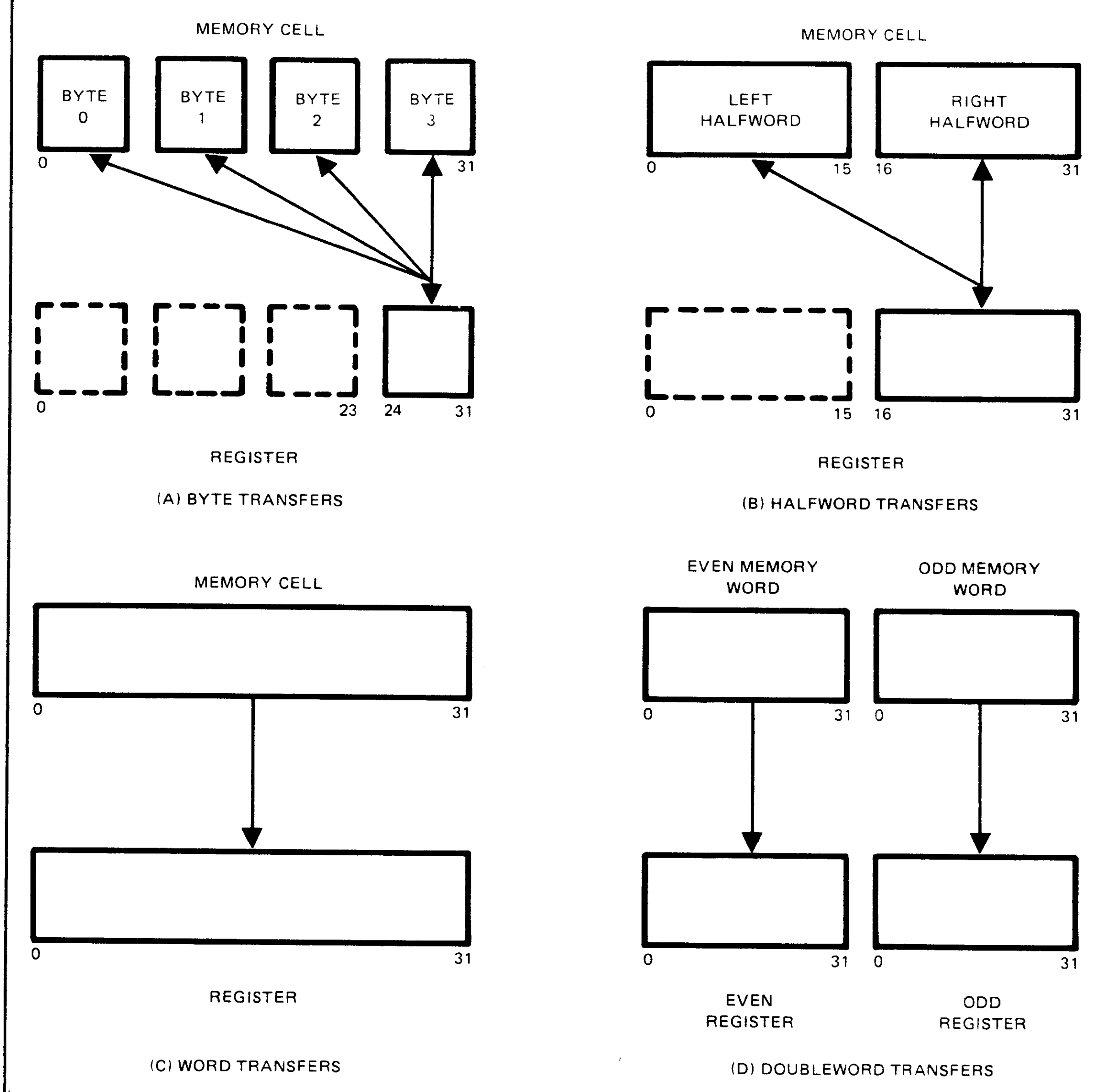
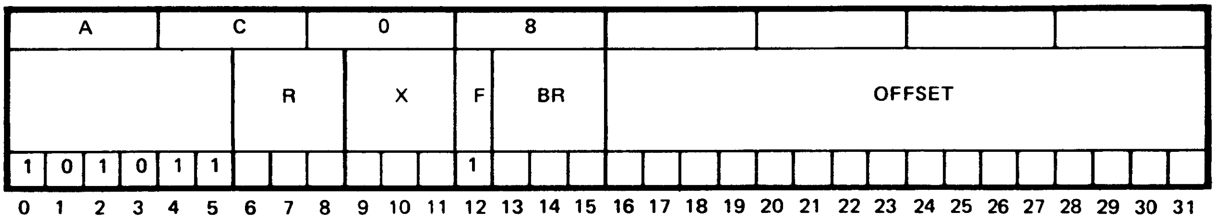


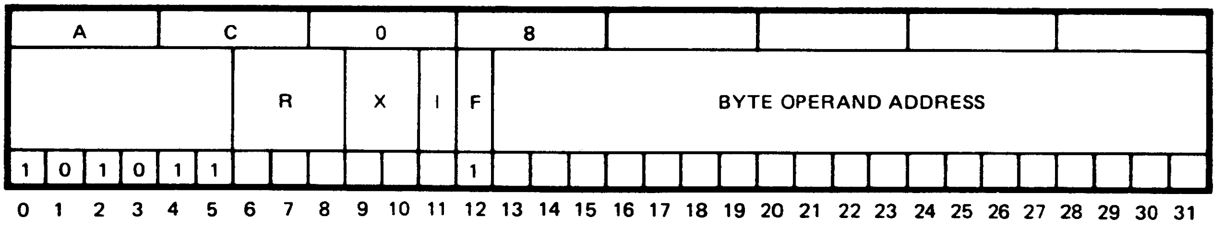
Figure 6-4. Positioning of Information Transferred Between Memory and Registers

LOAD BYTE AC08

LB d,*m,x



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

830131

DEFINITION

The effective byte location (EBL) specified by the effective byte address (EBA) is accessed and transferred to bit positions 24-31 of the general purpose register (GPR) specified by R. Bit positions 0-23 of the GPR specified by R are cleared to zeros.

SUMMARY EXPRESSION

(EBL) → R_{24-31}

Zeros → R_{0-23}

CONDITON CODE RESULTS

CC1: Always zero
CC2: Is set if (R_{0-31}) is greater than zero
CC3: Always zero
CC4: Is set if (R_{0-31}) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 are added to the instruction offset to obtain the logical address. The contents of memory byte 001101 are transferred to bits 24-31 of GPR1; bits 0-23 of GPR1 are cleared. CC2 is set to indicate that the contents of GPR1 are greater than zero.

Memory Location:	01000
Hexadecimal Instruction:	AC8E1100 (R=1, X=0, BR=6)
Assembly Language Coding:	LB 1,X'1100' (6)

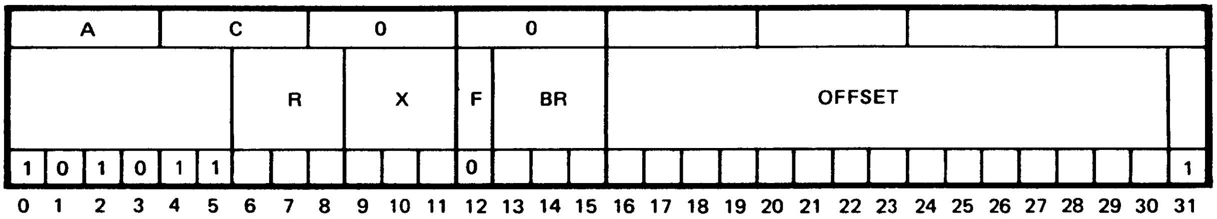
Before	PSD1	GPR1	BR6	Memory Byte 001101
	02001000	517CD092	00000001	00A60000
After	PSD1	GPR1	BR6	Memory Byte 001101
	22001004	000000A6	00000001	00A60000

NONBASE REGISTER MODE EXAMPLE

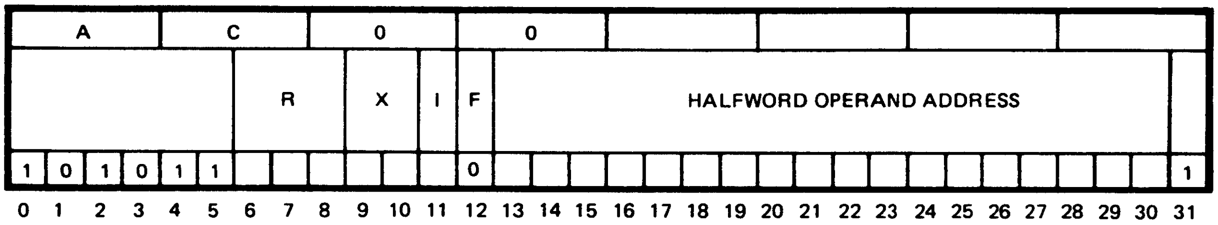
The contents of memory byte 01101 are transferred to bits 24-31 of GPR1; bits 0-23 of GPR1 are cleared. CC2 is set because the contents of GPR1 are greater than zero.

Memory Location:	01000
Hexadecimal Instruction:	AC 88 11 01 (R=1, X=0, I=0)
Assembly Language Coding:	LB 1,X'1101'

Before	PSD1	GPR1	Memory Byte 01101
	00001000	517CD092	00A60000
After	PSD1	GPR1	Memory Byte 01101
	20001004	000000A6	00A60000



BASE REGISTER FORMAT



830132

NONBASE REGISTER FORMAT

DEFINITION

The effective halfword location (EHL) specified by the effective halfword address (EHA) is accessed and the sign bit (bit 16) is extended left 16 bit positions to form a word. This word is transferred to the general purpose register (GPR) specified by R.

SUMMARY EXPRESSION

(EHL)SE → R

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R_{0-31}) is greater than zero
CC3: Is set if (R_{0-31}) is less than zero
CC4: Is set if (R_{0-31}) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of memory halfword 0005A2 are transferred to bits 16-31 of GPR4. Bits 0-15 of GPR4 are set by the sign extension. CC3 is set.

Memory Location:	00408
Hexadecimal Instruction:	AE060503 (R=4, X=0, BR=6)
Assembly Language Coding:	LH 4,X'502'(6)

Before	PSD1	GPR4	BR6	Memory Halfword 0005A2
	12000408	5C00D34A	000000A0	930C
After	PSD1	GPR4	BR6	Memory Halfword 0005A2
	1200040C	FFFF930C	000000A0	930C

NONBASE REGISTER MODE EXAMPLE

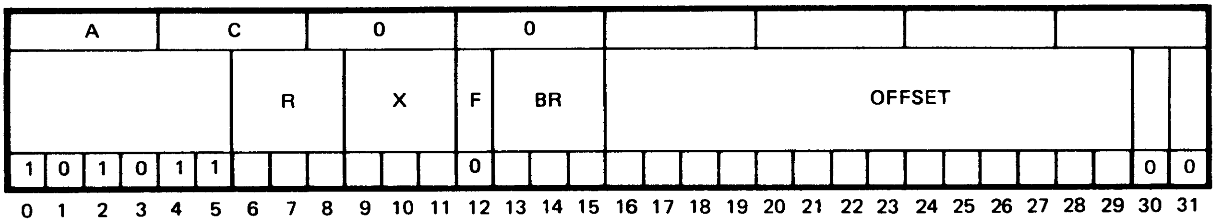
The contents of memory halfword 00502 are transferred to bits 16-31 of GPR4. Bits 0-15 of GPR4 are set by the sign extension. CC3 is set.

Memory Location:	00408
Hexadecimal Instruction:	AE 00 05 03 (R=4, X=0, I=0)
Assembly Language Coding:	LH 4,X'502'

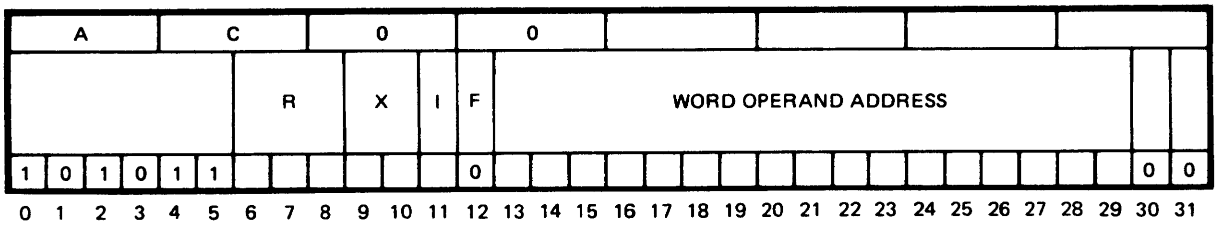
Before	PSD1	GPR4	Memory Halfword 00502
	10000408	5C00D34A	930C
After	PSD1	GPR4	Memory Halfword 00502
	1000040C	FFFF930C	930C

LOAD WORD
AC00

LW
d,*m,x



BASE REGISTER FORMAT



830133

NONBASE REGISTER FORMAT

DEFINITION

The effective word location (EWL) specified by the effective word address (EWA) is accessed and transferred to the general purpose register (GPR) specified by R.

SUMMARY EXPRESSION

(EWL) → R

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R_{0-31}) is greater than zero
CC3: Is set if (R_{0-31}) is less than zero
CC4: Is set if (R_{0-31}) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of GPR4, BR6, and the instruction offset are added to obtain the logical address. The contents of memory word 0027A4 are transferred to GPR7.

Memory Location:	002390
Hexadecimal Instruction:	AFC6 2700 (R=7, X=4, BR=6,)
Assembly Language Coding:	LW 7, X '2700' (6), 4

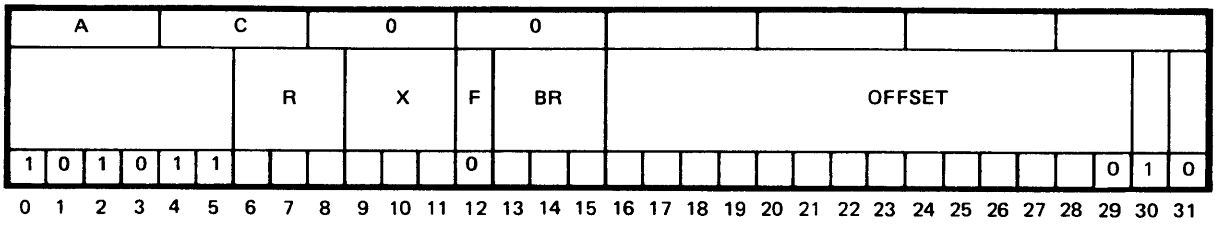
Before	PSD1	GPR7	GPR4	BR6	Memory word 0027A4
	02002390	0056879A	00000004	000000A0	4D61A28C
After	PSD1	GPR7	GPR4	BR6	Memory word 0027A4
	22002394	4D61A28C	00000004	000000A0	4D61A28C

NONBASE REGISTER MODE EXAMPLE

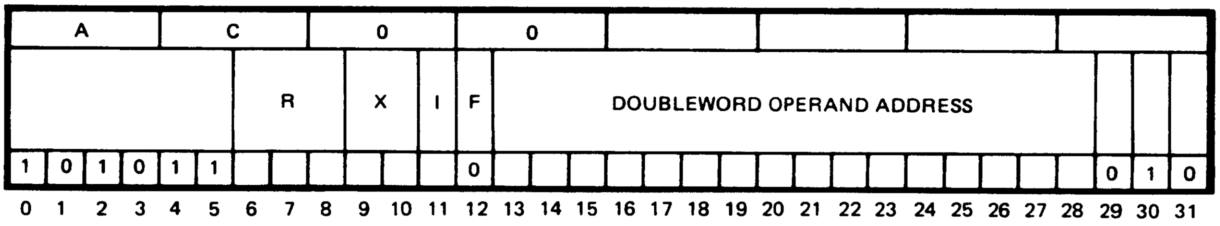
The contents of memory word 027A4 are transferred to GPR7.

Memory Location:	02380
Hexadecimal Instruction:	AF8027A4 (R=7, X=0, I=0)
Assembly Language Coding:	LW 7, X'27A4'

Before	PSD1	GPR7	Memory Word 027A4
	00002390	0056879A	4D61A28C
After	PSD1	GPR7	Memory Word 027A4
	20002394	4D61A28C	4D61A28C



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

830134

DEFINITION

The effective doubleword location (EDL) specified by the effective doubleword address (EDA) is accessed and transferred to the general purpose register (GPR) specified by R and R+1. R+1 is the GPR one greater than that specified by R. The least-significant effective word location (EWL+1) is accessed first and transferred to R+1. The most-significant EWL is accessed last and transferred to R.

NOTE

The GPR specified by R must be an even-numbered register.

SUMMARY EXPRESSION

(EWL+1) → R+1

(EWL) → R

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R,R+1) is greater than zero
CC3: Is set if (R,R+1) is less than zero
CC4: Is set if (R,R+1) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of memory word 008B7C are transferred to GPR7 and the contents of memory word 008B78 are transferred to GPR6. CC3 is set.

Memory Location:	281C4
Hexadecimal Instruction:	AF06800A (R=6, X=0, BR=6)
Assembly Language Coding:	LD 6,X'8008' (6)

Before	PSD1	GPR6	GPR7	BR6
	420281C4	03F609C3	39BB510E	00000B70
	Memory Word 008B78		Memory Word 008B7C	
	F05B169A		137F8CA2	
After	PSD1	GPR6	GPR7	BR6
	120281C8	F05B169A	137F8CA2	00000B70
	Memory Word 008B78		Memory Word 008B7C	
	F05B169A		137F8CA2	

NONBASE REGISTER MODE EXAMPLE

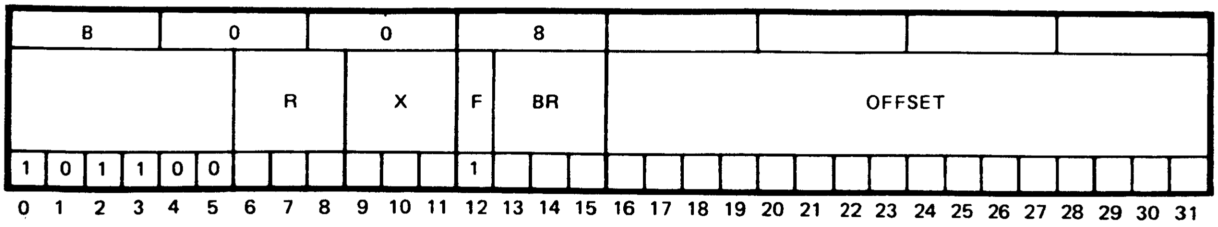
The contents of memory word 28B7C are transferred to GPR7 and the contents of memory word 28B78 are transferred to GPR6. CC3 is set.

Memory Location:	281C4
Hexadecimal Instruction:	AF 02 8B 7A (R=6, X=0, I=0)
Assembly Language Coding:	LD 6,X'28B78'

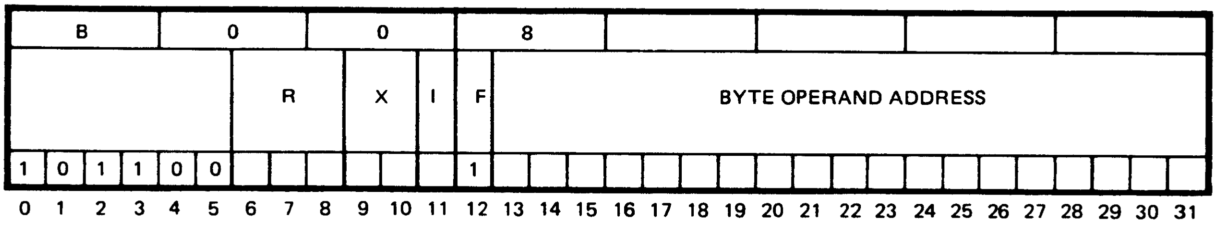
Before	PSD1	GPR6	GPR7
	400281C4	03F609C3	39BB510E
	Memory Word 28B78	Memory Word 28B7C	
	F05B169A	137F8CA2	
After	PSD1	GPR6	GPR7
	100281C8	F05B169A	137F8CA2
	Memory Word 28B78	Memory Word 28B7C	
	F05B169A	137F8CA2	

LOAD MASKED BYTE
B008

LMB
d,*m,x



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

830135

DEFINITION

The effective byte location (EBL) specified by the effective byte address (EBA) is accessed and masked (logical AND function) with the least-significant byte (bits 24-31) of the mask register (R4). The result of the mask operation is transferred to bit positions 24-31 of the general purpose register (GPR) specified by R. Bit positions 0-23 of the GPR specified by R are cleared to zeros.

SUMMARY EXPRESSION

$(EBL)\&(R4_{24-31}) \rightarrow (R_{24-31})$

$\text{Zeros} \rightarrow R_{0-23}$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R_{0-31}) is greater than zero
CC3: Always zero
CC4: Is set if (R_{0-31}) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of memory byte 0000A3 are logically ANDed with the rightmost byte of GPR4; the result is transferred to bits 24-31 of GPR1. Bits 0-23 of GPR1 are cleared. CC2 is set.

Memory Location:	00900
Hexadecimal Instruction:	B08E00A0 (R=1, X=0, BR=6)
Assembly Language Coding:	LMB 1,X'A0'(6)

Before	PSD1	GPR1	GPR4	BR6	Memory Byte 0000A3
	02000900	AA3689B0	000000F0	00000003	29
After	PSD1	GPR1	GPR4	BR6	Memory Byte 0000A3
	22000904	00000020	000000F0	00000003	29

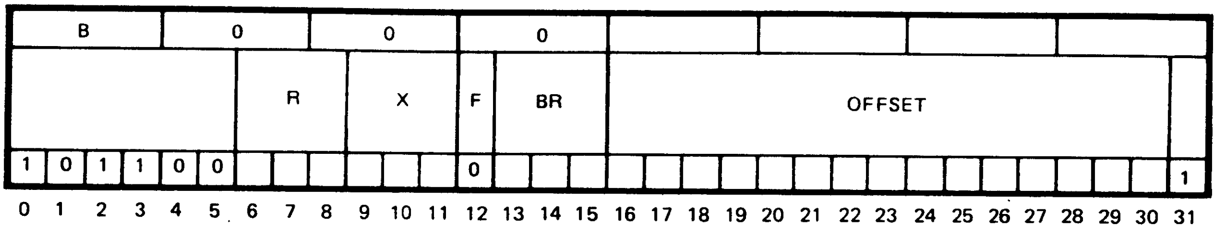
NONBASE REGISTER MODE EXAMPLE

The contents of memory byte 000A3 are logically ANDed with the rightmost byte of GPR4; the result is transferred to bits 24-31 of GPR1. Bits 0-23 of GPR1 are cleared. CC2 is set.

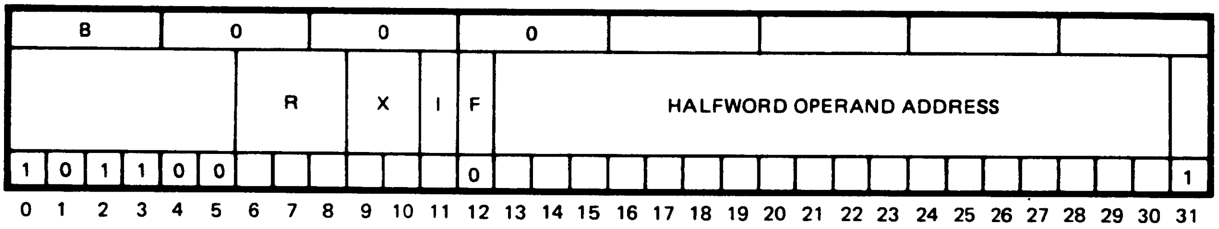
Memory Location:	00900
Hexadecimal Instruction:	B0 88 00 A3 (R=1, X=0, I=0)
Assembly Language Coding:	LMB 1,X'A3'

Before	PSD1	GPR1	GPR4	Memory Byte 000A3
	00000900	AA3689B0	000000F0	29
After	PSD1	GPR1	GPR4	Memory Byte 000A3
	20000904	00000020	000000F0	29

LOAD MASKED HALFWORD B000 — LMH d,*m,x



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

DEFINITION

The effective halfword location (EHL) specified by the effective halfword address (EHA) is accessed, and the sign bit (bit 16) is extended 16 bit positions to the left to form a word. This word is then masked (logical AND function) with the contents of the mask register (R4). The resulting word is transferred to the general purpose register (GPR) specified by R.

SUMMARY EXPRESSION

(EHL)SE&(R4) → R

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R_{0-31}) is greater than zero
CC3: Is set if (R_{0-31}) is less than zero
CC4: Is set if (R_{0-31}) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of memory halfword 0003A3 are accessed, the sign is extended 16 bit positions, the result is logically ANDed with the contents of GPR4, and the final result is transferred to GPR5. CC2 is set.

Memory Location:	00300
Hexadecimal Instruction:	B2860303 (R=5, X=0, BR=6)
Assembly Language Coding:	LMA 5,X'303'(6)

Before	PSD1	GPR4	GPR5	BR6
	02000300	0FF00FF0	C427B319	000000A0
	Memory Halfword 0003A3			
	A58D			
After	PSD1	GPR4	GPR5	BR6
	22000304	0FF00FF0	000580	000000A0
	Memory Halfword 0003A3			
	A58D			

NONBASE REGISTER MODE EXAMPLE

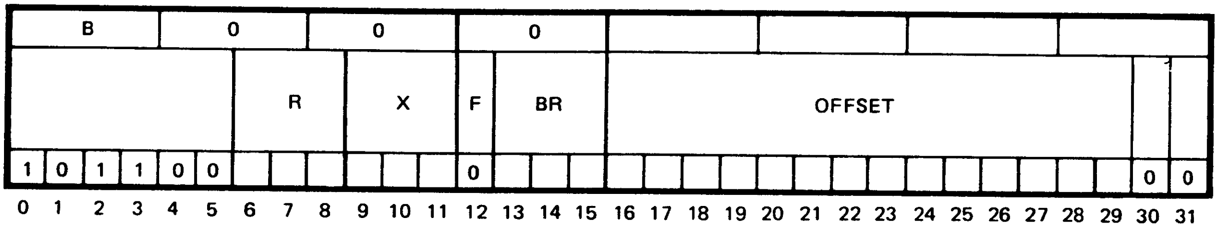
The contents of memory halfword 003A3 are accessed, the sign is extended 16 bit positions, the result is logically ANDed with the contents of GPR4, and the final result is transferred to GPR5. CC2 is set.

Memory Location:	00300
Hexadecimal Instruction:	B2 80 03 A3 (R=5, X=0, I=0)
Assembly Language Coding:	LMH 5,X'3A3'

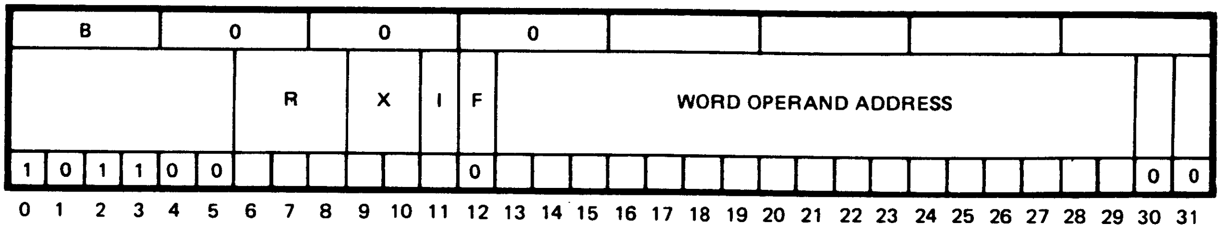
Before	PSD1	GPR4	GPR5	Memory Halfword 003A3
	08000300	0FF00FF0	C427B319	A58D
After	PSD1	GPR4	GPR5	Memory Halfword 003A3
	20000304	0FF00FF0	000580	A58D

LOAD MASKED WORD
B000

LMW
d,*m,x



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

DEFINITION

The effective word location (EWL) specified by the effective word address (EWA) is accessed and masked (logical AND function) with the contents of the mask register (R4). The resulting word is transferred to the general purpose register (GPR) specified by R.

SUMMARY EXPRESSION

(EWL)&(R4) → R

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R_{0-31}) is greater than zero
CC3: Is set if (R_{0-31}) is less than zero
CC4: Is set if (R_{0-31}) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of memory word 000FFC are ANDed with the contents of GPR4. The result is transferred to GPR7. CC3 is set.

Memory Location:	00FF0
Hexadecimal Instruction:	B3860FF0 (R=7, X=0, BR=6)
Assembly Language Coding:	LMW 7,X'FF0'(6)

Before	PSD1	GPR4	GPR7	BR6	Memory Word 000FFC
	02000F00	FF00007C	12345678	0000000C	8923F8E8
After	PSD1	GPR4	GPR7	BR6	Memory Word 000FFC
	12000F04	FF00007C	89000068	0000000C	8923F8E8

NONBASE REGISTER MODE EXAMPLE

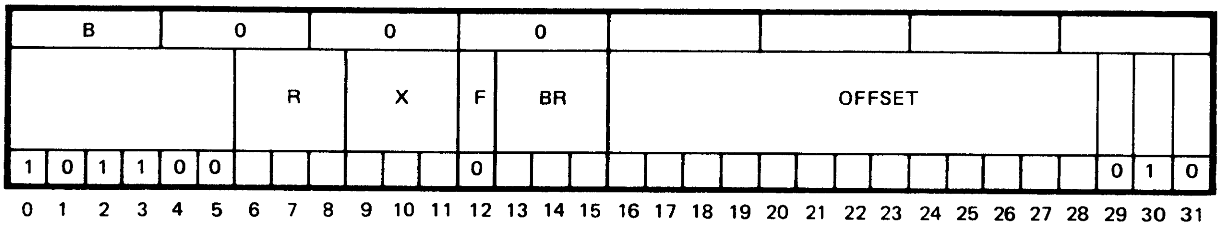
The contents of memory word 00FFC are ANDed with the contents of GPR4. The result is transferred to GPR7, and CC3 is set.

Memory Location:	00F00
Hexadecimal Instruction:	B3800FFC (R=7, X=0, I=0)
Assembly Language Coding:	LMW 7,X'FFC'

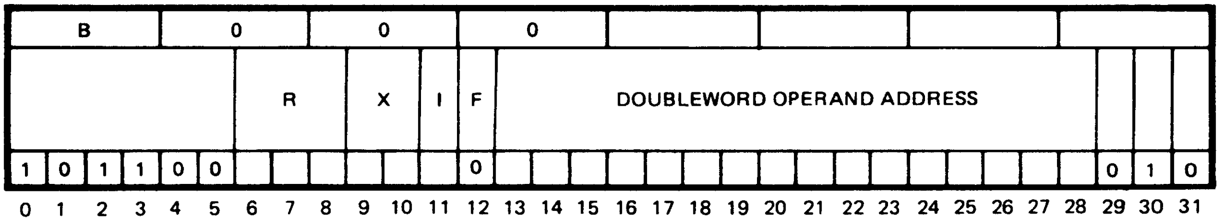
Before	PSD1	GPR4	GPR7	Memory Word 00FFC
	00000F00	FF00007C	12345678	8923F8E8
After	PSD1	GPR4	GPR7	Memory Word 00FFC
	10000F04	FF00007C	89000068	8923F8E8

LOAD MASKED DOUBLEWORD
B000

LMD
d,*m,x



BASE REGISTER FORMAT



830138

NONBASE REGISTER FORMAT

DEFINITION

The effective doubleword location (EDL) specified by the effective doubleword address (EDA) is accessed, and the contents of each word are masked (logical AND function) with the contents of the mask register (R4). The least-significant effective word location (EWL+1) is masked first and transferred to R+1. The most-significant EWL is then masked and transferred to GPR specified by R. R+1 is the GPR one greater than R.

NOTE

The GPR specified by R must be an even-numbered register.

SUMMARY EXPRESSION

(EWL+1)&(R4) → R+1

(EWL)&(R4) → R

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R,R+1) is greater than zero
CC3: Is set if (R,R+1) is less than zero
CC4: Is set if (R,R+1) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of memory word 0002F4 are ANDed with the content of GPR4, and the result is transferred to GPR7. The contents of memory word 0002F0 are ANDed with GPR4, and the result is transferred to GPR6. CC2 is set.

Memory Location:	00200
Hexadecimal Instruction:	B3060202 (R=6, X=0, BR=6)
Assembly Language Coding:	LMD 6,X'200'(6)

Before	PSD1	GPR4	GPR6	GPR7	BR6
	02000200	3F3F3F3F	12345678	9ABCDEF0	000000F0
	Memory Word 0002F0		Memory Word 0002F4		
	AE69D10C		63B208F0		
After	PSD1	GPR4	GPR6	GPR7	BR6
	22000204	3F3F3F3F	2E29110C	23320830	000000F0
	Memory Word 0002F0		Memory Word 0002F4		
	AE69D10C		63B208F0		

NONBASE REGISTER MODE EXAMPLE

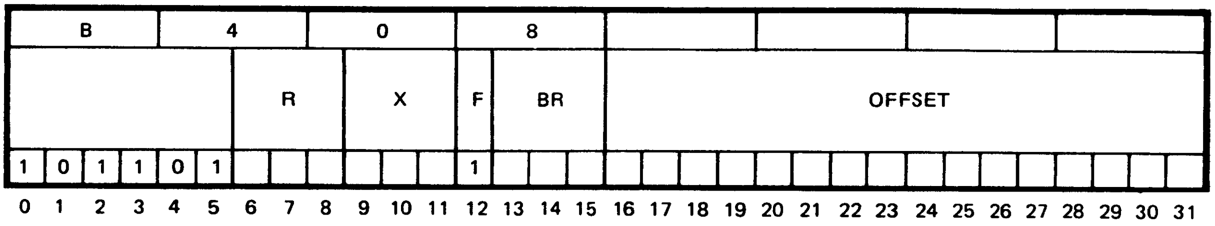
The contents of memory word 002F4 is ANDed with the content of GPR4, and the result is transferred to GPR7. The contents of memory word 002F0 is ANDed with the contents of GPR4, and the result is transferred to GPR6. CC2 is set.

Memory Location:	00200
Hexadecimal Instruction:	B3 00 02 F2 (R=6, X=0, I=0)
Assembly Language Coding:	LMD 6,X'2F0'

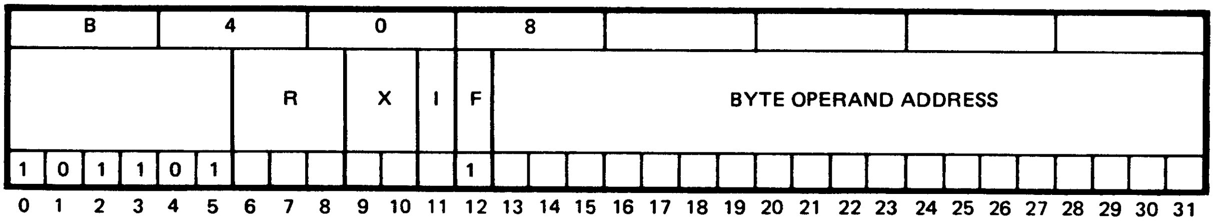
Before	PSD1	GPR4	GPR6	GPR7
	00000200	3F3F3F3F	12345678	9ABCDEF0
	Memory Word 002F0		Memory Word 002F4	
	AE69D10C		63B208F0	
After	PSD1	GPR4	GPR6	GPR7
	20000204	3F3F3F3F	2E29110C	23320830
	Memory Word 002F0		Memory Word 002F4	
	AE69D10C		63B208F0	

LOAD NEGATIVE BYTE
B408

LNB
d,*m,x



BASE REGISTER FORMAT



830139

NONBASE REGISTER FORMAT

DEFINITION

The effective byte location (EBL) specified by the effective byte address (EBA) is accessed, and 24 zeros are appended to the most-significant end to form a word. The twos complement of this word is then taken and transferred to the general purpose register (GPR) specified by R.

SUMMARY EXPRESSION

- 00-23, (EBL) → R

CONDITION CODE RESULTS

CC1: Always zero
CC2: Always zero
CC3: Is set if (R_{0-31}) is less than zero
CC4: Is set if (R_{0-31}) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of memory byte 00D102 are prefixed with 24 zeros to form a word; the result is negated and transferred to GPR1. CC3 is set.

Memory Location:	0D000
Hexadecimal Instruction:	B48ED100 (R=1, X=0, BR=6)
Assembly Language Coding:	LNB 1,X'D100'(6)

Before	PSD1	GPR1	BR6	Memory Byte 00D102
	0200D000	00000000	00000002	3A
After	PSD1	GPR1	BR6	Memory Byte 00D102
	0A00D004	FFFFFFC6	00000002	3A

NONBASE REGISTER MODE EXAMPLE

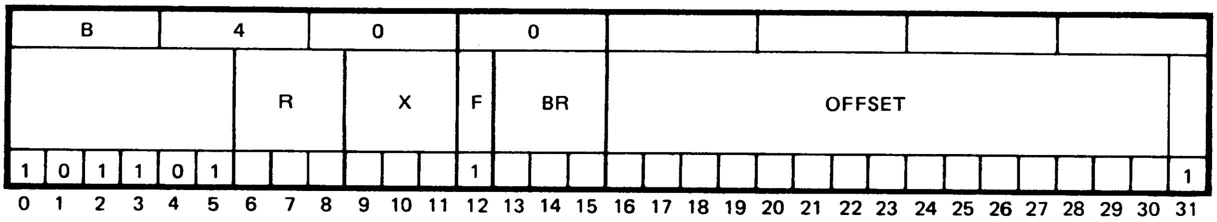
The contents of memory byte 0D102 are prefixed with 24 zeros to form a word; the result is negated and transferred to GPR1. CC3 is set.

Memory Location:	0D000
Hexadecimal Instruction:	B4 88 D1 02 (R=1, X=0, I=0)
Assembly Language Coding:	LNB 1,X'D102'

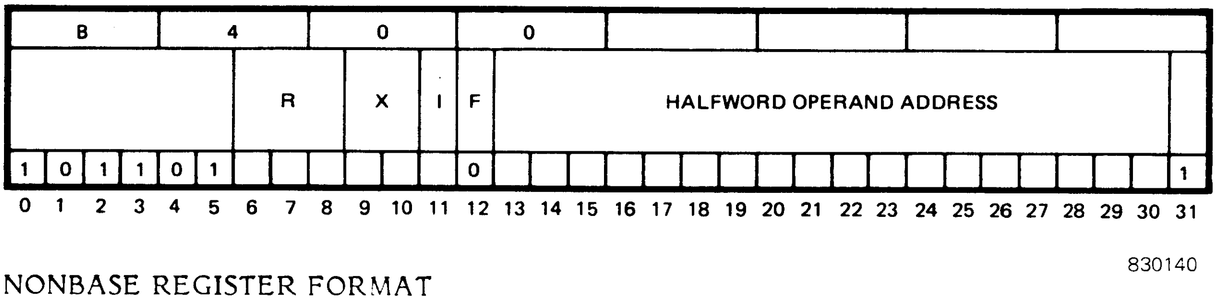
Before	PSD1	GPR1	Memory Byte 0D102
	0000D000	00000000	3A
After	PSD1	GPR1	Memory Byte 0D102
	0800D004	FFFFFFC6	3A

LOAD NEGATIVE HALFWORD B400

LNH d,*m,x



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

DEFINITION

The effective halfword location (EHL) specified by the effective halfword address (EHA) is accessed and the sign bit (bit 16) is extended 16 bit positions to the left to form a word. The twos complement of this word is then transferred to the general purpose register (GPR) specified by R.

SUMMARY EXPRESSION

$- (EHL)_{SE} \rightarrow R$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R_{0-31}) is greater than zero
CC3: Is set if (R_{0-31}) is less than zero
CC4: Is set if (R_{0-31}) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of memory halfword 0084B2 are sign extended and negated. The result is transferred to GPR4. CC2 is set.

Memory Location:	08000
Hexadecimal Instruction:	B6068403 (R=4, X=0, BR=6)
Assembly Language Coding:	LNH 4,X'8402'(6)

Before	PSD1	GPR4	BR6	Memory Halfword 0084B2
	42008000	12345678	000000B0	960C
After	PSD1	GPR4	BR6	Memory Halfword 0084B2
	22008004	000069F4	000000B0	960C

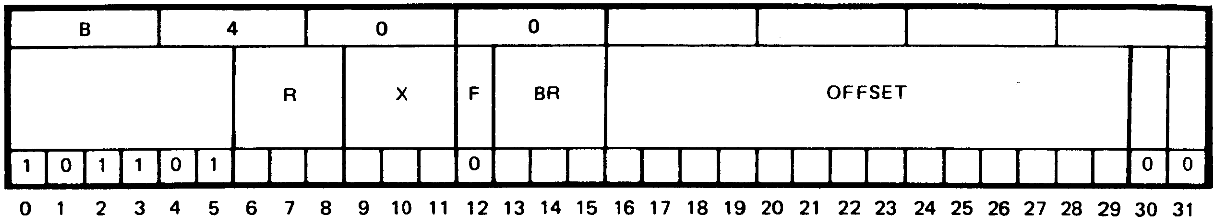
NONBASE REGISTER MODE EXAMPLE

The contents of memory halfword 08402 are sign extended and negated. The result is transferred to GPR4. CC2 is set.

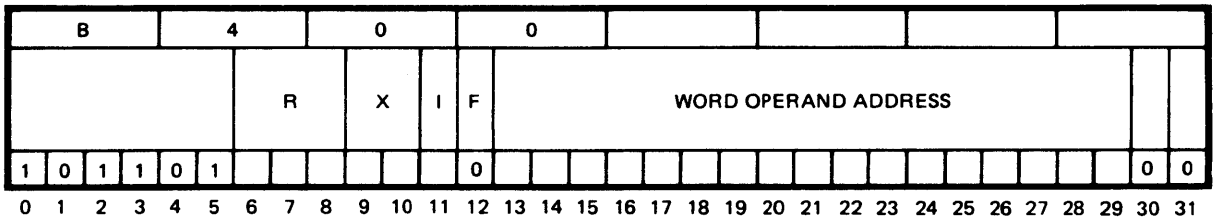
Memory Location:	08000
Hexadecimal Instruction:	B6 00 84 03 (R=4, X=0, I=0)
Assembly Language Coding:	LNH 4,X'8402'

Before	PSD1	GPR4	Memory Halfword 08402
	40008000	12345678	960C
After	PSD1	GPR4	Memory Halfword 08402
	20008004	000069F4	960C

LOAD NEGATIVE WORD LNW
B400 d,*m,x



BASE REGISTER FORMAT



830141

NONBASE REGISTER FORMAT

DEFINITION

The effective word location (EWL) specified by the effective word address (EWA) is accessed, and its twos complement is transferred to the general purpose register (GPR) specified by R.

SUMMARY EXPRESSION

-(EWL) → R

CONDITION CODE RESULTS

CC1: Is set if arithmetic exception
CC2: Is set if (R_{0-31}) is greater than zero
CC3: Is set if (R_{0-31}) is less than zero
CC4: Is set if (R_{0-31}) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of memory word 0006C8 are negated and transferred to GPR5. CC3 is set.

Memory Location:	00500
Hexadecimal Instruction:	B68606C0 (R=5, X=0, BR=6)
Assembly Language Coding:	LNW 5,X'06C0' (6)

Before	PSD1	GPR5	BR6	Memory Word 0006C8
	0A000500	00000000	00000008	185E0D76
After	PSD1	GPR5	BR6	Memory Word 0006C8
	12000504	E7A1F28A	00000008	185E0D76

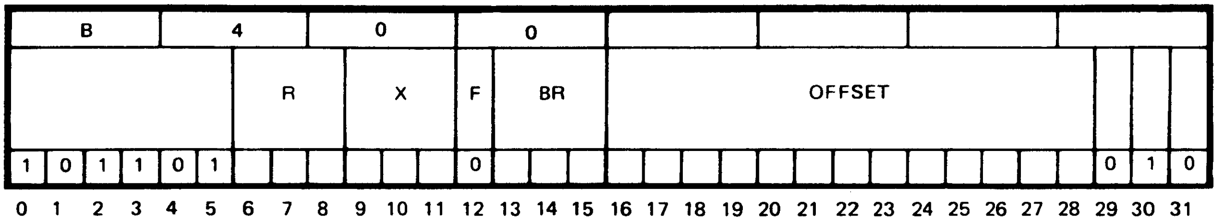
NONBASE REGISTER MODE EXAMPLE

The contents of memory word 006C8 are negated and transferred to GPR5. CC3 is set.

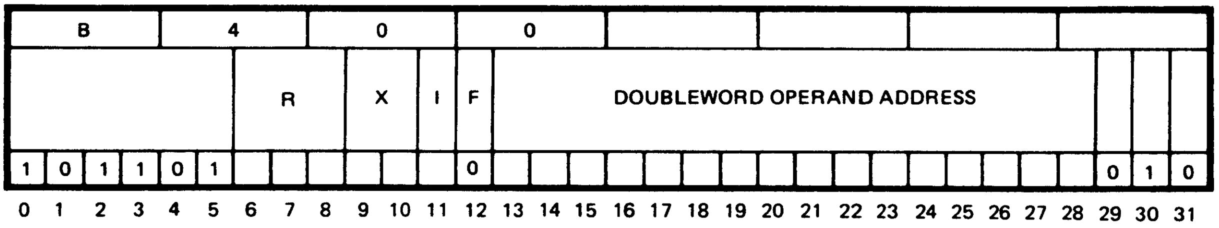
Memory Location:	00500
Hexadecimal Instruction:	B6 80 06 C8 (R=5, X=0, I=0)
Assembly Language Coding:	LNW 5,X'6C8'

Before	PSD1	GPR5	Memory Word 006C8
	08000500	00000000	185E0D76
After	PSD1	GPR5	Memory Word 006C8
	10000504	E7A1F28A	185E0D76

LOAD NEGATIVE DOUBLEWORD **LND**
B400 **d,*m,x**



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

830142

DEFINITION

The effective doubleword location (EDL) specified by the effective doubleword address (EDA) is accessed and its twos complement is formed. The least-significant effective word location (EWL+1) is complemented first and the result is transferred to R+1. The most-significant word is complemented, and the result is transferred to the GPR specified by R. R+1 is the GPR one greater than that specified by R.

Note

The GPR specified by R must be an even numbered register.

SUMMARY EXPRESSION

-(EWL+1) → R+1
-(EWL) → R

CONDITION CODE RESULTS

CC1: Is set if arithmetic exception
CC2: Is set if (R,R+1) is greater than zero
CC3: Is set if (R,R+1) is less than zero
CC4: Is set if (R,R+1) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of memory word 0024A4 is negated and transferred to GPR3. The contents of memory word 0024A0 is negated and transferred to GPR2. CC3 is set.

Memory Location:	02344
Hexadecimal Instruction:	B5062002 (R=2, X=0, BR=6)
Assembly Language Coding:	LND 2,X'2000'(6)

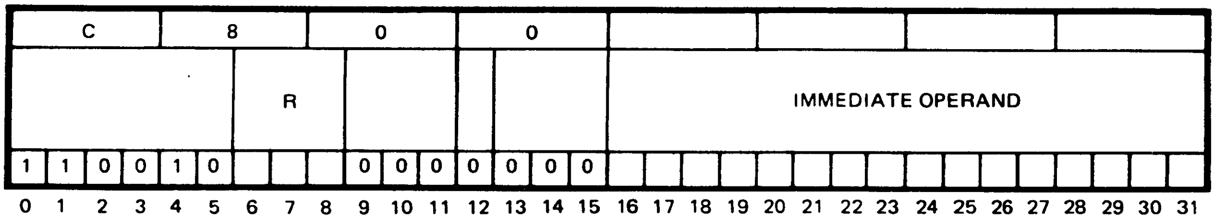
Before	PSD1	GPR2	GPR3	BR6
	02002344	01234567	89ABCDEF	000004A0
	Memory Word 0024A0		Memory Word 0024A4	
	00000000		00000001	
After	PSD1	GPR2	GPR3	BR6
	12002348	FFFFFFFF	FFFFFFFF	000004A0
	Memory Word 0024A0		Memory Word 0024A4	
	00000000		00000001	

NONBASE REGISTER MODE EXAMPLE

The contents of memory word 024A4 is negated and transferred to GPR3. The contents of memory word 0024A0 is negated and transferred to GPR2. CC3 is set.

Memory Location:	02344
Hexadecimal Instruction:	B5 00 24 A2 (R=2, X=0, I=0)
Assembly Language Coding:	LND 2,X'24A0'

Before	PSD1	GPR2	GPR3
	00002344	01234567	89ABCDEF
	Memory word 024A0		Memory Word 024A4
	00000000		00000001
After	PSD1	GPR2	GPR3
	10002348	FFFFFFFF	FFFFFFFF
	Memory Word 024A0		Memory Word 024A4
	00000000		00000001



830143

DEFINITION

The halfword immediate operand in the instruction word (IW) is sign-extended (bit 16 extended 16 positions to the left) to form a word. This word is transferred to the general purpose register (GPR) specified by R.

SUMMARY EXPRESSION

$(IW_{16-31})SE \rightarrow R$

CONDITION CODE RESULTS

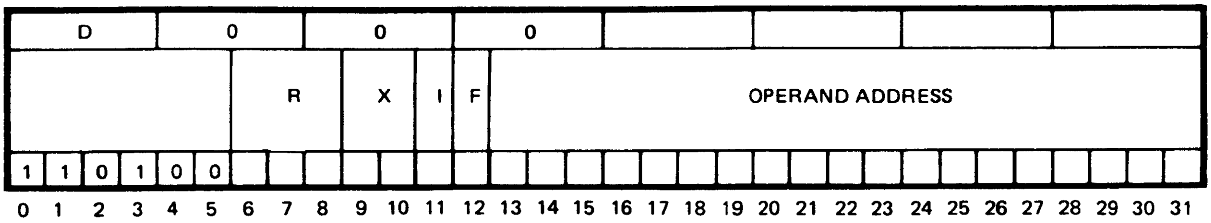
CC1: Always zero
CC2: Is set if (R_{0-31}) is greater than zero
CC3: Is set if (R_{0-31}) is less than zero
CC4: Is set if (R_{0-31}) is equal to zero

NONBASE AND BASE REGISTER MODE EXAMPLE

The halfword operand is sign-extended and the result is transferred to GPR1. CC3 is set.

Memory Location:	0630C
Hexadecimal Instruction:	C8 80 FF FB (R=1)
Assembly Language Coding:	LI 1,-5

Before	PSD1 0000630C (Nonbase) 0200630C (Base)	GPR1 12345678
After	PSD1 10006310 (Nonbase) 12006310 (Base)	GPR1 FFFFFFFB



DEFINITION

The effective address (EA) of the operand is generated and transferred to the general purpose register (GPR) specified by R.

SUMMARY EXPRESSION

Nonextended Addressing Mode

$EA \rightarrow R_{13-31}$

$F \rightarrow R_{12}$

$\text{Zeros} \rightarrow R_{2-11}$

Extended Addressing Mode

$EA \rightarrow R_{8-31}$

$\text{Zeros} \rightarrow R_{2-7}$

In both addressing modes, nonextended and extended, bits 0 and 1 of the GPR are contingent upon the indirect bit (I) of the instruction:

If I=0, ones $\rightarrow R_{0-1}$

If I=1, the contents of bits 0 and 1 of the last word of the indirect chain are copied into bits 0 and 1 of the GPR, respectively.

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

Note

This instruction is illegal in the base register mode.

NONBASE REGISTER MODE EXAMPLE

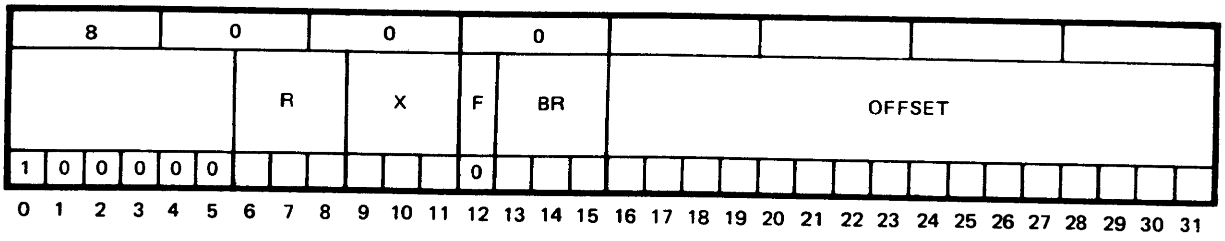
The effective address (EA) is transferred to general purpose register (GPR) R1, bits 13-31. Bits 0 and 1 of R1 is set to denote no indirect addressing.

Memory Location:	1000
Hexadecimal Instruction:	D0 80 40 00 (R=1, X=0, I=0)
Assembly Language Coding:	LEA 1,X'4000'

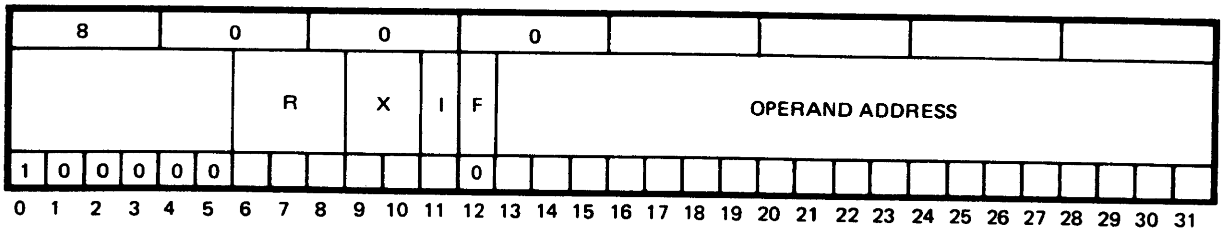
Before	PSD1	GPR1	Memory Word 4000
	08001000	00000000	AC881203
After	PSD1	GPR1	Memory Word 4000
	08001004	C0004000	AC881203

LOAD EFFECTIVE ADDRESS REAL 8000

LEAR d,*m,x



BASE REGISTER FORMAT



830145

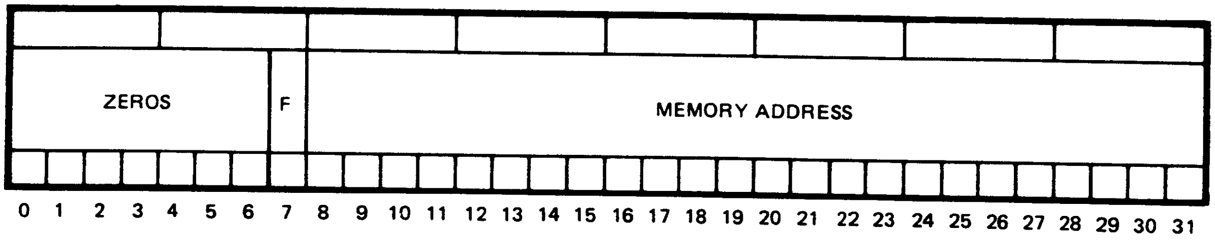
NONBASE REGISTER FORMAT

DEFINITION

This instruction stores the real memory address (MA) of the operand in the general purpose register (GPR) specified by R.

NOTE

> In base and nonbase register mode the format of the 25-bit memory address transferred to the GPR is as follows:



830355

SUMMARY EXPRESSION

Zeros → R_{0-6}

F → R_7

MA → R_{8-31}

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

NOTE

In the mapped mode, if the referenced map entry is invalid, the CPU will assert the demand page fault trap. This instruction constitutes an access to the page, and thus will set the MID access (A) bit if necessary.

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The real memory address (MA) is transferred to bits 8-31 of general purpose register (GPR) R1.

Memory Location:	01000
Hexadecimal Instruction:	83062000 (R=6, X=0, BR=6)
Assembly Language Coding:	LEAR 6,X'2000'(6)

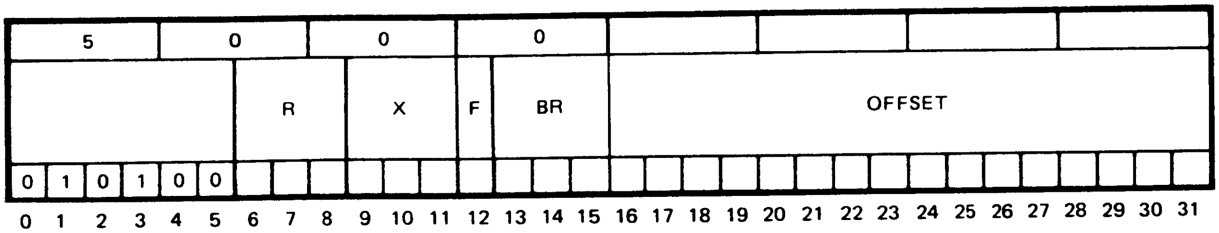
Before	PSD1	GPR6	BR6	CPU IN MAP MODE
	A2001000	00055555	00000050	MAP BLOCK 1=X'8001'
	PSD2			
	8000XXXX			XXXX=CPIX
After	PSD1	GPR6	BR6	
	A2001004	00002050	00000050	MAP BLOCK 1=X'8801'
	PSD2			
	8000XXXX			XXXX=CPIX

NONBASE REGISTER MODE EXAMPLE

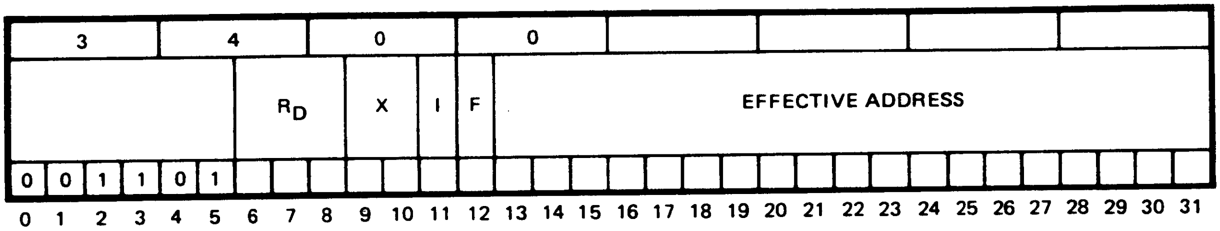
Memory Location:	01000
Hexadecimal Instruction:	83 00 20 00 (R=6, X=0, I=0)
Assembly Language Coding:	LEAR 6,X'2000'

Before	PSD1	GPR6	CPU IN MAP MODE
	A0001000	00055555	MAP BLOCK 1=X'8001'
	PSD2		
	8000XXXX		XXXX=CPIX
After	PSD1	GPR6	
	A0001004	00002000	MAP BLOCK 1=X'8801'
	PSD2		
	8000XXXX		XXXX=CPIX

LOAD ADDRESS



BASE REGISTER FORMAT (5000)



NONBASE REGISTER FORMAT (3400)

DEFINITION

The effective address (EA) of the operand is generated and stored in the general purpose register (GPR) specified by R.

SUMMARY EXPRESSION

Nonextended Addressing Mode

Zeros → R_{0-11}

F → R_{12}

EA → R_{13-31}

Extended Addressing Mode or Base Register Mode

Zeros → R_{0-7}

EA → R_{8-31}

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

NOTE

No address specification checks are performed.

BASE REGISTER MODE EXAMPLE

The contents of BR6, the contents of index register (X) GPR2 and the instruction offset are added to obtain the logical address. This address is loaded into GPR4.

Memory Location:	1000
Hexadecimal Instruction:	52261300 (R=4, X=2, BR=6)
Assembly Language Coding:	LA 4,X'1300'(6), 2

Before	PSD1	GPR2	GPR4	BR6
	22001000	00001000	02020202	00000050
After	PSD1	GPR2	GPR4	BR6
	22001004	00001000	00002350	00000050

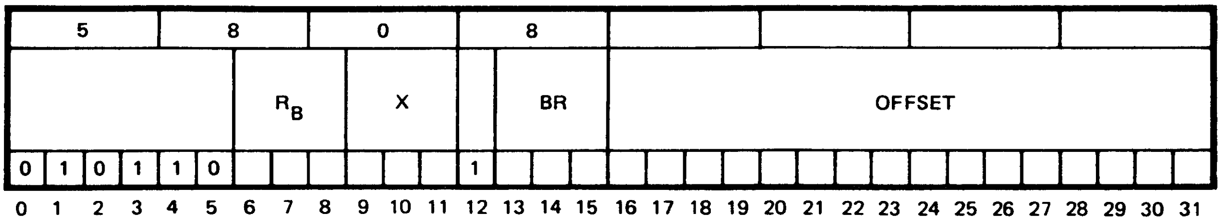
NONBASE REGISTER MODE EXAMPLE

Memory Location:	1000
Hexadecimal Instruction:	36 40 13 00 (R=4, X=2, I=0)
Assembly Language Coding:	LA 4,X'1300',2

Before	PSD1	GPR2	GPR4
	20001000	00001000	02020202
After	PSD1	GPR2	GPR4
	20001004	00001000	00002300

LOAD ADDRESS BASE REGISTER
5808

LABR
d,m,x



830147

DEFINITION

Load the effective address (EA) into the base register specified by R_B.

SUMMARY EXPRESSION

EA → R_B 8-31

Zeros → R_B0-7

CONDITION CODE RESULTS

CC1: Unchanged
CC2: Unchanged
CC3: Unchanged
CC4: Unchanged

NOTES

1. This instruction is used for the base register mode only.

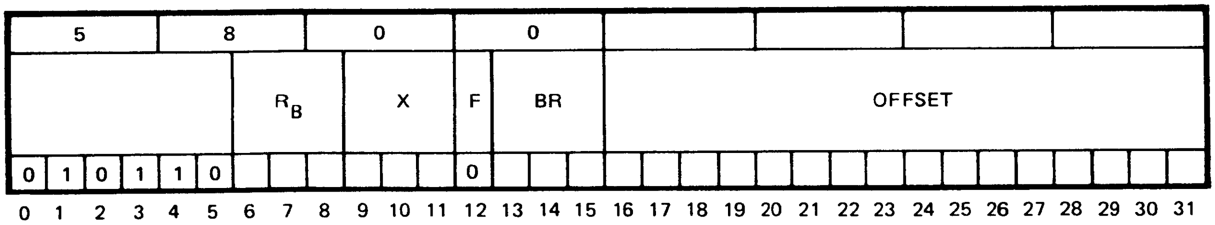
2. No address specification checks are performed.

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The address is loaded into base register R_B2.

Memory Location:	2000
Hexadecimal Instruction:	590E2000 (R_B=2, X=0, BR=6)
Assembly Language Coding:	LABR 2,X'2000'(6)

Before	PSD1	R_B2	BR6
	02002000	12345678	00000050
After	PSD1	R_B2	BR6
	02002004	00002050	00000050



DEFINITION

830148

The effective address (EA) formed by the sum of the contents of the specified base register (BR), offset, and index register is subtracted from the contents of the base register specified by R_B. The result is stored in the base register specified by R_B.

SUMMARY EXPRESSION

R_B-(BR+I+OFFSET) → R_B

CONDITION CODE RESULTS

CC1: Unchanged
CC2: Unchanged
CC3: Unchanged
CC4: Unchanged

NOTES

1. This instruction is used for the base register mode only.

2. No address specification checks are performed.

BASE REGISTER MODE EXAMPLE

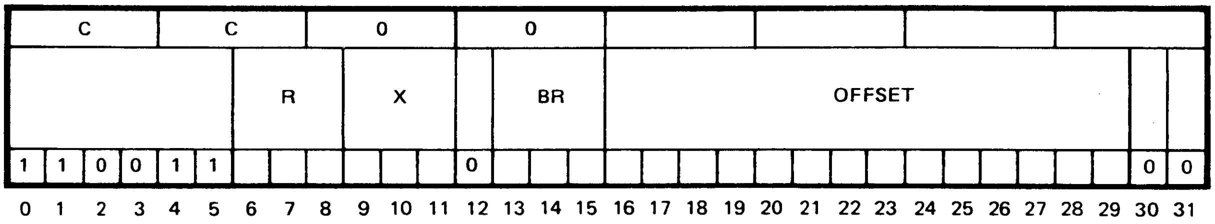
The contents of BR6 and the instruction offset are added to obtain the logical address. This address is subtracted from the contents of base register R_B2. The result is stored in R_B2.

Memory Location:	03000
Hexadecimal Instruction:	59062000 (R_B=2, X=0, BR=6)
Assembly Language Coding:	SUABR 2,X'2000'(6)

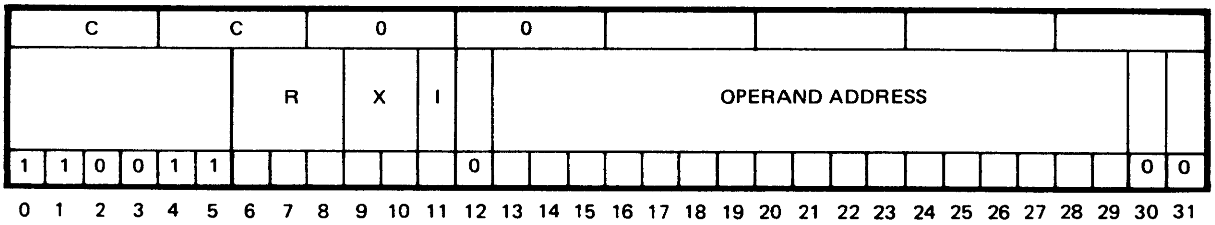
Before	PSD1 02003000	R_B2 00004050	BR6 00000050
After	PSD1 02003004	R_B2 00002000	BR6 00000050

LOAD FILE
CC00

LF
d,*m,x



BASE REGISTER FORMAT



830149

NONBASE REGISTER FORMAT

DEFINITION

This instruction is used to load one to eight general purpose registers (GPR). The effective word location (EWL) specified by the effective word address (EWA) in the instruction word is accessed and transferred to the GPR specified by R. Next, the EWA and the GPR address are incremented. The next sequential EWL is then transferred to the next sequential GPR. Successive transfers continue until GPR7 is loaded from memory.

NOTES

1. If the F and C bits are changed during indexing or indirection, such final address specified is not a word address, and an address specification trap will occur.

2. If an operand address generated by this instruction crosses a MAP block boundary, an address specification trap will occur. This applies when in either the mapped or unmapped mode.

SUMMARY EXPRESSION

(EWL) → R

(EWL+1) → R+1
. .
. .
. .
(EWL+n) → R7

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of memory word 000240 are transferred to GPR4, of memory word 000244 to GPR5, of memory word 000248 to GPR6, and of memory word 00024C to GPR7.

Memory Location:	00300
Hexadecimal Instruction:	CE060200 (R=4, X=0, BR=6)
Assembly Language Coding:	LF 4,X'200'(6)

Before	PSD1	GPR4	GPR5	GPR6	GPR7	BR6
	0A000300	00000000	00000000	00000000	00000000	00000040
	Memory Word 000240			Memory Word 000244		
	00000001			00000002		
	Memory Word 000248			Memory Word 00024C		
	00000003			00000004		
After	PSD1	GPR4	GPR5	GPR6	GPR7	BR6
	0A000304	00000001	00000002	00000003	00000004	00000040
	Memory Word 000240			Memory Word 000244		
	00000001			00000002		
	Memory Word 000248			Memory Word 00024C		
	00000003			00000004		

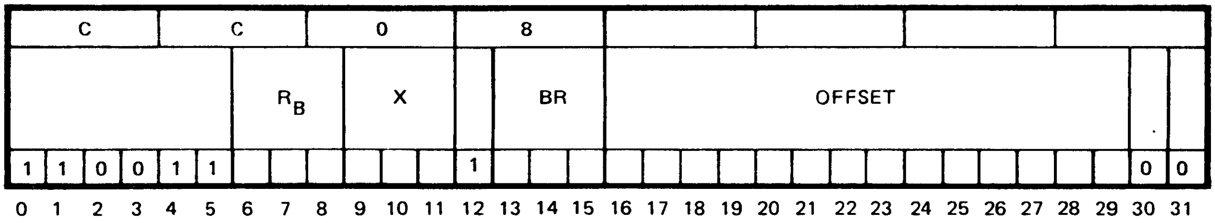
NONBASE REGISTER MODE EXAMPLE

The contents of memory word 00200 are transferred to GPR4, of memory word 00204 to GPR5, of memory word 00208 to GPR6, and of memory word 0020C to GPR7.

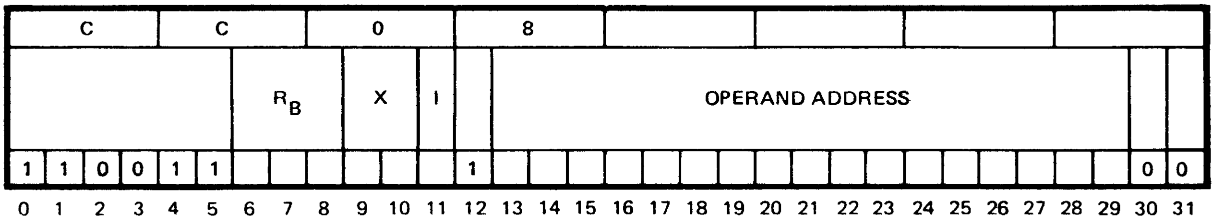
Memory Location:	00300
Hexadecimal Instruction:	CE 00 02 00 (R=4, X=0, I=0)
Assembly Language Coding:	LF 4,X'200'

Before	PSD1	GPR4	GPR5	GPR6	GPR7
	08000300	00000000	00000000	00000000	00000000
	Memory Word 00200		Memory Word 00204		
	00000001		00000002		
	Memory Word 00208		Memory Word 0020C		
	00000003		00000004		
After	PSD1	GPR4	GPR5	GPR6	GPR7
	08000304	00000001	00000002	00000003	00000004
	Memory Word 00200		Memory Word 00204		
	00000001		00000002		
	Memory Word 00208		Memory Word 0020C		
	00000003		00000004		

LOAD BASE FILE LFBR
CC08 d,*m,x



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

DEFINITION

This instruction is used to load one to eight base registers. The contents of the effective word location (EWL), as addressed by the effective word address (EWA), is accessed and transferred to the base register specified R_B. Next, the EWA and R_B are incremented and the next sequential memory word is transferred to the next R. Successive transfers continue until R_B7 is loaded from memory.

SUMMARY EXPRESSION

$$\begin{array}{lcl} (EWL) & \rightarrow & R_B \\ (EWL+1) & \rightarrow & R_B+1 \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \cdot & & \cdot \\ (EWL+n) & \rightarrow & R_B+7 \end{array}$$

CONDITION CODE RESULTS

CC1:	Unchanged
CC2:	Unchanged
CC3:	Unchanged
CC4:	Unchanged

NOTES

1. This instruction may be executed in either Base Register Mode or Nonbase Register Mode.
2. Although the F bit is set in this instruction, which normally indicates a byte operand, the operand for this instruction is a set of one or more words.
3. If the effective word address (EWA) bits 30 and 31 are not zero, an address specification trap will occur.
4. If an operand address generated by this instruction crosses a MAP block boundary, an address specification trap will occur. This applies when in either the mapped or unmapped mode.

BASE REGISTER MODE EXAMPLE

The contents of BR3 and the instruction offset are added to obtain the logical address. The contents of memory word 000220 are transferred to $R_{B}4$, of memory word 000224 to $R_{B}5$, of memory word 000228 to $R_{B}6$, and of memory word 00022C to $R_{B}7$.

Memory Location:	02000
Hexadecimal Instruction:	CE0B0200 (R_{B}=4, X=0, BR=3)
Assembly Language Coding:	LFBR 4,X'200'(3)

Before	PSD1	$R_{B}4$	$R_{B}5$	$R_{B}6$	$R_{B}7$	BR3
	02002000	00000000	00000000	00000000	00000000	00000020
	Memory Word 000220			Memory Word 000224		
	00000001			00000002		
	Memory Word 000228			Memory Word 00022C		
	00000003			00000004		
After	PSD1	$R_{B}4$	$R_{B}5$	$R_{B}6$	$R_{B}7$	BR3
	02002004	00000001	00000002	00000003	00000004	00000020
	Memory Word 000220			Memory Word 000224		
	00000001			00000002		
	Memory Word 000228			Memory Word 00022C		
	00000003			00000004		

NONBASE REGISTER MODE EXAMPLE

The contents of memory word 000200 are transferred to $R_{B}4$, of memory word 000204 to $R_{B}5$, of memory word 000208 to $R_{B}6$, and of memory word 00020C to $R_{B}7$.

Memory Location:	02000
Hexadecimal Instruction:	CE080200 (R_{B}=4, X=0, I=0)
Assembly Language Coding:	LFBR 4,X'200'

Before	PSD1	$R_{B}4$	$R_{B}5$	$R_{B}6$	$R_{B}7$
	00002000	00000000	00000000	00000000	00000000
	Memory Word 0200			Memory Word 0204	
	00000001			00000002	
	Memory Word 0208			Memory Word 020C	
	00000003			00000003	
After	PSD1	$R_{B}4$	$R_{B}5$	$R_{B}6$	$R_{B}7$
	00002004	00000001	00000002	00000003	00000004
	Memory Word 0200			Memory Word 0204	
	00000001			00000002	
	Memory Word 0208			Memory Word 020C	
	00000003			00000004	

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
5				C				0				0																			
						R_B			X				BR			OFFSET															
0	1	0	1	1	1							0																		0	0

DEFINITION

The effective word location (EWL) specified by the effective word address (EWA) is accessed and transferred to the base register specified by R_B.

SUMMARY EXPRESSION

$(EWL) \rightarrow R_B$

CONDITION CODE RESULTS

CC1: Unchanged
CC2: Unchanged
CC3: Unchanged
CC4: Unchanged

NOTES

1. This instruction is used for the base register mode only.

2. If the effective word address (EWA) bits 30 and 31 are not zero, an address specification trap will occur.

BASE REGISTER MODE EXAMPLE

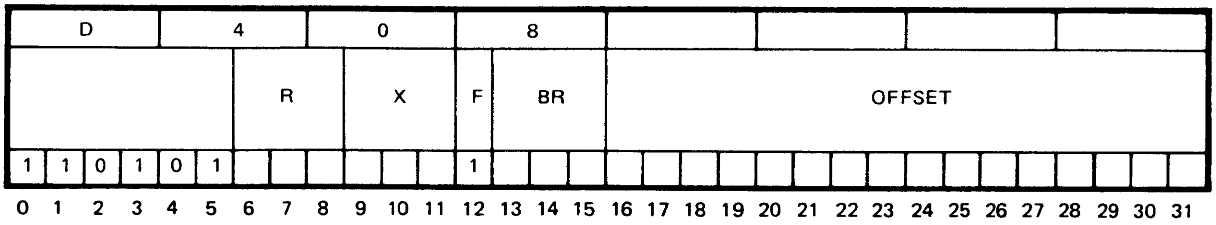
The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of memory word 000304 are transferred to R_B3.

Memory Location:	02000
Hexadecimal Instruction:	5D860300 (R_B3, X=0, BR=6)
Assembly Language Coding:	LWBR 3,X'300'(6)

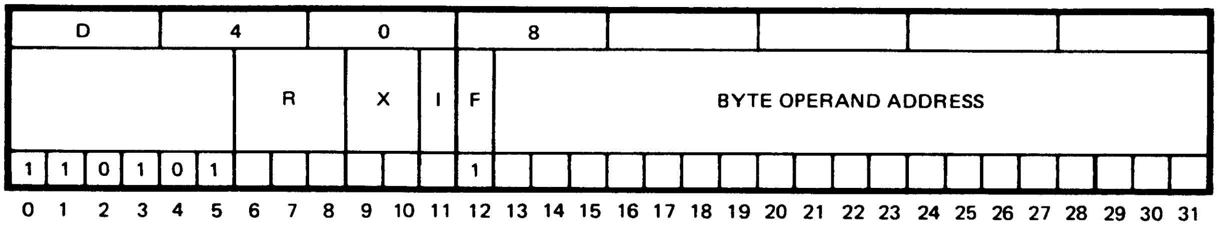
	PSD1	R_B3	BR6	Memory Word 000304
Before	02002000	00000000	00000004	12345678
	PSD1	R_B3	BR6	Memory Word 000304
After	02002004	12345678	00000004	12345678

STORE BYTE D408

STB s,*m,x



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

DEFINITION

The least-significant byte (bits 24-31) of the general purpose register (GPR) specified by R is transferred to the effective byte location (EBL) specified by the effective byte address (EBA) in the instruction word. The other three bytes of the memory word containing the byte specified by the EBA remain unchanged.

SUMMARY EXPRESSION

$(R_{24-31}) \rightarrow$ EBL

CONDITION CODE RESULTS

CC1:	No change
CC2:	No change
CC3:	No change
CC4:	No change

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of bits 24-31 of GPR1 are transferred to memory byte 003A13.

Memory Location:	03708
Hexadecimal Instruction:	D48E3A00 (R=1, X=0, BR=6)
Assembly Language Coding:	STB 1,X'3A00'(6)

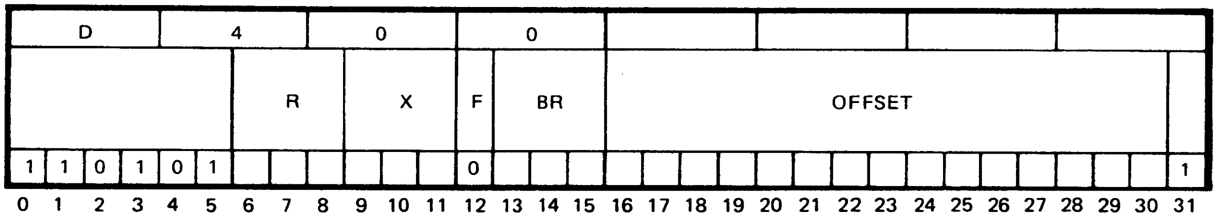
Before	PSD1	GPR1	BR6	Memory Byte 003A13
	12003708	01020304	00000013	78
After	PSD1	GPR1	BR6	Memory Byte 003A13
	1200370C	01020304	00000013	04

NONBASE REGISTER MODE EXAMPLE

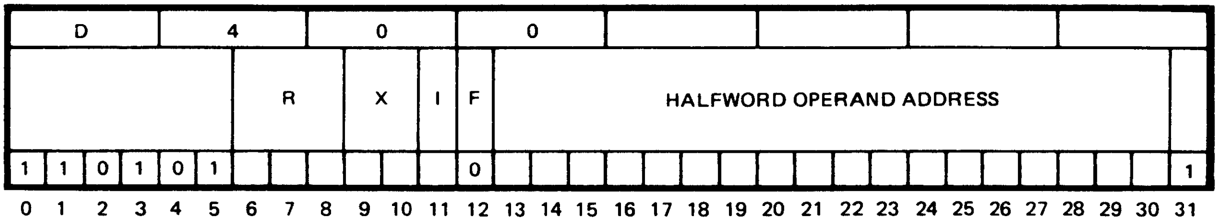
The contents of bits 24-31 of GPR1 are transferred to memory byte 03A13.

Memory Location:	03708
Hexadecimal Instruction:	D4 88 3A 13 (R=1, X=0, I=0)
Assembly Language Coding:	STB 1,X'3A13'

Before	PSD1	GPR1	Memory Byte 03A13
	10003708	01020304	78
After	PSD1	GPR1	Memory Byte 03A13
	1000370C	01020304	04



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

DEFINITION

The least-significant halfword (bits 16-31) of the general purpose register (GPR) specified by R is transferred to the effective halfword location (EHL) specified by the effective halfword address (EHA) in the instruction word. The other halfword of the memory word containing the halfword specified by the EHA remains unchanged.

SUMMARY EXPRESSION

$(R_{16-31}) \rightarrow$ EHL

CONDITION CODE RESULTS

CC1:	No change
CC2:	No change
CC3:	No change
CC4:	No change

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of the right halfword of GPR4 are transferred to memory location 008312.

Memory Location:	082A4
Hexadecimal Instruction:	D6068303 (R=4, X=0, BR=6)
Assembly Language Coding:	STH 4,X'8302'(6)

Before	PSD1	GPR4	BR6	Memory Halfword 008312
	020082A4	01020304	00000010	A492
After	PSD1	GPR4	BR6	Memory Halfword 008312
	020082A8	01020304	00000010	0304

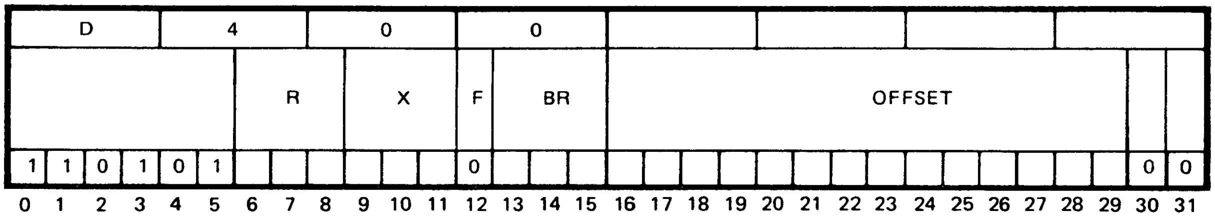
NONBASE REGISTER MODE EXAMPLE

The contents of the right halfword of GPR4 are transferred to memory halfword 08312.

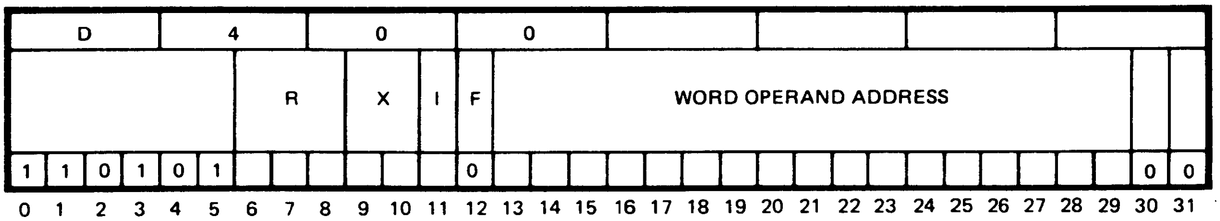
Memory Location:	082A4
Hexadecimal Instruction:	D6 00 83 13 (R=4, X=0, I=0)
Assembly Language Coding:	STH 4,X'8312'

Before	PSD1	GPR4	Memory Halfword 08312
	000082A4	01020304	A492
After	PSD1	GPR4	Memory Halfword 08312
	000082A8	01020304	0304

STORE WORD STW
D400 s,*m,x



BASE REGISTER FORMAT



830154

NONBASE REGISTER FORMAT

DEFINITION

The word in the general purpose register (GPR) specified by R is transferred to the effective word location (EWL) specified by the effective word address (EWA) in the instruction word.

SUMMARY EXPRESSION

(R) → EWL

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of GPR6 are transferred to memory word 003B3C.

Memory Location:	03904
Hexadecimal Instruction:	D7063B00 (R=6, X=0, BR=6)
Assembly Language Coding:	STW 6,X'3B00'(6)

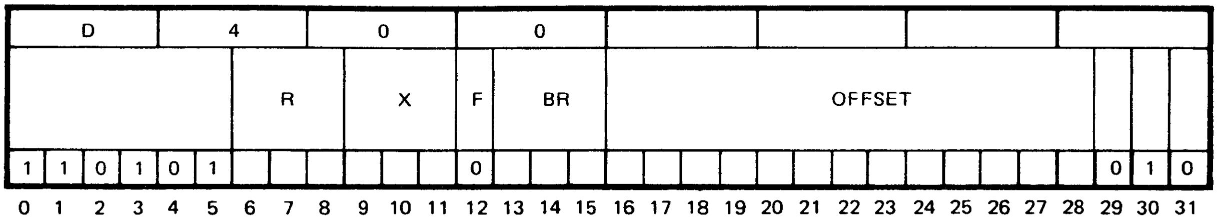
Before	PSD1	GPR6	BR6	Memory Word 003B3C
	12003904	0485A276	0000003C	00000000
After	PSD1	GPR6	BR6	Memory Word 003B3C
	12003908	0485A276	0000003C	0485A276

NONBASE REGISTER MODE EXAMPLE

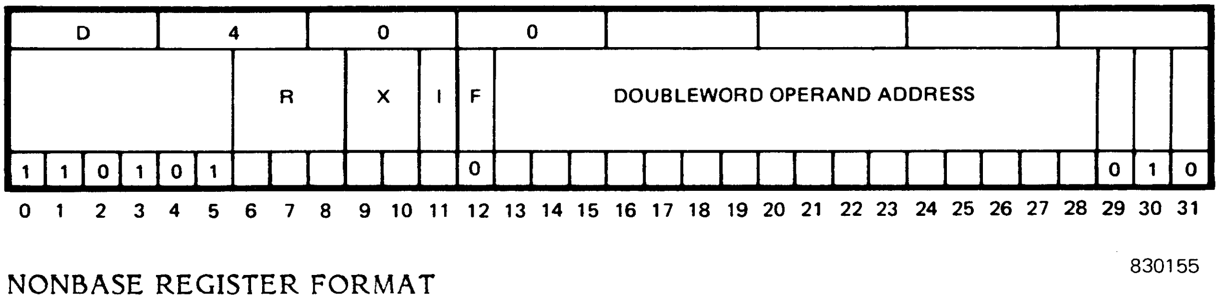
The contents of GPR6 are transferred to memory word 03B3C.

Memory Location:	03904
Hexadecimal Instruction:	D7 00 3B 3C (R=6, X=0, I=0)
Assembly Language Coding:	STW 6,X'3B3C'

Before	PSD1	GPR6	Memory Word 03B3C
	10003904	0485A276	00000000
After	PSD1	GPR6	Memory Word 03B3C
	10003908	0485A276	0485A276



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

DEFINITION

The doubleword in the general purpose register (GPR) specified by R and R+1 (R+1 is the GPR one greater than specified by R) is transferred to the effective doubleword location (EWL) specified by the effective doubleword address (EDA). The word in the GPR specified by R+1 is transferred to the least-significant word of the doubleword memory location first.

NOTE

The GPR specified by R must be an even-numbered register.

SUMMARY EXPRESSION

(R+1) → EWL+1

(R) → EWL

CONDITION CODE RESULTS

CC1:	No change
CC2:	No change
CC3:	No change
CC4:	No change

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of GPR7 are transferred to memory word 065C4C; the contents of GPR6 are transferred to memory word 065C48.

Memory Location:	0596C
Hexadecimal Instruction:	D7065C4A (R=6, X=0, BR=6)
Assembly Language Coding:	STD 6,X'5C48'(6)

Before	PSD1	GPR6	GPR7	BR6
	2200596C	E24675C2	5923F8E8	00060000
	Memory Word 065C48		Memory Word 065C4C	
	0A400729		8104A253	
After	PSD1	GPR6	GPR7	BR6
	22005970	E24675C2	5923F8E8	00060000
	Memory Word 065C48		Memory Word 065C4C	
	E24675C2		5923F8E8	

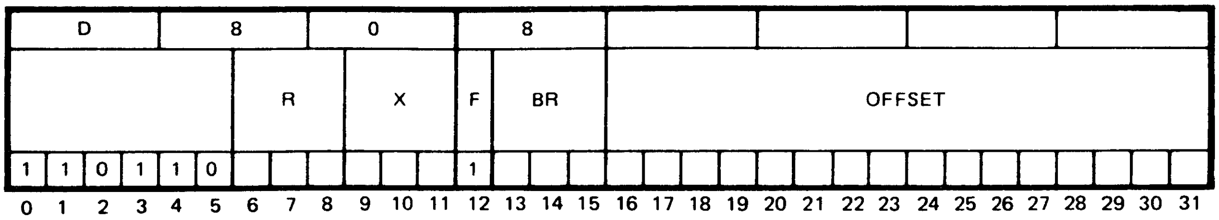
NONBASE REGISTER MODE EXAMPLE

The contents of GPR7 are transferred to memory word 05C4C; the contents of GPR6 are transferred to memory word 05C48.

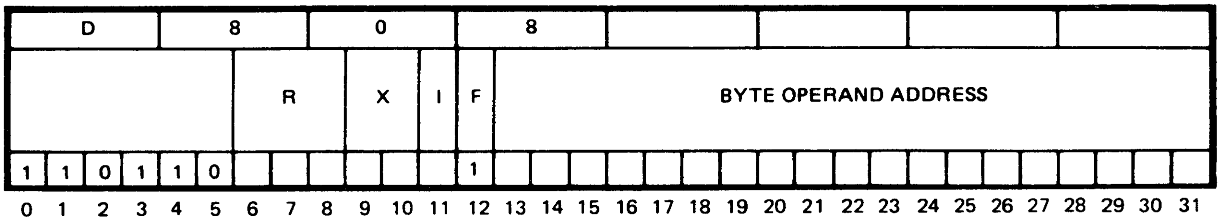
Memory Location:	0596C
Hexadecimal Instruction:	D7 00 5C 4A (R=6, X=0, I=0)
Assembly Language Coding:	STD 6,X'5C48'

Before	PSD1	GPR6	GPR7
	2000596C	E24675C2	5923F8E8
	Memory Word 05C48		Memory Word 05C4C
	0A400729		8104A253
After	PSD1	GPR6	GPR7
	20005970	E24675C2	5923F8E8
	Memory Word 05C48		Memory Word 05C4C
	E24675C2		5923F8E8

STORE MASKED BYTE STMB
D808 s,*m,x



BASE REGISTER FORMAT



830156

NONBASE REGISTER FORMAT

DEFINITION

The least-significant byte (bits 24-31) of the general purpose register (GPR) specified by R is masked (logical AND function) with the least-significant byte of the mask register (R4). The resulting byte is transferred to the effective byte location (EBL) specified by the effective byte address (EBA) in the instruction word. The other three bytes of the memory word containing the byte specified by the EBA remain unchanged.

SUMMARY EXPRESSION

$(R_{24-31})\&(R4_{24-31}) \rightarrow$ EBL

CONDITION CODE RESULTS

CC1:	No change
CC2:	No change
CC3:	No change
CC4:	No change

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The least-significant byte of GPR0 is ANDed with the least-signifcant byte of GPR4. The result is transferred to memory byte 001E91.

Memory Location:	01D80
Hexadecimal Instruction:	D80E1E00 (R=0, X=0, BR=6)
Assembly Language Coding:	STMB 0,X'1E00'(6)

Before	PSD1	GPR0	GPR4	BR6	Memory Byte 001E91
	12001D80	AC089417	0000FFFC	00000091	12943456
After	PSD1	GPR0	GPR4	BR6	Memory Byte 001E91
	12001D84	AC089417	0000FFFC	00000091	12143456

NONBASE REGISTER MODE EXAMPLE

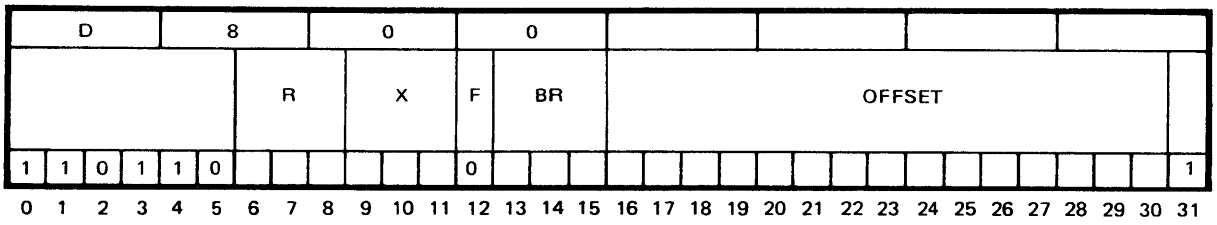
The least-significant byte of GPR0 is ANDed with the least-significant byte of GPR4. The result is transferred to memory byte 01E91.

Memory Location:	01D80
Hexadecimal Instruction:	D8 08 1E 91 (R=0, X=0, I=0)
Assembly Language Coding:	STMB 0,X'1E91'

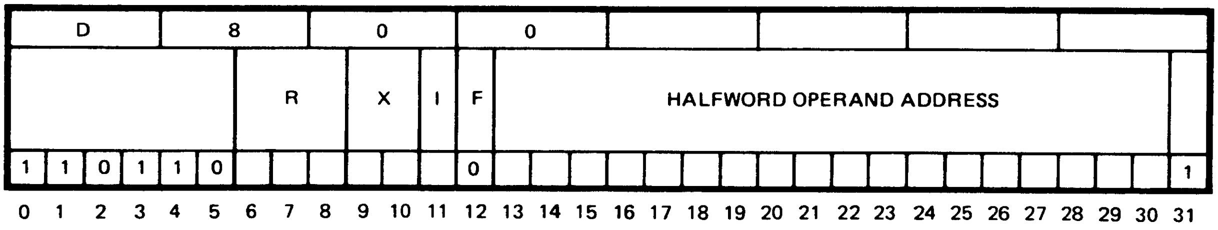
Before	PSD1	GPR0	GPR4	Memory Byte 01E91
	10001D80	AC089417	0000FFFC	12943456
After	PSD1	GPR0	GPR4	Memory Byte 01E91
	10001D84	AC089417	0000FFFC	12143456

STORE MASKED HALFWORD D800

STMH s,*m,x



BASE REGISTER FORMAT



830157

NONBASE REGISTER FORMAT

DEFINITION

The least-significant halfword (bits 16-31) of the general purpose register (GPR) specified by R is masked (logical AND function) with the least-significant halfword of the mask register (R4). The resulting halfword is transferred to the effective halfword location (EHL) specified by the effective halfword address (EHA) in the instruction word. The other halfword of the memory word containing the halfword specified by the EHA remains unchanged.

SUMMARY EXPRESSION

$$(R_{16-31})\&(R4_{16-31}) \rightarrow \text{EHL}$$

CONDITION CODE RESULTS

CC1:	No change
CC2:	No change
CC3:	No change
CC4:	No change

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The least-significant halfword of GPR5 is ANDed with the least-significant halfword of GPR4, and the result is transferred to memory halfword 0011A2.

Memory Location:	01000
Hexadecimal Instruction:	DA861103 (R=5, X=0, BR=6)
Assembly Language Coding:	STMH 5,X'1102'(6)

Before	PSD1	GPR4	GPR5	BR6
	22001000	00003FFC	716A58AB	000000A0
	Memory Halfword 0011A2			
	0000			
After	PSD1	GPR4	GPR5	BR6
	22001004	00003FFC	716A58AB	000000A0
	Memory Halfword 001A2			
	18A8			

NONBASE REGISTER MODE EXAMPLE

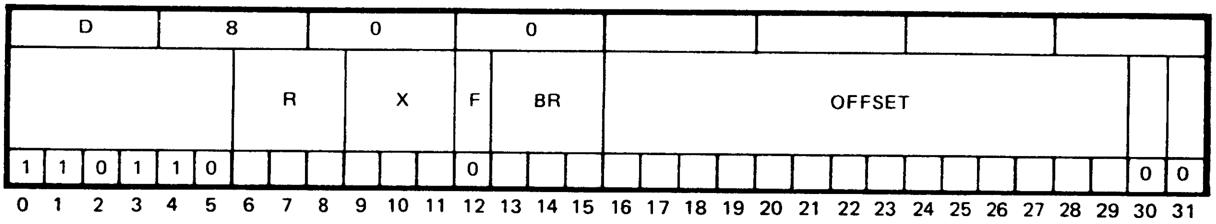
The least-significant halfword of GPR5 is ANDed with the least-significant halfword of GPR4, and the result is transferred to memory halfword 011AD.

Memory Location:	01000
Hexadecimal Instruction:	DA 80 11 AF (R=5, X=0, I=0)
Assembly Language Coding:	STMH 5,X'11AE'

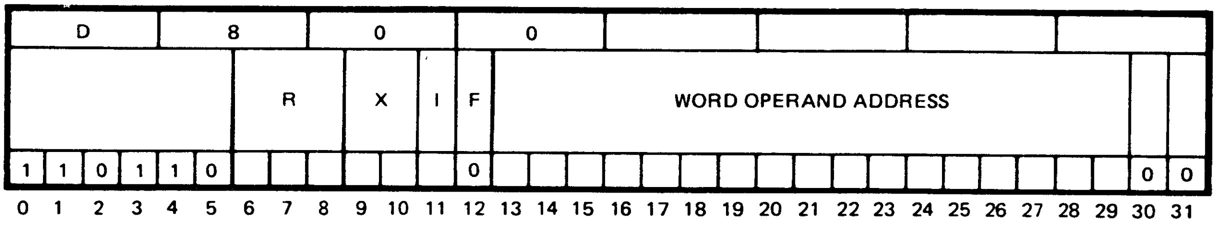
Before	PSD1	GPR4	GPR5	Memory Halfword 011AE
	20001000	00003FFC	716A58AB	0000
After	PSD1	GPR4	GPR5	Memory Halfword 011AE
	20001004	00003FFC	716A58AB	18A8

STORE MASKED WORD
D800

STMW
s,*m,x



BASE REGISTER FORMAT



830158

NONBASE REGISTER FORMAT

DEFINITION

The word in the general purpose register (GPR) specified by R is masked (logical AND function) with the contents of the mask register (R4). The resulting word is transferred to the effective word location (EWL) specified by the effective word address.

SUMMARY EXPRESSION

(R)&(R4) → EWL

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of GPR6 are ANDed with the contents of GPR4. The result is transferred to memory word 00437C.

Memory Location:	04000
Hexadecimal Instruction:	DB064370 (R=6, X=0, BR=6)
Assembly Language Coding:	STMW 6,X'4370'(6)

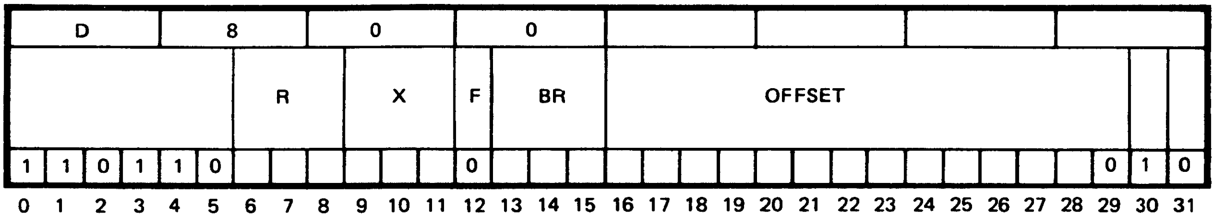
Before	PSD1	GPR4	GPR6	BR6	Memory Word 00437C
	0A004000	00FF00FF	718C3594	0000000C	12345678
After	PSD1	GPR4	GPR6	BR6	Memory Word 00437C
	0A004004	00FF00FF	718C3594	0000000C	008C0094

NONBASE REGISTER MODE EXAMPLE

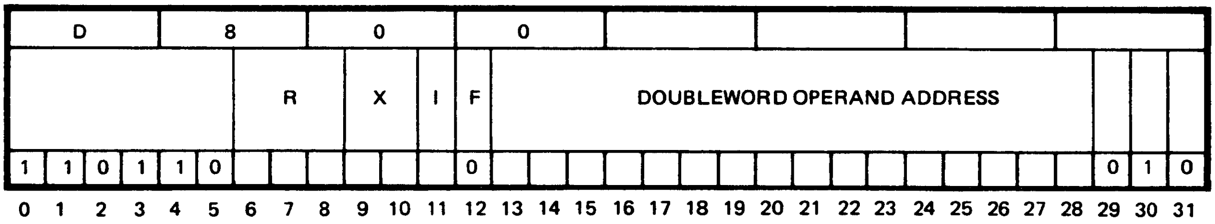
The contents of GPR6 are ANDed with the contents of GPR4. The result is transferred to memory word 0437C.

Memory Location:	04000
Hexadecimal Instruction:	DB 00 43 7C (R=6, X=0, I=0)
Assembly Language Coding:	STMW 6,X'437C'

Before	PSD1	GPR4	GPR6	Memory Word 0437C
	08004000	00FF00FF	718C3584	12345678
After	PSD1	GPR4	GPR6	Memory Word 0437C
	08004004	00FF00FF	718C3594	008C0094



BASE REGISTER FORMAT



830159

NONBASE REGISTER FORMAT

DEFINITION

Each word of the doubleword in the general purpose register (GPR) specified by R and R+1 is masked (logical AND function) with the contents of the mask register (R4). R+1 is the GPR one greater than specified by R. The resulting doubleword is transferred to the effective word location (EWL) specified by the effective doubleword address (EDA) in the instruction word. The least-significant EWL (EWL+1) from R+1 is transferred first.

NOTE

The GPR specified by R must be an even-numbered register.

SUMMARY EXPRESSION

(R+1)&(R4) → EWL+1

(R)&(R4) → EWL

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of GPR7 are ANDed with the contents of GPR4, and the result is transferred to memory word 00A654. The contents of GPR6 are ANDed with the contents of GPR4, and the result is transferred to memory word 00A650.

Memory Location:	0A498
Hexadecimal Instruction:	DB06A052 (R=6, X=0, BR=6)
Assembly Language Coding:	STMD 6,X'A050'(6)

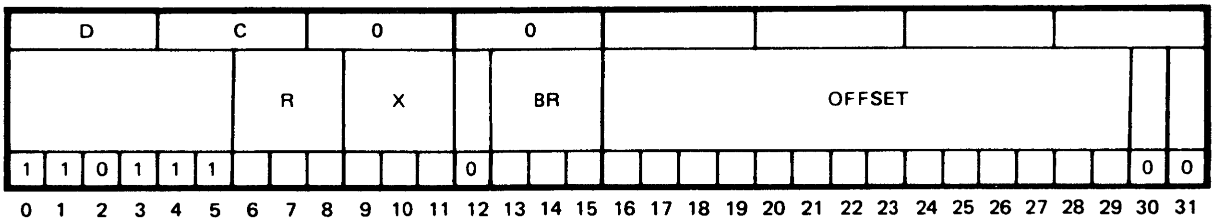
Before	PSD1	GPR4	GPR6	GPR7	BR6
	1200A498	0007FFFC	AC88A819	988B1407	00000600
	Memory Word 00A650		Memory Word 00A654		
	51CD0923		AE69D10C		
After	PSD1	GPR4	GPR6	GPR7	BR6
	1200A49C	0007FFFC	AC88A819	988B1407	00000600
	Memory Word 00A650		Memory Word 00A654		
	0000A818		00031404		

NONBASE REGISTER MODE EXAMPLE

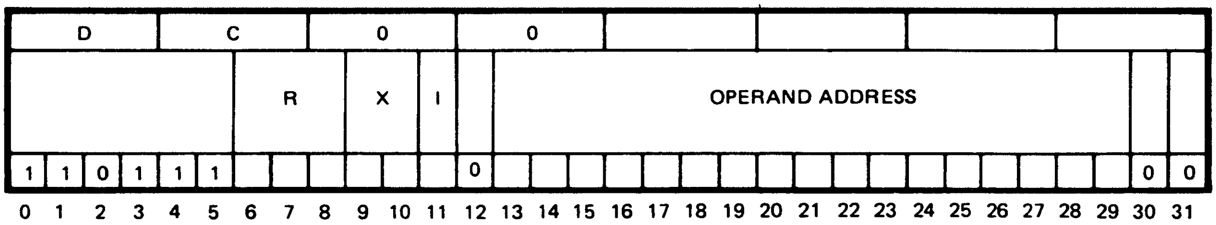
The contents of GPR7 are ANDed with the contents of GPR4, and transferred to memory word 00A654. The contents of GPR6 are ANDed with the contents of GPR4, and the result transferred to memory word 00A650.

Memory Location:	0A498
Hexadecimal Instruction:	DB 00 A6 52 (R=6, X=0, I=0)
Assembly Language Coding:	STMD 6,X'A650'

Before	PSD1		GPR4	GPR6	GPR7
	1000A498	0007FFFC	AC88A819	988B1407	
	Memory Word 0A650		Memory Word 0A654		
	51CD0923		AE69D10C		
After	PSD1		GPR4	GPR6	GPR7
	1000A49C	0007FFFC	AC88A819	988B1407	
	Memory Word 0A650		Memory Word 0A654		
	0000A818		00031404		



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

DEFINITION

This instruction transfers the contents of one to eight general purpose registers (GPR) to specified word locations. The contents of the GPR specified by R are transferred to the effective word location (EWL) specified by the effective word address (EWA). The next sequential GPR is then transferred to the next sequential word location. Successive transfers continue until GPR7 is stored into memory.

NOTES

1. If the F and C bits are changed during indexing or indirection, such final address specified is not a word address, and an address specification trap will occur.
2. If an operand address generated by this instruction crosses a MAP block boundary, an address specification trap will occur. This applies when in either the mapped or unmapped mode.

SUMMARY EXPRESSION

(R) → EWL

(R+1) → EWL+1

.   .
.   .
.   .

(R7) → EWL+n

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of GPR4 are transferred to memory word 002120, of GPR5 to 002124, of GPR6 to 002128, and of GPR7 to 00212C.

Memory Location:	02000
Hexadecimal Instruction:	DE062100 (R=4, X=0, BR=6)
Assembly Language Coding:	STF 4,X'2100'(6)

Before	PSD1	GPR4	GPR5	GPR6	GPR7	BR6
	42002000	11111111	22222222	33333333	44444444	00000020

Memory Word 002120	Memory Word 002124
00210000	00210C00
Memory Word 002128	**Memory Word 00212C**
00210800	00210C00

After	PSD1	GPR4	GPR5	GPR6	GPR7	BR6
	42002004	11111111	22222222	33333333	44444444	00000020

Memory Word 002120	Memory Word 002124
11111111	22222222
Memory Word 002128	Memory Word 00212C
33333333	44444444

NONBASE REGISTER MODE EXAMPLE

The contents of GPR4 are transferred to memory word 02100, of GPR5 to 02104, of GPR6 to 02108, and of GPR7 to 0210C.

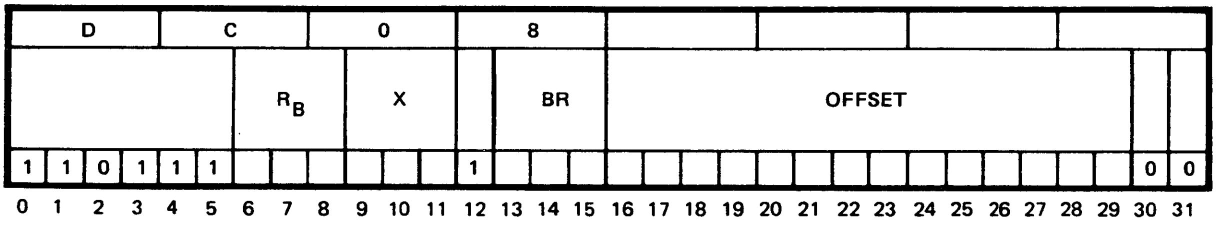
Memory Location:	02000
Hexadecimal Instruction:	DE 00 21 00 (R=4, X=0, I=0)
Assembly Language Coding:	STF 4,X'2100'

Before	PSD1	GPR4	GPR5	GPR6	GPR7
	40002000	11111111	222222222	33333333	44444444

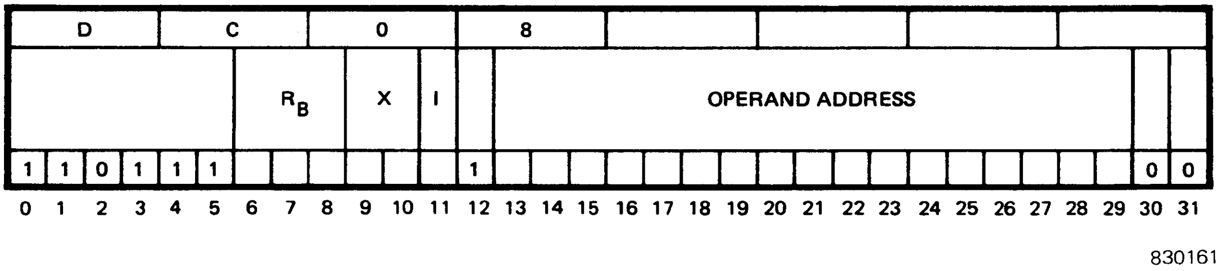
Memory Word 02100	Memory Word 02104
00210000	0210400
Memory Word 02108	Memory Word 0210C
00210800	00210C00

After	PSD1	GPR4	GPR5	GPR6	GPR7
	40002004	11111111	22222222	33333333	44444444

Memory Word 02100	Memory Word 02104
11111111	22222222
Memory Word 02108	Memory Word 0210C
33333333	44444444



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

DEFINITION

This instruction is used to transfer the contents of one to eight base register (R_B) to specified word locations. The contents of the base register specified by R_B are transferred to the effective word location (EWL) specified by the effective word address (EWA). The next sequential base register is then transferred to the next sequential word location. Successive transfers continue until R_B7 is stored into memory.

SUMMARY EXPRESSION

(R_B) → – EWL
(R_B+1) → EWL+1
. .
. .
. .
(R_B7) → EWL+n

CONDITION CODE RESULTS

CC1: Unchanged
CC2: Unchanged
CC3: Unchanged
CC4: Unchanged

NOTES

1. This instruction may be executed in either Base Register Mode or Nonbase Register mode.

2. If the effective word address (EWA) bits 30 and 31 are not zero, an address specification will occur.

3. If an operand address generated by this instruction crosses a MAP block boundary, an address specification trap will occur. This applies when in either the mapped or unmapped mode.

BASE REGISTER MODE EXAMPLE

The contents of BR3 and the instruction offset are added to obtain the logical address. The contents of R_B6 are transferred to memory word 000220, and the contents of R_B7 are transferred to memory word 000224.

Memory Location:	02000
Hexadecimal Instruction:	DF0B0200 (R_B=6, X=0, BR=3)
Assembly Language Coding:	STFBR 6,X'200'(3)

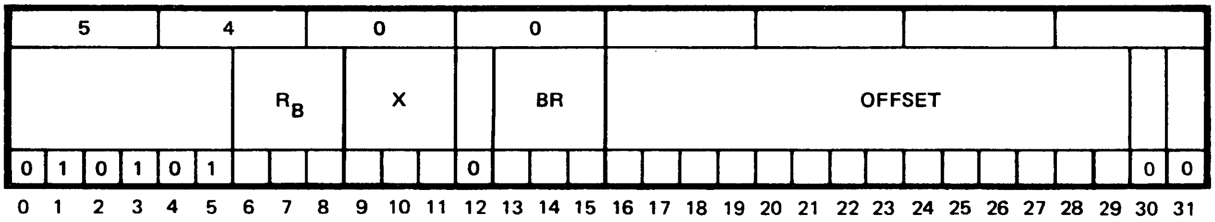
Before	PSD1	R_B6	R_B7	BR3
	02002000	11111111	22222222	00000020
	Memory Word 000220		Memory Word 000224	
	0021C000		00220000	
After	PSD1	R_B6	R_B7	BR3
	02002004	11111111	22222222	00000020
	Memory Word 000220		Memory Word 000224	
	11111111		22222222	

NONBASE REGISTER MODE EXAMPLE

The contents of R_B6 are transferred to memory word 0200, and the contents of R_B7 to memory word 0204.

Memory Location:	02000
Hexadecimal Instruction:	DF080200 (R_B6)
Assembly Language Coding:	STFBR 6,X'200'

Before	PSD1	R_B6	R_B7	Memory Word 0200
	00002000	11111111	22222222	0021C00
	Memory Word 0204			
	00220000			
After	PSD1	R_B6	R_B7	Memory Word 0204
	00002004	11111111	22222222	11111111
	Memory Word 0204			
	22222222			



830162

DEFINITION

The contents of the base register specified by R_B are stored in the effective word location (EWL) specified by the effective word address (EWA).

SUMMARY EXPRESSION

$(R_B) \rightarrow \text{EWL}$

CONDITION CODE RESULTS

CC1: Unchanged
CC2: Unchanged
CC3: Unchanged
CC4: Unchanged

NOTES

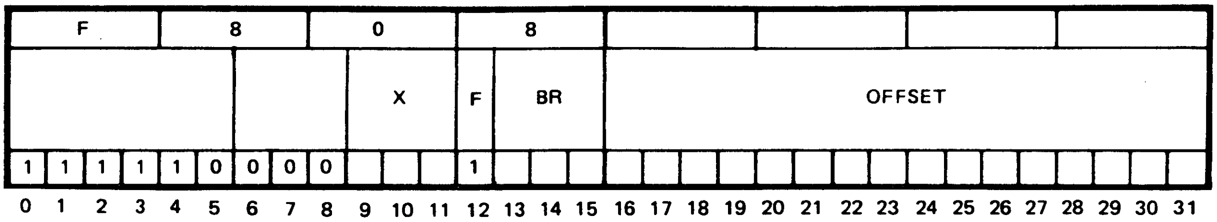
1. This instruction is used for the base register mode only.
2. If the effective word address (EWA) bits 30 and 31 are not zero, an address specification trap will occur.

BASE REGISTER MODE EXAMPLE

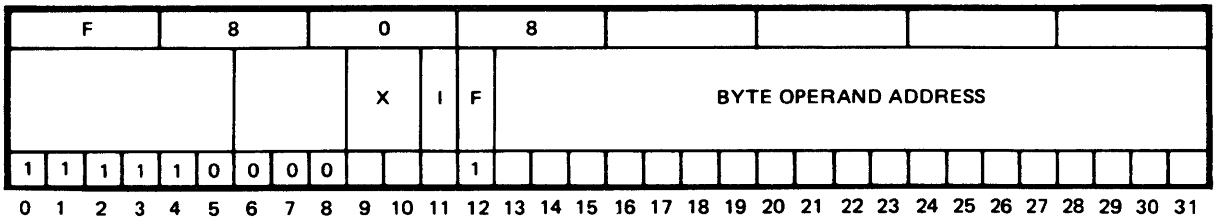
The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of R_B3 are transferred to memory word 000304.

Memory Location:	02004
Hexadecimal Instruction:	55860300 (R_B=3, X=0, BR=6)
Assembly Language Coding:	STWBR 3,X'300'(6)

Before	PSD1	R_B3	R_B6	Memory Word 000304
	02002004	22222222	00000004	00210004
After	PSD1	R_B3	R_B6	Memory Word 000304
	02002008	22222222	00000004	22222222



BASE REGISTER FORMAT



830163

NONBASE REGISTER FORMAT

DEFINITION

The effective byte location (EBL) in memory specified by the effective byte address (EBA) is reset to zero. The other three bytes of the memory word containing the byte specified by the EBA remain unchanged.

SUMMARY EXPRESSION

Zeros → EBL

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of memory byte 00049F are reset to zero.

Memory Location:	00308
Hexadecimal Instruction:	F80E0400 (X=0, BR=6)
Assembly Language Coding:	ZMB X'0400'(6)

Before	PSD1	BR6	Memory Byte 00049F	
		12000308	0000009F	6C
After	PSD1	BR6	Memory Byte 00049F	
		1200030C	0000009F	00

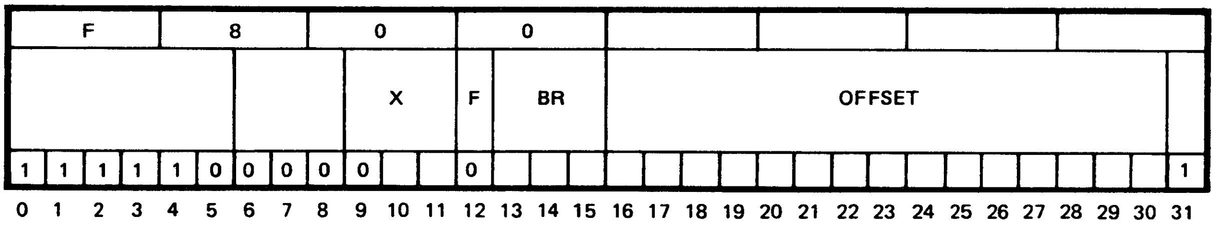
NONBASE REGISTER MODE EXAMPLE

The contents of memory byte 0049F are reset to zero.

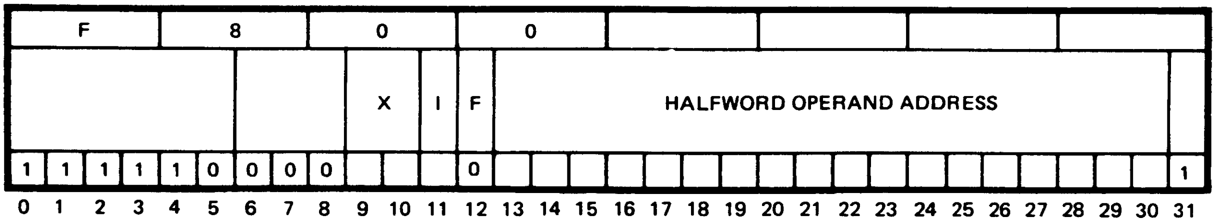
Memory Location:	00308
Hexadecimal Instruction:	F8 08 04 9F (X=0, I=0)
Assembly Language Coding:	ZMB X'49F'

Before	PSD1	Memory Byte 0049F
	10000308	6C
After	PSD1	Memory Byte 0049F
	1000030C	00

ZERO MEMORY HALFWORD — ZMH
F800 — *m,x



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

830164

DEFINITION

The effective halfword location (EHL) specified by the effective halfword address (EHA) is reset to zero. The remaining halfword containing the 16-bit location in memory specified by EHA remains unchanged.

SUMMARY EXPRESSION

Zeros → EHL

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of memory halfword 00A422 are reset to zero.

Memory Location:	2895C
Hexadecimal Instruction:	F806A403 (X=0, BR=6)
Assembly Language Coding:	ZMH X'A402' (6)

Before	PSD1	BR6	Memory Halfword 00A422
	0A02895C	00000020	9AE3
After	PSD1	BR6	Memory Halfword 00A422
	0A028960	00000020	0000

NONBASE REGISTER MODE EXAMPLE

The contents of memory halfword 2A426 are reset to zero.

Memory Location:	2895C
Hexadecimal Instruction:	F8 00 2A 42 7 (X=0, I=0)
Assembly Language Coding:	ZMH X'2A426'

Before	PSD1	Memory Halfword 2A426
	0802895C	9AE3
After	PSD1	Memory Halfword 2A426
	08028960	0000

F				8				0				0																			
									X			F	BR			OFFSET															
1	1	1	1	1	0	0	0	0				0																		0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

BASE REGISTER FORMAT

F				8				0				0																			
									X		I	F	WORD OPERAND ADDRESS																		
1	1	1	1	1	0	0	0	0				0																		0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

830165

NONBASE REGISTER FORMAT

DEFINITION

The effective word location (EWL) in memory specified by the effective word address (EWA) is reset to zero.

SUMMARY EXPRESSION

Zeros → EWL

CONDITION CODE RESULTS

CC1:	No change
CC2:	No change
CC3:	No change
CC4:	No change

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of memory word 005F90 are reset to zero.

Memory Location:	05A14
Hexadecimal Instruction:	F8065F00 (X=0, BR=6)
Assembly Languge Coding:	ZMW X'5F00' (6)

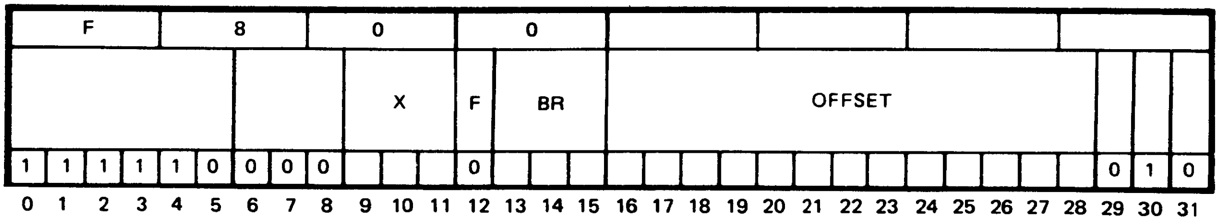
Before	PSD1	BR6	Memory Word 005F90
	02005A14	00000090	12345678
After	PSD1	BR6	Memory Word 005F90
	02005A18	00000090	00000000

NONBASE REGISTER MODE EXAMPLE

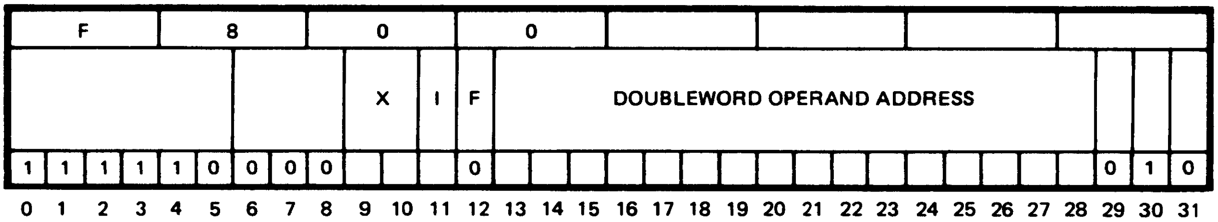
The contents of memory word 05F90 are reset to zero.

Memory Location:	05A14
Hexadecimal Instruction:	F8 00 5F 90 (X=0, I=0)
Assembly Language Coding:	ZMW X'5F90'

Before	PSD1	Memory Word 05F90
	00005A14	12345678
After	PSD1	Memory Word 05F90
	00005A18	00000000



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

830166

DEFINITION

The effective doubleword location (EDL) in memory specified by the effective doubleword address (EDA) is reset to zero. The least-significant effective word location (EWL) is reset to zero first.

SUMMARY EXPRESSION

Zeros → EWL+1

Zeros → EWL

CONDITION CODE RESULTS

CC1:	No change
CC2:	No change
CC3:	No change
CC4:	No change

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of memory words 005D6C and 005D68 are reset to zero.

Memory Location:	15B3C
Hexadecimal Instruction:	F806506A (X=0 , BR=6)
Assembly Language Coding:	ZMD X'5068'(6)

Before	PSD1 12015B3C	BR6 00000D00	Memory Word 005D68 617E853C
	Memory Word 005D6C A2976283		
After	PSD1 12015B40	BR6 00000D00	Memory Word 005D68 00000000
	Memory Word 005D6C 00000000		

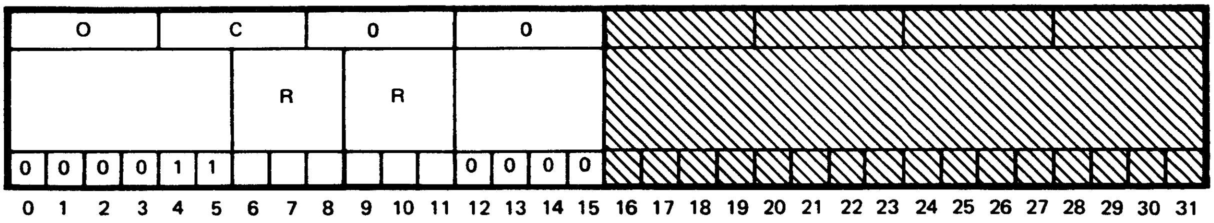
NONBASE REGISTER MODE EXAMPLE

The contents of memory words 15D6C and 15D68 are reset to zero.

Memory Location:	15B3C
Hexadecimal Instruction:	F8 01 5D 6A (X=0, I=0)
Assembly Language Coding:	ZMD X'15D68'

Before	PSD1 10015B3C	Memory Word 15D68 617E853C	Memory Word 15D6C A2976283
After	PSD1 10015B40	Memory Word 15D68 00000000	Memory Word 15D6C 00000000

ZERO REGISTER ZR
OC00 d



830167

DEFINITION

The two R fields must specify the same general purpose register (GPR). The word specified by R (bits 6-8) is logically exclusive ORed with the word specified by R (bits 9-11) resulting in zero. This result is then transferred to the GPR specified by R.

NOTE

> ZR is an exclusive OR, register to register instruction with the source and destination registers being the same register.

SUMMARY EXPRESSION

$(R) \oplus (R) \rightarrow R$

CONDITION CODE RESULTS

CC1:	Always zero
CC2:	Always zero
CC3:	Always zero
CC4:	Always set

NONBASE AND BASE REGISTER MODE EXAMPLE

The contents of GPR1 are cleared to zero. CC4 is set.

Memory Location:	309A6
Hexadecimal Instruction:	0C90 (R=1)
Assembly Language Coding:	ZR 1

Before	PSD1 100309A6 (Nonbase) 120309A6 (Base)	GPR1 8495A6B7
After	PSD1 080309A8 (Nonbase) 0A0309A8 (Base)	GPR1 00000000

This page intentionally left blank

6.2.2 Register Transfer Instructions

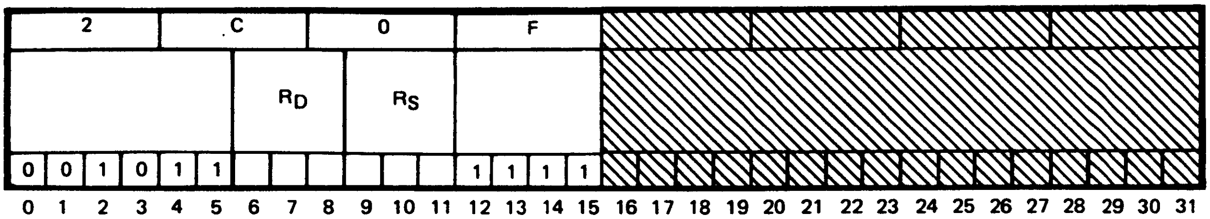
The register transfer instruction group provides the capability to transfer or exchange information between registers. Provisions have been made in some instructions to allow twos complement, ones complement, and mask operations to be performed during execution.

6.2.2.1 Instruction Format

The register transfer instructions use the standard interregister format.

6.2.2.2 Condition Code

A condition code is set during execution of most register transfer instructions to indicate whether the contents of the destination register (R_D) are greater than, less than, or equal to zero.



830168

DEFINITION

The word in the scratchpad location specified by the contents of R_S, bits 8-15, is transferred to the general purpose register (GPR) specified by R_D. The contents of R_S are not modified and only bits 8-15 are used by the instruction.

SUMMARY EXPRESSION

(Scratchpad addressed by $R_{S\ 8-15}$) → R_D

CONDITION CODE RESULTS

CC1:	No change
CC2:	No change
CC3:	No change
CC4:	No change

NOTES

1. TSCR is a privileged instruction.

2. The valid address range for R_S to address the 256 scratchpad locations is $XX00XXXX_{16}$ to $XXFFXXXX_{16}$.

NONBASE AND BASE REGISTER MODE EXAMPLE

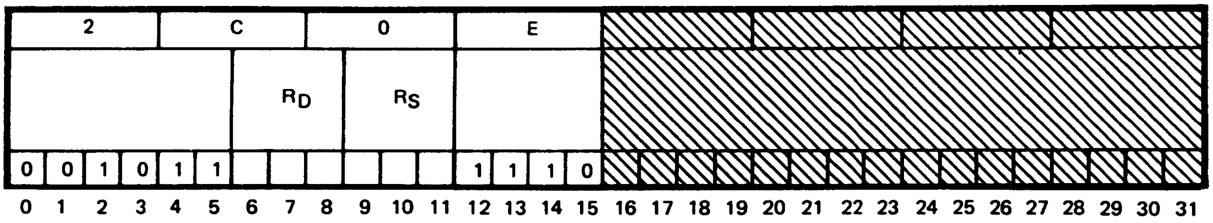
The contents of the scratchpad location specified by R_S, bits 8-15, is transferred to general purpose register (GPR) specified by R_D.

Memory Location:	40D68
Hexadecimal Instruction:	2E CF (R_S=4, R_D=5)
Assembly Language Coding:	TSCR 4,5

Before	Scratchpad Location X'3F' 0034789A	PSD1 A0040D68 (Nonbase) A2040D68 (Base)	GPR4 003F0000	GPR5 12340000
After	Scratchpad Location X'3F' 0034789A	PSD1 A0040D6A (Nonbase) A2040D6A (Base)	GPR4 003F0000	GPR5 0034789A

TRANSFER REGISTER TO SCRATCHPAD
 2C0E

TRSC
s,d



830169

DEFINITION

The word located in the general purpose register (GPR) specified by R_S is transferred to the scratchpad location specified by R_D bits 8-15. The contents of R_D are not modified by the instruction and only bits 8-15 are used by the instruction.

SUMMARY EXPRESSION

(R_S) → Scratchpad addressed by R_D 8-15

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

NOTES

1. TRSC is a privileged instruction.

2. The valid address range for R_D to address the 256 scratchpad locations is $XX00XXXX_{16}$ to $XXFFXXXX_{16}$.

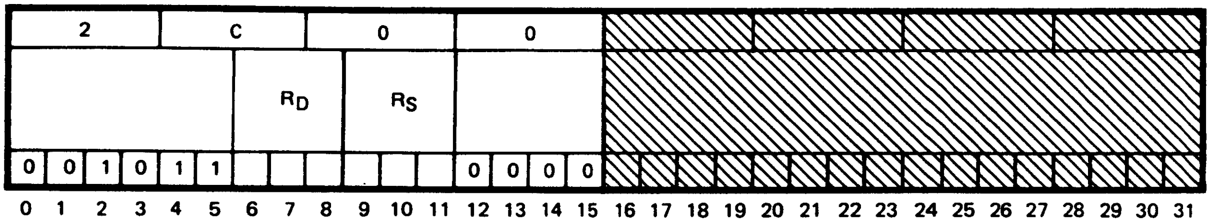
NONBASE AND BASE REGISTER MODE EXAMPLE

The contents of GPR3 is transferred to scratchpad location specified by GPR5, bits 8-15.

Memory Location:	01000
Hexadecimal Instruction:	2E BE (R_S=3 R_D=5)
Assembly Language Coding:	TRSC 3,5

Before	Scratchpad	PSD1	GPR3	GPR5
	Location X'10'	A0001000 (Nonbase)	AAAA5555	00100000
	11112222	A2001000 (Base)		
After	Scratchpad	PSD1	GPR3	GPR5
	Location X'10'	A0001002 (Nonbase)	AAAA5555	00100000
	AAAA5555	A2001002 (Base)		

TRANSFER REGISTER TO REGISTER TRR
2C00 s,d



830170

DEFINITION

The contents of the word in the general purpose register (GPR) specified by R_S is transferred to the GPR specified by R_D.

SUMMARY EXPRESSION

$(R_S) \rightarrow R_D$

CONDITION CODE RESULTS

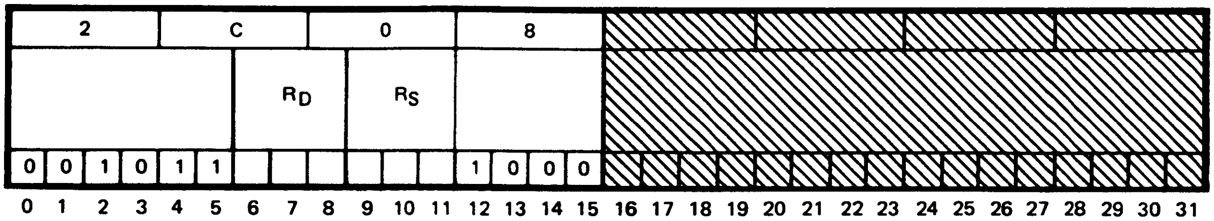
CC1: Always zero
CC2: Is set if (R_D) is greater than zero
CC3: Is set if (R_D) is less than zero
CC4: Is set if (R_D) is equal to zero

NONBASE AND BASE REGISTER MODE EXAMPLE

The contents of GPR2 are transferred to GPR1. CC2 is set.

Memory Location:	00206
Hexadecimal Instruction:	2C A0 (R_D=1, R_S=2)
Assembly Language Coding:	TRR 2,1

Before	PSD1 00000206 (Nonbase) 02000206 (Base)	GPR1 00000000	GPR2 000803AB
After	PSD1 20000209 (Nonbase) 22000209 (Base)	GPR1 000803AB	GPR2 000803AB



830171

DEFINITION

The contents of the word in the general purpose register (GPR) specified by R_S is masked (logical AND function) with the contents of the mask register (R4). The resulting word is transferred to the GPR specified by R_D.

SUMMARY EXPRESSION

$(R_S)\&(R4) \rightarrow R_D$

CONDITION CODE RESULTS

CC1:	Always zero
CC2:	Is set if (R_D) is greater than zero
CC3:	Is set if (R_D) is less than zero
CC4:	Is set if (R_D) is equal to zero

NONBASE AND BASE REGISTER MODE EXAMPLE

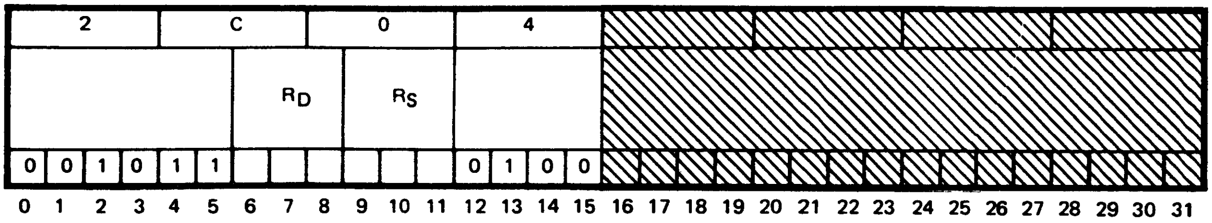
The contents of GPR2 are ANDed with the contents of GPR4, and the result is transferred to GPR1. CC2 is set.

Memory Location:	00206
Hexadecimal Instruction:	2C A8 (R_D=1, R_S=2)
Assembly Language Coding:	TRRM 2,1

Before	PSD1	GPR1	GPR2	GPR4
	00000206 (Nonbase)	00000000	000803AB	0007FFFD
	02000206 (Base)			
After	PSD1	GPR1	GPR2	GPR4
	20000209 (Nonbase)	000003A9	000803AB	0007FFFD
	22000209 (Base)			

TRANSFER REGISTER NEGATIVE 2C04

TRN s,d



830172

DEFINITION

The twos complement of the word in the general purpose register (GPR) specified by R_S is formed and transferred to the GPR specified by R_D.

SUMMARY EXPRESSION

$-(R_S) \rightarrow R_D$

CONDITION CODE RESULTS

CC1: Is set if arithmetic exception
CC2: Is set if (R_D) is greater than zero
CC3: Is set if (R_D) is less than zero
CC4: Is set if (R_D) is equal to zero

NONBASE AND BASE REGISTER MODE EXAMPLE

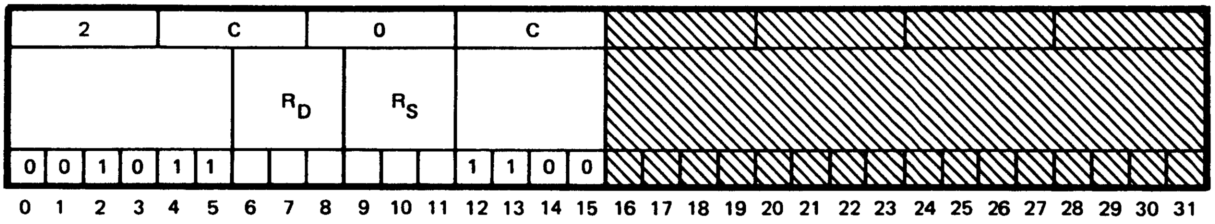
The contents of GPR6 are negated and transferred to GPR7. CC3 is set.

Memory Location:	00AAE
Hexadecimal Instruction:	2F E4 (R_D=7, R_S=6)
Assembly Language Coding:	TRN 6,7

Before	PSD1	GPR6	GPR7
	00000AAE (Nonbase)	00000FFF	12345678
	02000AAE (Base)		
After	PSD1	GPR6	GPR7
	10000AB1 (Nonbase)	00000FFF	FFFFF001
	12000AB1 (Base)		

TRANSFER REGISTER NEGATIVE MASKED 2C0C

TRNM
s,d



830173

DEFINITION

The twos complement of the word in the general purpose register (GPR) specified by R_S is formed and masked (logical AND function) with the contents of the mask register (R4). The resulting word is transferred to the GPR specified by R_D.

SUMMARY EXPRESSION

$-(R_S)\&(R4) \rightarrow R_D$

CONDITION CODE RESULTS

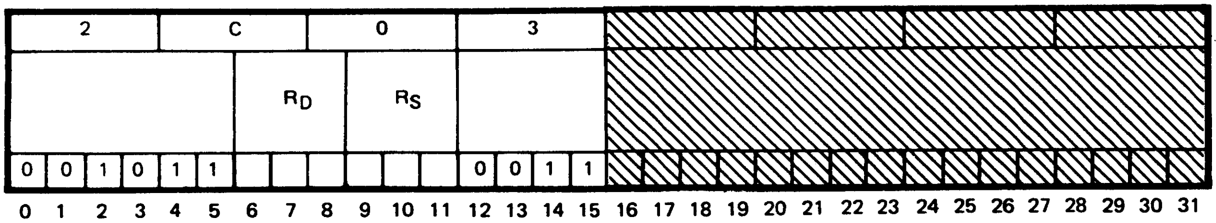
CC1:	Is set if arithmetic exception
CC2:	Is set if (R_D) is greater than zero
CC3:	Is set if (R_D) is less than zero
CC4:	Is set if (R_D) is equal to zero

NONBASE AND BASE REGISTER MODE EXAMPLE

The contents of GPR6 are negated; the result is ANDed with the content of GPR4 and transferred to GPR7. CC2 is set.

Memory Location:	00AAE
Hexadecimal Instruction:	2F EC (R_D=7, R_S=6)
Assembly Language Coding:	TRNM 6,7

Before	PSD1	GPR4	GPR6	GPR7
	00000AAE (Nonbase)	7FFFFFFF	00000FFF	12345678
	02000AAE (Base)			
After	PSD1	GPR4	GPR6	GPR7
	20000AB1 (Nonbase)	7FFFFFFF	00000FFF	7FFFF001
	22000AB1 (Base)			



830174

DEFINITION

The ones complement of the word in the general purpose register (GPR) specified by R_S is formed and transferred to the GPR specified by R_D.

SUMMARY EXPRESSION

$(\bar{R}_S) \rightarrow R_D$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R_D) is greater than zero
CC3: Is set if (R_D) is less than zero
CC4: Is set if (R_D) is equal to zero

TRANSFER REGISTER COMPLEMENT (Cont.)

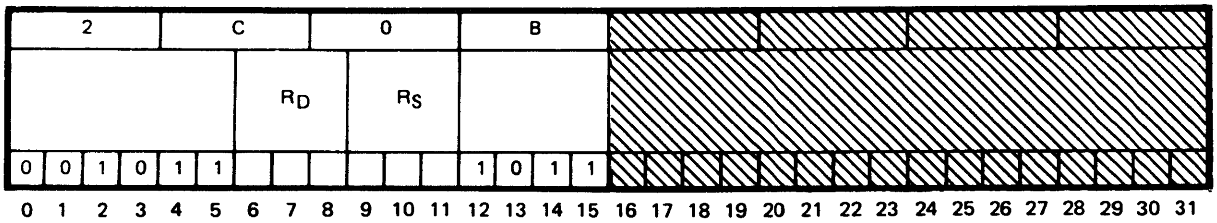
NONBASE AND BASE REGISTER MODE EXAMPLE

The contents of GPR6 are complemented and transferred to GPR7. CC3 is set.

Memory Location:	01001
Hexadecimal Instruction:	2F E3 (R_D=7, R_S=6)
Assembly Language Coding:	TRC 6,7

Before	PSD1	GPR6	GPR7
	0800100A (Nonbase)	55555555	00000000
	0A00100A (Base)		
After	PSD1	GPR6	GPR7
	1000100D (Nonbase)	55555555	AAAAAAAA
	1200100D (Base)		

TRANSFER REGISTER COMPLEMENT MASKED — TRCM
2C0B — s,d



830175

DEFINITION

The ones complement of the word in the general purpose register (GPR) specified by R_S is formed and masked (logical AND function) with the contents of the mask register (R4). The result is transferred to the GPR specified by R_D.

SUMMARY EXPRESSION

$(\overline{R}_S)\&(R4) \rightarrow R_D$

CONDITION CODE RESULTS

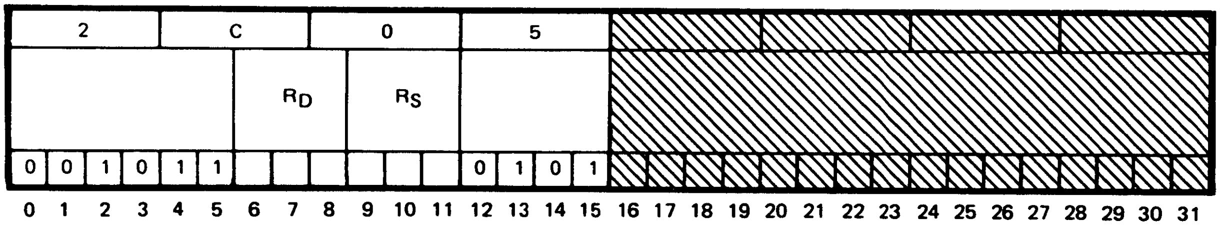
CC1: Always zero
CC2: Is set if (R_D) is greater than zero
CC3: Is set if (R_D) is less than zero
CC4: Is set if (R_D) is equal to zero

NONBASE AND BASE REGISTER MODE EXAMPLE

The contents of GPR6 are complemented and ANDed with the contents of GPR4; the result is transferred to GPR7. CC2 is set.

Memory Location:	0100A
Hexadecimal Instruction:	2F EB (R_D=7, R_S=6)
Assembly Language Coding:	TRCM 6,7

Before	PSD1	GPR4	GPR6	GPR7
	0800100A (Nonbase) 0A00100A (Base)	00FFFF00	55555555	00000000
After	PSD1	GPR4	GPR6	GPR7
	2000100D (Nonbase) 2200100D (Base)	00FFFF00	55555555	00AAAA00



DEFINITION

The contents of the word in the general purpose register (GPR) specified by R_S is exchanged with the contents of the word in the GPR specified by R_D.

SUMMARY EXPRESSION

$(R_S) \rightarrow R_D$

$(R_D) \rightarrow R_S$

CONDITION CODE RESULTS

CC1:	Always zero
CC2:	Is set if original (R_D) is greater than zero
CC3:	Is set if original (R_D) is less than zero
CC4:	Is set if original (R_D) is equal to zero

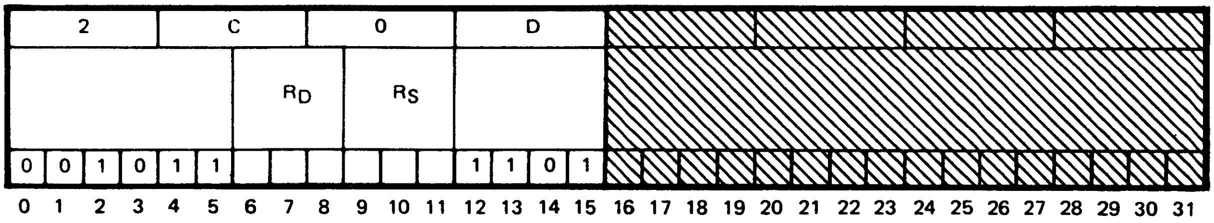
NONBASE AND BASE REGISTER MODE EXAMPLE

The contents of GPR1 and GPR2 are exchanged. CC4 is set.

Memory Location:	02002
Hexadecimal Instruction:	2C A5 (R_D=1, R_S=2)
Assembly Language Coding:	XCR 2,1

Before	PSD1	GPR1	GPR2
	40002002 (Nonbase)	00000000	AC8823C1
	42002002 (Base)		
After	PSD1	GPR1	GPR2
	08002005 (Nonbase)	AC8823C1	00000000
	0A002005 (Base)		

EXCHANGE REGISTERS MASKED XCRM
2C0D s,d



830177

DEFINITION

The contents of the general purpose registers (GPR) specified by R_S and R_D are each masked (logical AND function) with the contents of the mask register (R4). The results of both masked operations are exchanged and replace the contents of R_S and R_D.

SUMMARY EXPRESSION

$(R_S)\&(R4) \rightarrow R_D$

$(R_D)\&(R4) \rightarrow R_S$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if original (R_D) and (R4) is greater than zero
CC3: Is set if original (R_D) and (R4) is less than zero
CC4: Is set if original (R_D) and (R4) is equal to zero

NONBASE AND BASE REGISTER MODE EXAMPLE

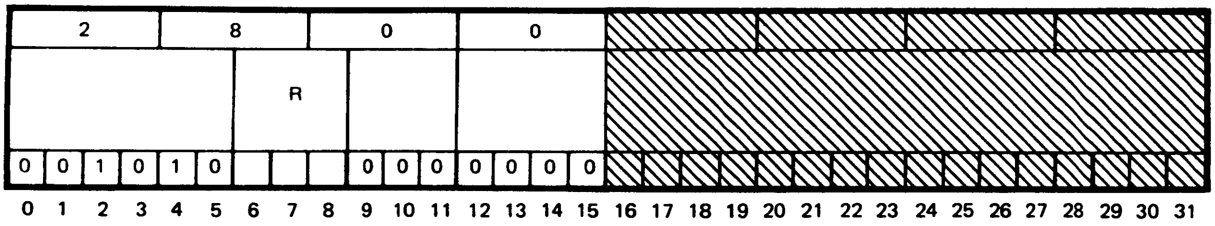
The contents of GPR1 and GPR2 are each ANDed with the contents of GPR4. The results of the masking operation are exchanged and transferred to GPR2 and GPR1, respectively. CC4 is set.

Memory Location:	02002
Hexadecimal Instruction:	2C AD (R_D=1, R_S=2)
Assembly Language Coding:	XCRM 2,1

Before	PSD1	GPR1	GPR2	GPR4
	40002002 (Nonbase)	6B000000	AC8823C1	000FFFFF
	42002002 (Base)			
After	PSD1	GPR1	GPR2	GPR4
	08002005 (Nonbase)	000823C1	00000000	000FFFFF
	0A002005 (Base)			

TRANSFER REGISTER TO PSD 2800

TRSW s



830178

DEFINITION

The contents of bit positions 1-4 and 13-30 (nonbase register mode) or bit position 1-4 and 8-30 (base register mode) of the general purpose register (GPR) specified by R are transferred to the corresponding bit positions, 1-4 and 13-30, in the first word of the program status doubleword (PSD).

SUMMARY EXPRESSION

$(R_{1-4,\ 13-30})$	→	$PSD_{1-4,\ 13-30}$	(Non-base Register)
$(R_{1-4,\ 8-30})$	→	$PSD_{1-4,\ 8-30}$	(Base Register)

CONDITION CODE RESULTS

CC1:	Is set if (R_1) is equal to one
CC2:	Is set if (R_2) is equal to one
CC3:	Is set if (R_3) is equal to one
CC4:	Is set if (R_4) is equal to one

BASE REGISTER MODE EXAMPLE

The contents of GPR0, bits 1-4 and 8-30, are transferred to PSD1, bits 1-30. Bit 0 and bits 5-7 of PSD1 are unchanged.

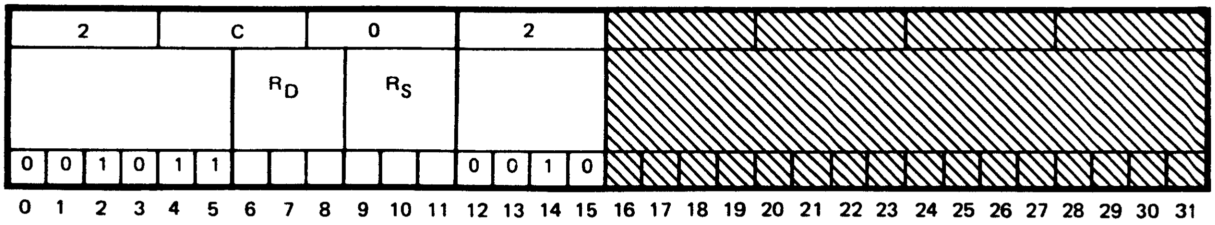
Memory Location:		0069E
Hexadecimal Instruction:		28 00 (R=0)
Assembly Language Coding:		TRSW 0
Before	PSD1 6200069E	GPR0 A0000B4C
After	PSD1 22000B4C	GPR0 A0000B4C

NONBASE REGISTER MODE EXAMPLE

The contents of GPR0, bits 1-4 and 13-30, are transferred to PSD1, bits 1-30. Bit 0 and bits 5-12 of PSD1 are unchanged.

Memory Location:		0069E
Hexadecimal Instruction:		28 00 (R=0)
Assembly Language Coding:		TRSW 0
Before	PSD1 6000069E	GPR0 A0000B4C
After	PSD1 20000B4C	GPR0 A0000B4C

TRANSFER BR TO GPR **TBRR**
2C02 **s,d**



830179

DEFINITION

The word in the base register specified by RS_B is transferred to the general purpose register specified by R_D.

SUMMARY EXPRESSION

$(RS_B) \rightarrow R_D$

CONDITION CODE RESULTS

CC1:	Always zero
CC2:	Is set if (R_D) is greater than zero
CC3:	Is set if (R_D) is less than zero
CC4:	Is set if (R_D) is equal to zero

NOTE

This instruction is used for the base register mode only.

BASE REGISTER MODE EXAMPLE

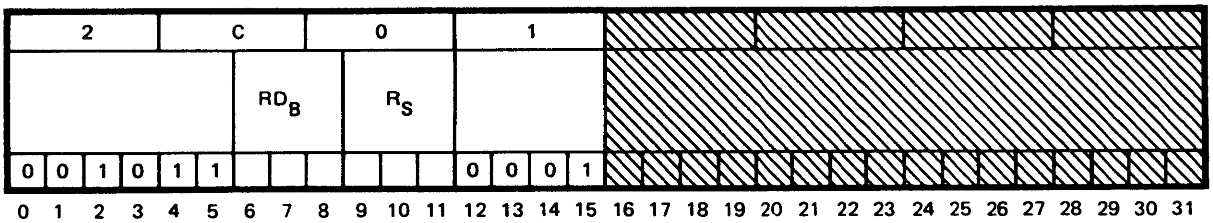
The contents of RS_B4 are transferred to GPR5.

Memory Location:	3008
Hexadecimal Instruction:	2EC2 (R_D=5, RS_B=4)
Assembly Language Coding:	TBRR 4, 5

Before	PSD1 02003008	GPR5 02031678	BR4 03030303
After	PSD1 0200300A	GPR5 03030303	BR4 03030303

TRANSFER GPR TO BR 2C01

TRBR s,d



830180

DEFINITION

The word in the general purpose register specified by R_S is transferred to the base register specified by RD_B.

SUMMARY EXPRESSION

$(R_S) \rightarrow RD_B$

CONDITION CODE RESULTS

CC1: Unchanged
CC2: Unchanged
CC3: Unchanged
CC4: Unchanged

NOTE

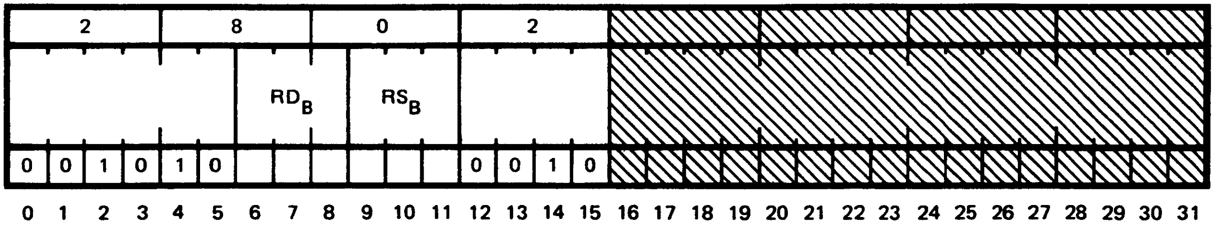
This instruction is used for the base register mode only.

BASE REGISTER MODE EXAMPLE

The contents of GPR5 are transferred to RD_B4.

Memory Location:	300C
Hexadecimal Instruction:	2E51 (R_S=5, RD_B=4)
Assembly Language Coding:	TRBR 5, 4

Before	PSD1 0200300C	GPR5 03030303	BR4 02031678
After	PSD1 0200300E	GPR5 03030303	BR4 03030303



830181A

DEFINITION

The contents of the base registers specified by RD_B and RS_B are exchanged.

SUMMARY EXPRESSION

$(RD_B) \rightarrow RS_B$
$(RS_B) \rightarrow RD_B$

CONDITION CODE RESULTS

CC1: Unchanged
CC2: Unchanged
CC3: Unchanged
CC4: Unchanged

NOTE

This instruction is used in the base register mode only.

BASE REGISTER MODE EXAMPLE

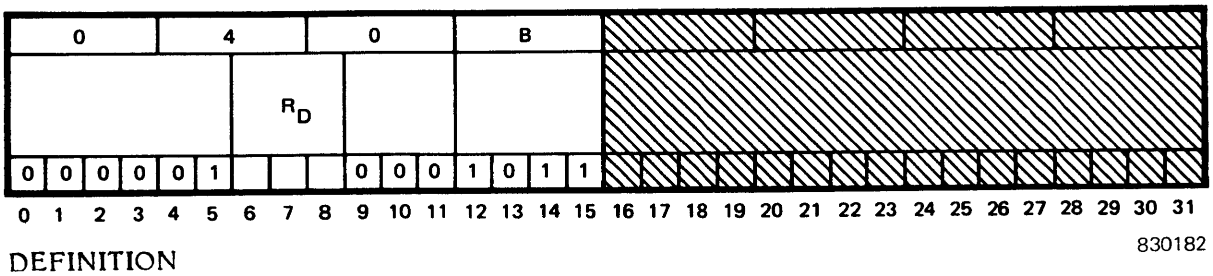
The contents of RD_B3, and RS_B4 are exchanged.

Memory Location:	200C
Hexadecimal Instruction:	2A32 (RD_B=3, RS_B=4)
Assembly Language Coding:	XCBR 3, 4

Before	PSD1 0200200C	BR3 11111111	BR4 22222222
After	PSD1 0200200E	BR3 22222222	BR4 11111111

This page intentionally left blank

READ PROCESSOR STATUS WORD TWO (V6 ONLY) RPSWT
040B d



DEFINITION

If the bit 0 of the general purpose register specified by R_D is set to zero the execution of this instruction causes the contents of the Processor Status Word Two to be read and copied to R_D. The value of the PSD transferred by this instruction is the same as when saved as the result of an interrupt or a trap. The Retain Bit (bit 47 of the PSD or bit 15 of PSW2) is undefined in the resultant value. The value in the interrupt control flags field (bits 48 and 49 of the PSD or bits 16 and 17 of PSW2) will not necessarily match the value of the most recent PSD loaded. Bit 48 is undefined and bit 49 will reflect the current state of the Block External Interrupt Flag in the V6 CPU.

If bit 0 of the general purpose register specified by R_D is set to one (1) the V6 CPU Configuration Word will be returned in R_D.

SUMMARY EXPRESSION

$(PSD2) \rightarrow R_D$

CONDITION CODE RESULTS

The condition codes are not changed by this instruction.

NOTES

1. This instruction is not privileged.
2. The RPSWT instruction operates differently for V6 and V9 CPU. This section describes the operation for the V6 CPU.

NONBASE AND BASE REGISTER MODE EXAMPLE (GPR BIT0=0)

The contents of PSD2 are transferred to GPR4.

Memory Location:	3EDA
Hexadecimal Instruction:	060B (R_D=4)
Assembly Language Coding:	RPSWT 4

Before	PSD1 F0003EDA (Nonbase) F2003EDA (Base)	GPR4 00003082	PSD2 80004058
After	PSD1 F0003EDD (Nonbase) F2003EDD (Base)	GPR4 80004058	PSD2 80004058

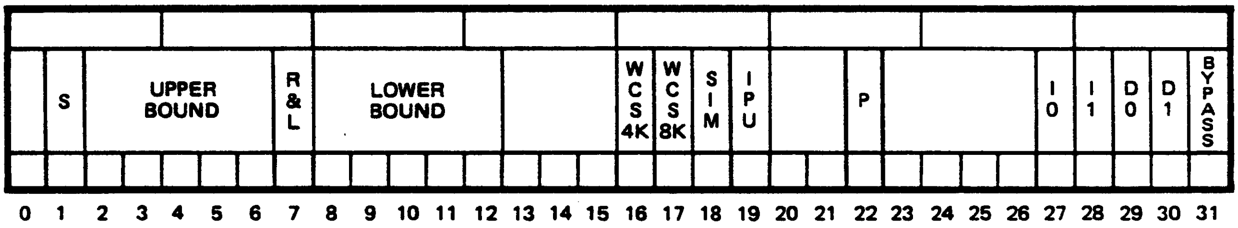
NONBASE AND BASE REGISTER MODE EXAMPLE (GPR bit 0=1)

The contents of the V6 CPU Configuration Control Word are transferred to GPR4.

Memory Location:		3EDA
Hexadecimal Instruction:	060B (R_D=4)	
Assembly Language Coding:	RPSWT 4	

Before	PSD1	GPR4	CCW
	F0003EDA (Nonbase)	80000000	0000001F
	F2003EDA (Base)		
After	PSD1	GPR4	CCW
	F0003EDD (Nonbase)	0000001F	0000001F
	F2003EDD (Base)		

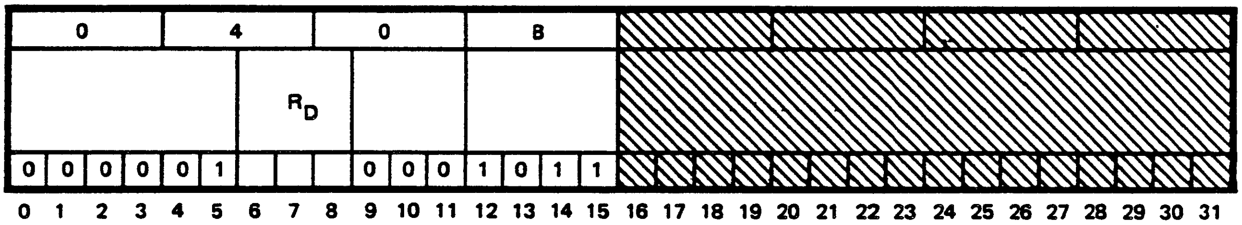
Contents of GPR specified by R_D:



830182A

Bits 0 = Reserved
1 = Shared Memory Enabled (=1)/Disabled(=0)
2-6 = Upper Bound of Shared Memory
7 = Read & Lock Enabled(=1)/Disabled(=0)
8-12 = Lower Bound of Shared Memory
13-15 = Reserved
16 = 4K WCS Option Present(=1)/inoperable(=0)
17 = 8K WCS Option Present(=1)/inoperable(=0)
18 = Firmware Control Store Mode ROMSIM(=1)/PROM(=0)
19 = IPU Present(=1)/IPU Inoperable(=0)
20-21 = Reserved
22 = Access Protection (=1)/No Access Protection (=0)
23-26 = Reserved
27 = Instruction Cache Bank 0 on (=1)/Off(=0)
28 = Instruction Cache Bank 1 on (=1)/Off(=0)
29 = Data Cache Bank 0 on (=1)/Off(=0)
30 = Data Cache Bank 1 on (=1)/Off(=0)
31 = Instruction Cache Enabled (=1)/Disabled(=0)

READ PROCESSOR STATUS WORD TWO (V9 ONLY) RPSWT
040B d



830182A

DEFINITION

The read processor status word two (RPSWT) instruction is used to read the second word of the processor status word (PSD2), the V9 CPU configuration word (CCW), or, if shadow memory is installed, the shadow memory configuration word for the V9 CPU (SMCWC) or IPU (SMCWI).

The contents of the GPR specified by R are examined and if equal to zeros, PSD2 is read and transferred to R; if bit 00 is set, the CCW is read and transferred; if bit 01 is set, SMCWC is read and transferred; if bit 02 is set, SMCWI is read and transferred. In the case of the CCW, SMCWC, or SMCWI conditions, bits 03 through 31 of R are ignored.

The value of PSD2 transferred by this instruction is the same as when saved as the result of an interrupt or trap. The retain bit (bit 47 of the PSD) is undefined in the resultant value. The value in the interrupt control flags field (bits 48 and 49 of the PSD) will not necessarily match the value of the most recent PSD loaded. Bit 48 is undefined and bit 49 will reflect the current state of the block external interrupt flag in the V9 CPU.

NOTES

1. Only one of the above conditions can be present at any one time.
2. RPSWT is an unprivileged instruction.
3. RPSWT is designed for diagnostic purposes and for reading the status or configuration information.
4. The RPSWT instruction operates differently for V6 and V9 CPU. This section describes the operation for the V9 CPU.

CONDITION CODE RESULTS

CC1:	No change
CC2:	No change
CC3:	No change
CC4:	No change

RPSWT EXAMPLES:

1. Contents of R_D equal to zero.

Memory Location:	03EDA
Hexadecimal Instruction:	06 0B (R=4)
Assembly Language Coding:	RPSWT 4

The contents of PSD2 are transferred to GPR4.

Before	PSD1	GPR4	PSD2
	F0003EDA (Nonbase)	00000000	00000818
	F2003EDA (Base)		
After	PSD1	GPR4	PSD2
	F0003EDC (Nonbase)	00000818	00000818
	F2003EDC (Base)		

RPSWT EXAMPLES (Cont.)

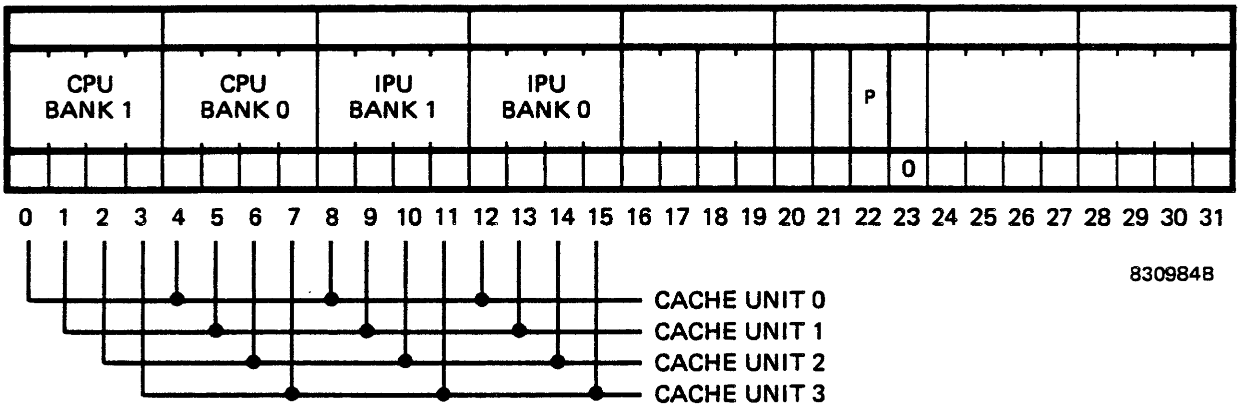
2. Bit 00 of R_D = 1.

Memory Location:	03EDA
Hexadecimal Instruction:	06 0B (R=4)
Assembly Language Coding:	RPSWT 4

The contents of the CPU Configuration Word (CCW) are transferred to GPR4.

Before	PSD1		GPR4	CCW
	F0003EDA	(Nonbase)	80000000	000004B4
	F2003EDA	(Base)		
After	PSD1		GPR4	CCW
	F0003EDC	(Nonbase)	000004B4	000004B4
	F2003EDC	(Base)		

Contents of R:



Bits 00-15	=0	Cache/Shadow Unit Present and Operational
	=1	Cache/Shadow Unit Not Present
16	=0	MACC Present and Operational in CP1
	=1	MACC Not Present in CP1
17	=0	MACC Present and Operational in CP2
	=1	MACC Not Present in CP2
18	=0	CP1 Present and Online
	=1	CP1 Not Present
19	=0	CP2 Present and Online
	=1	CP2 Not Present
20	=0	IPU Configured
	=1	No IPU
21	=0	Shared Memory Configured
	=1	No Shared Memory
22	=0	No Access Protection
	=1	Access Protection
23		Reserved
24-27		CPU Firmware Version
28-31		CPU Firmware Revision Level

RPSWT EXAMPLES (Cont.)

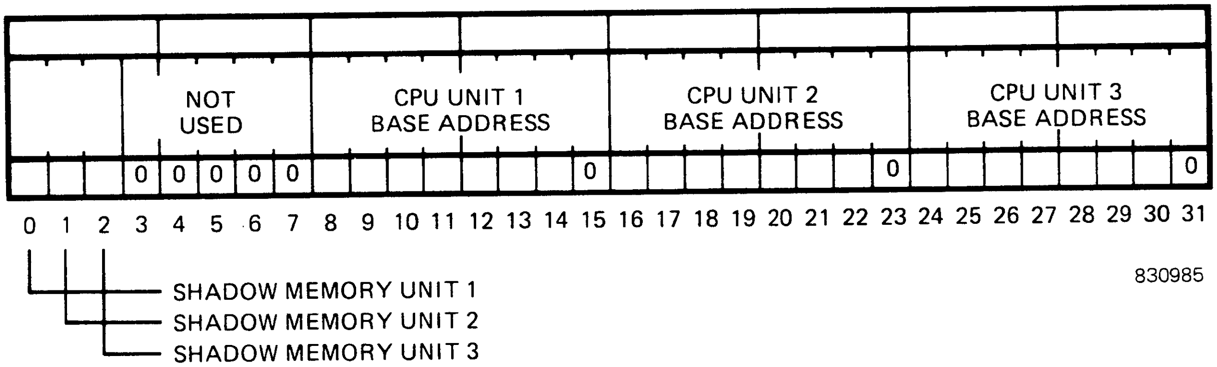
3. Bit 01 of R_D = 1.

Memory Location:	03EDA
Hexadecimal Instruction:	06 0B (R=4)
Assembly Language Coding:	RPSWT 4

Bit 01 of GPR4 is equal to one; transfer the contents of Shadow Memory Configuration Word for CPU (SMCWC) to GPR4.

Before	PSD1		GPR4	SMCWC
	F0003EDA	(Nonbase)	40000000	C01E2E00
	F2003DCA	(Base)		
After	PSD1		GPR4	SMCWC
	F0003EDC	(Nonbase)	C01E2E00	C01E2E00
	F2003EDC	(Base)		

Contents of R:



Bits 00-02	=1	Shadow Memory Unit Present and Operational
	=0	Shadow Memory Unit Not Present or Operational
03-07		Not Used
08-14	=	Shadow Memory Unit 1 Base Address (bits 08-14)
16-22	=	Shadow Memory Unit 2 Base Address (bits 08-14)
24-30	=	Shadow Memory Unit 3 Base Address (bits 08-14)

RPSWT (Cont.)

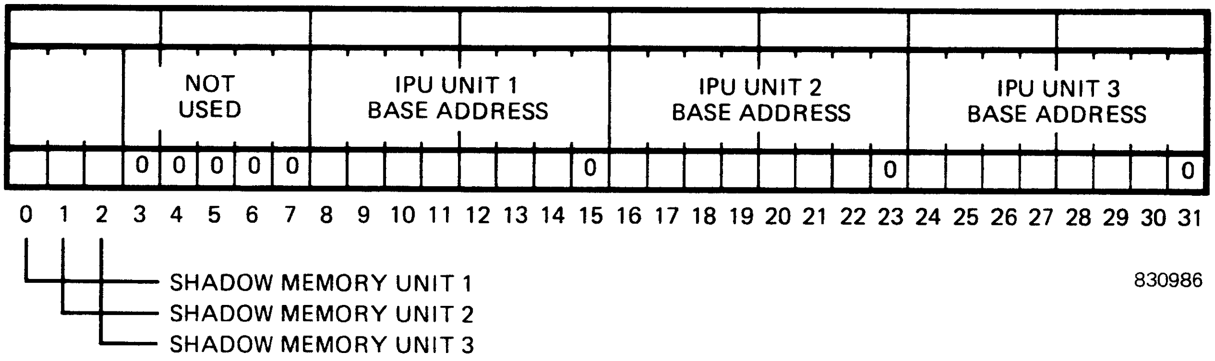
4. Bit 02 of R_D=1.

Memory Location: 03EDA
Hexadecimal Instruction: 06 0B (R=4)
Assembly Language Coding: RPSWT 4

Bit 02 of GPR4 is equal to a one; transfer the contents of Shadow Memory Configuration Word for IPU (SMCWI) to GPR4.

	PSD1	GPR4	SMCWI
Before	F0003EDA (Nonbase) F2003EDA (Base)	20000000	C01E2E00
After	F0003EDC (Nonbase) F2003EDC (Base)	C01E2E00	C01E2E00

Contents of R:



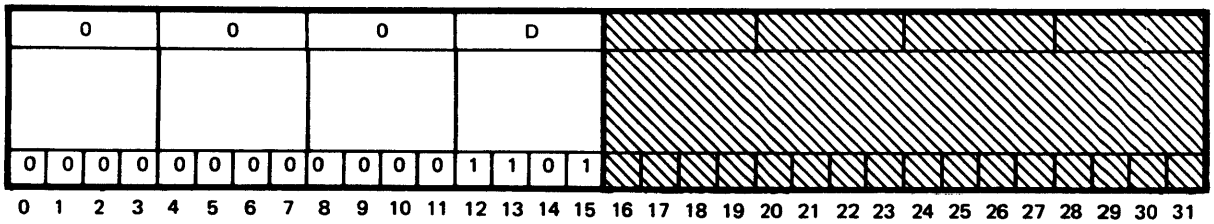
830986

Bits 00-02	=1	Shadow Memory Unit Present and Operational
	=0	Shadow Memory Unit Not Present and Invalid
03-07		Not Used
08-14	=	Shadow Memory Unit 1 Base Address (bits 08-14)
16-22	=	Shadow Memory Unit 2 Base Address (bits 08-14)
24-30	=	Shadow Memory Unit 3 Base Address (bits 08-14)

6.2.3 Memory Management Instructions

The CPU may operate in a mapped environment or an unmapped environment. When the CPU is operating mapped, the physical map registers are defined in the map image descriptor list (MIDL). Mapped mode allows translation of logical addresses to physical addresses for increased executable memory and better memory utilization. The MIDL can be used to link contiguous logical addresses to whatever physical map blocks are available.

The nonextended addressing option allows the CPU to access instructions or operands in the first 128K words of memory. The extended addressing option provides the access to any bit, byte, halfword, word, or doubleword operand residing anywhere up to 4M words. In base register mode, a maximum of 4M words are available for both instructions and operands.



830183

DEFINITION

The central processing unit (CPU) enters the extended addressing mode.

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

NOTE

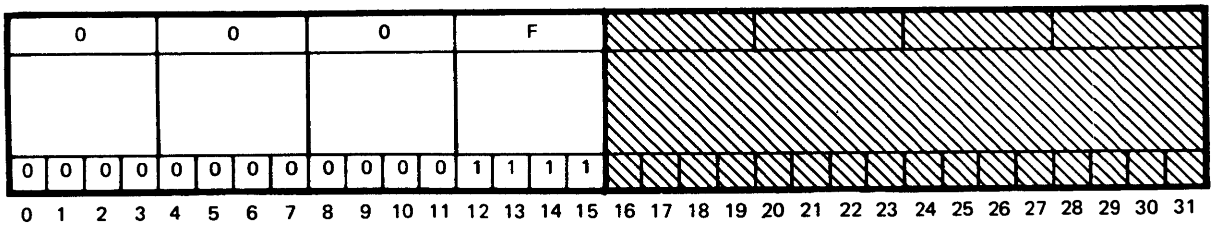
1. This instruction sets bit 5 in the first word of the PSD.

2. This instruction is illegal in the base register mode.

EXAMPLE

Before	PSD1 20001000
After	PSD1 24001002

CLEAR EXTENDED ADDRESSING
000F



830184

DEFINITION

The central processing unit (CPU) enters the normal (nonextended) addressing mode.

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

NOTE

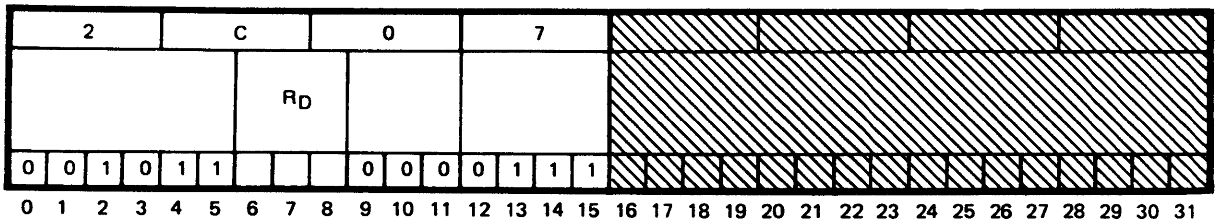
1. This instruction clears bit 5 in the first word of the PSD.

2. This instruction is illegal in the base register mode.

EXAMPLE

Before	PSD1 14002200
After	PSD1 10002202

LOAD Map — LMAP
2C07 — d



830185

DEFINITION

Loads the map image descriptor list (MIDL) from main memory into the central processing unit (CPU) map registers. R_D contains the real address of a program status doubleword (PSD) to be used in the map loading process.

SUMMARY EXPRESSION

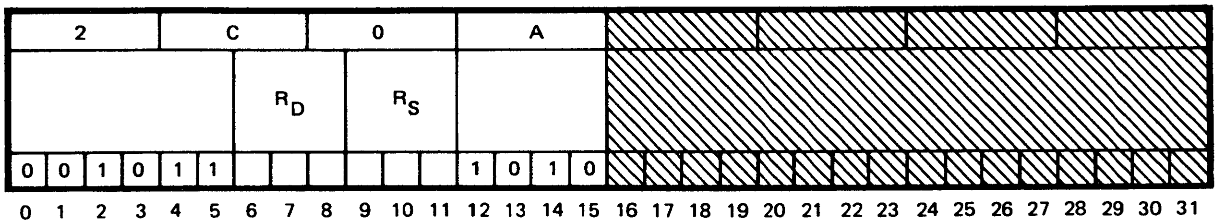
(MIDL) → Map registers

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

NOTES

1. LMAP is a privileged instruction.
2. This instruction is used primarily for diagnostic purposes.
3. The CPU must be unmapped.
4. Only map load functions are performed, with no context switching.



841314

DEFINITION

This instruction causes the map entry specified by R_S bits 21-31, to be transferred to the general purpose register (GPR) specified by R_D.

SUMMARY EXPRESSION

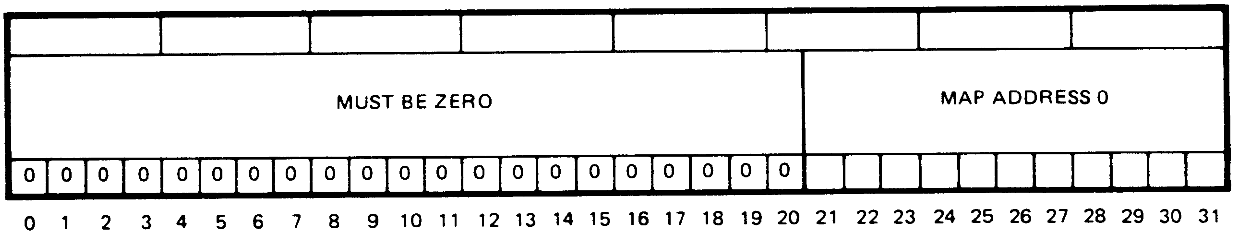
Map addressed by R_S(21-31) → R_D(16-31)
Buffer HIT/MISS → $R_{D(0)}$

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

1. TMAPR is a privileged instruction.

2. The format for R_S is as follows:

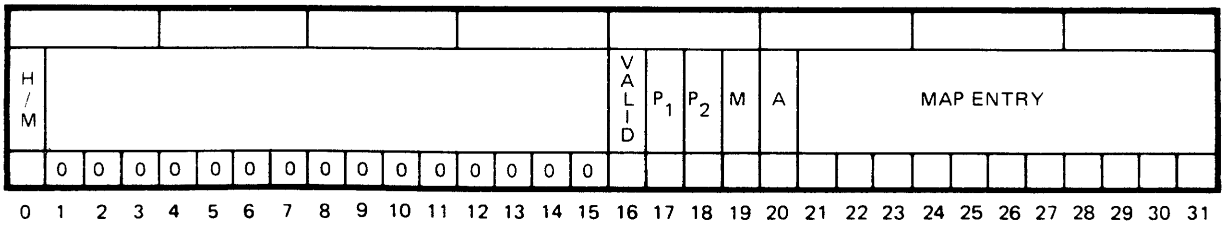


3. The CPU may be unmapped.

4. Bit 0 of R_D is defined as follows:

   0=MISS
   1=HIT

5. The format of R_D is as follows:



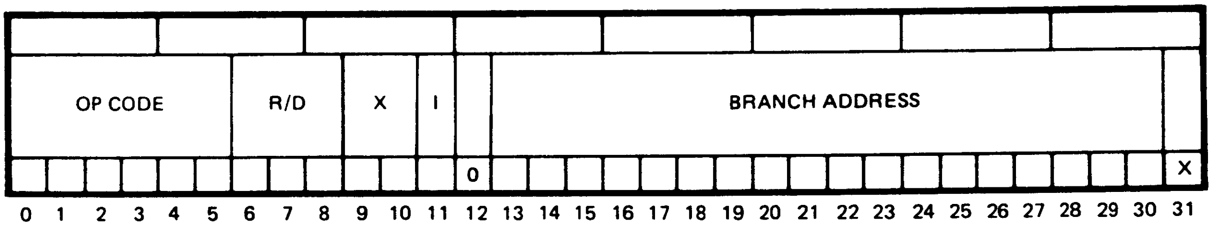
This page intentionally left blank

6.2.4 Branch Instructions

Branch instructions test for certain conditions and cause branching to another address if the conditions specified by the instruction are satisfied. Branch instructions permit referencing subroutines, repeating segments of programs, or returning to the next instruction within a sequence.

6.2.4.1 Instruction Format

The branch instruction group uses the standard memory reference instruction format (see Figure 6-1) in the base register mode, except that Bit 12 must be zero. However, in the nonbase register mode the branch instructions use the following variation to the memory reference instruction format:



830351

Bits 0-5 Define the operation code

Bits 6-8 Vary in usage as follows:

Instruction	Contents/Usage
BU,BFT	000
BCT,BCF	D field
BIB,BIH, BIW,BID	Register number
BL	001

Bits 9-10 Designate one of three index registers

Bit 11 Indicates whether an indirect addressing operation is to be performed

Bit 12 Is zero

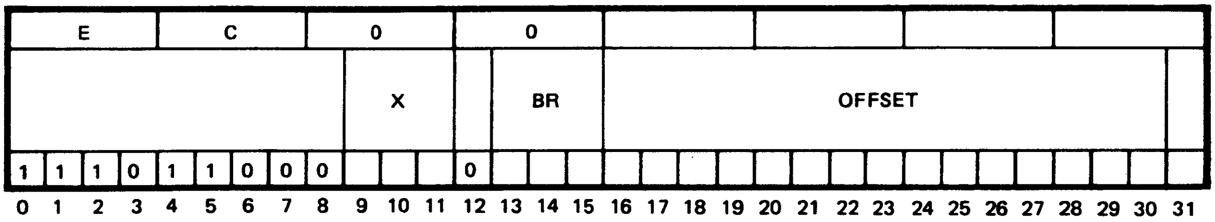
Bits 13-30 Specify the branch address when X and I fields are zero

Bit 31 Do not care

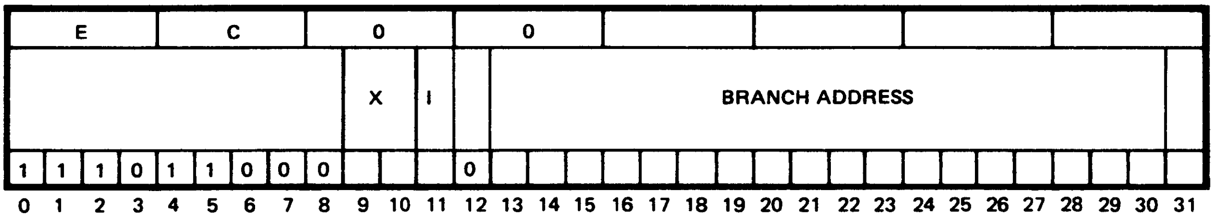
In the base register mode the PC holds a 24 bit address and in the non-base register mode a 19 bit address.

6.2.4.2 Condition Code

Condition codes are neither changed by branches in the base register mode nor in the non-base register mode if the instruction does not have the I bit set. Branches with the I bit set copy the condition codes to the PSD from the corresponding locations of the final location in the indirect chain.



BASE REGISTER MODE



830187

NONBASE REGISTER MODE

DEFINITION

In both, base register mode and nonbase register mode, the effective address (EA) is transferred to the Program Counter (PC) field in the Program Status Doubleword (PSD). However, in the nonbase register mode, if the indirect bit of the instruction word is set (I=1), the condition codes are set from the corresponding bits in the last memory word in the indirect chain. Bit 0 (the privileged state bit) of the PSD remains unchanged.

SUMMARY EXPRESSION

$EA \rightarrow PSD_{13-30}$ Nonbase register format

If I=1,

$(EWL)_{1-4} \rightarrow PSD_{1-4}$

$EA \rightarrow PSD_{08-30}$ Base register format

CONDITION CODE RESULTS

When indirect addressing is not specified, the condition codes remain unchanged. With indirect addressing, the condition codes are transferred as shown in the definition and the summary expression.

BASE REGISTER MODE EXAMPLE

The contents of BR6, GPR4 and the instruction offset are added and transferred to the PSD, bits 8-30.

Memory Location:	001000
Hexadecimal Instruction:	EC461400 (X=4, BR=6)
Assembly Language Coding:	BU X '1400' (6), 4

Before	PSD1	GPR4	BR6	Effective Address
	22001000	00000004	00000010	001414
After	PSD1	GPR4	BR6	
	22001414	00000004	00000010	

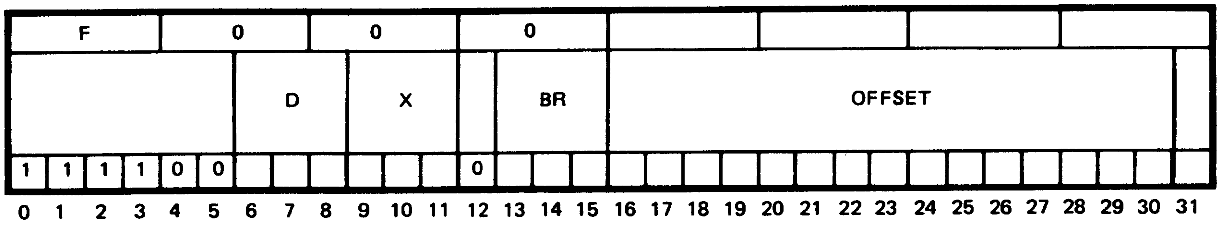
NONBASE REGISTER MODE EXAMPLE

The contents of bits 13-30 of the instruction replace the corresponding portion of the PSD. The condition code remains unchanged.

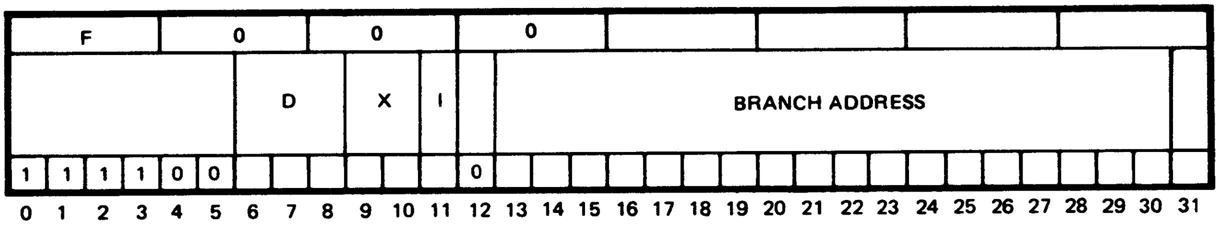
Memory Location:	01000
Hexadecimal Instruction:	EC 00 14 14 (X=0, I=0)
Assembly Language Coding:	BU X'1414'

Before	PSD1
	20001000
After	PSD1
	20001414

BRANCH CONDITION FALSE — BCF
F000 v,*m,x



BASE REGISTER FORMAT



830188

NONBASE REGISTER FORMAT

DEFINITION

In both base register mode and nonbase register mode the effective address (EA) in the instruction is transferred to the program counter (PC) field in the program status doubleword (PSD), if the condition specified by the D field (bits 6-8 of the instruction) is not present. The seven specifiable conditions are tabulated below. If the condition is as specified, the next instruction in sequence is executed. However, in the nonbase register mode, if the branch is taken and if the indirect bit of the instruction word is set (I=1) when the branch occurs, the condition codes are set from the corresponding bits in the last memory word in the indirect chain. Bits 0 and 5-12 of the PSD are unchanged.

D Field (Hex)	Branch Condition (Branch If)
1	CC1 = zero
2	CC2 = zero
3	CC3 = zero
4	CC4 = zero
5	CC2 and CC4 both = zero
6	CC3 and CC4 both = zero
7	CC1, CC2, CC3, and CC4 all = zero

SUMMARY EXPRESSION

When the branch is taken,

$EA \rightarrow PSD_{13-30}$ Nonbase register format

But if I=1,

$(EWL)_{1-4} \rightarrow PSD_{1-4}$

EA → PSD_{08-30} Base register format

When the branch is not taken,

PC + 1 word → PC

CONDITION CODE RESULTS

When the branch is not taken or indirect addressing is not specified, the condition codes remain unchanged. With indirect addressing and the branch taken, the condition codes are transferred as shown in the definition and the summary expression.

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added and transferred to the PSD.

Memory Location:	02094
Hexadecimal Instruction:	F1062100 (C_1 C_2 C_3 = 2, X=0, BR=6)
Assembly Language Coding:	BCF 2,X'2100'(6)

Before	PSD1 12002094	BR6 0000004C	Effective Address 00214C 38
After	PSD1 1200214C	BR6 0000004C	Effective Address 00214C 38

NONBASE REGISTER MODE EXAMPLE

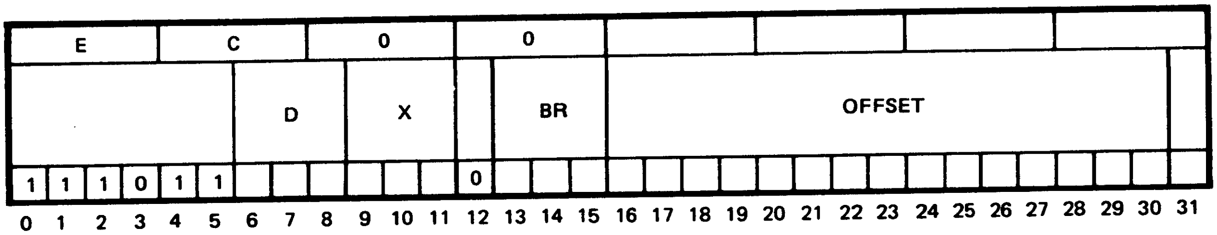
Condition code bit 2 is not set. The effective address (in this case bit 13-30 of the instruction) is transferred to the PSD.

Memory Location:	02094
Hexadecimal Instruction:	F1 00 21 4C ($C_1C_2C_3$=2, X=0, I=0)
Assembly Language Coding:	BCF 2,X'214C'

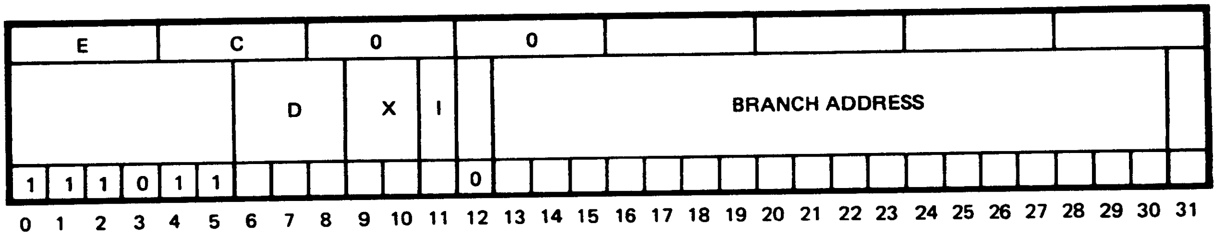
Before	PSD1 10002094	Effective Address 0214C 38
After	PSD1 1000214C	Effective Address 0214C 38

BRANCH CONDITION TRUE
EC00

BCT
v,*m,x



BASE REGISTER FORMAT



830189

NONBASE REGISTER FORMAT

DEFINITION

In both base register mode and nonbase register mode the effective address (EA) in the instruction is transferred to the program counter (PC) field in the program status doubleword (PSD), if the condition specified by the D field (bits 6-8 of the instruction) is present. The seven specifiable conditions are tabulated below. However, in the nonbase register mode, if the branch is taken and if the indirect bit of the instruction word is set (I=1) when the branch occurs, the condition codes are set from the corresponding bits in the last memory word in the indirect chain. Bits 0 and 5-12 of the PSD remain unchanged.

D Field (Hex)	Branch Condition (Branch If)
1	CC1 = one
2	CC2 = one
3	CC3 = one
4	CC4 = one
5	CC2 v CC4 = one
6	CC3 v CC4 = one
7	CC1 v CC2 v CC3 v CC4 = one

SUMMARY EXPRESSION

When the branch is taken,

$EA \rightarrow PSD_{13-30}$ Nonbase register format

But if I=1,

$(EWL)_{1-4} \rightarrow PSD_{1-4}$

$EA \rightarrow PSD_{08-30}$ Base register format

When the branch is not taken,

PC + 1 word → PC

CONDITION CODE RESULTS

When the branch is not taken or indirect addressing is not specified, the condition codes remain unchanged. With indirect addressing and the branch taken, the condition codes are transferred as shown in the definition and the summary expression.

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added and transferred to the PSD.

Memory Location:	01000
Hexadecimal Instruction:	EC861400 (D=1, X=0, BR=6)
Assembly Language Coding:	BCT 1,X'1400'(6)

Before	PSD1 52001000	BR6 00000014	Effective Address 001414 56
After	PSD1 52001414	BR6 00000014	Effective Address 001414 56

NONBASE REGISTER MODE EXAMPLE

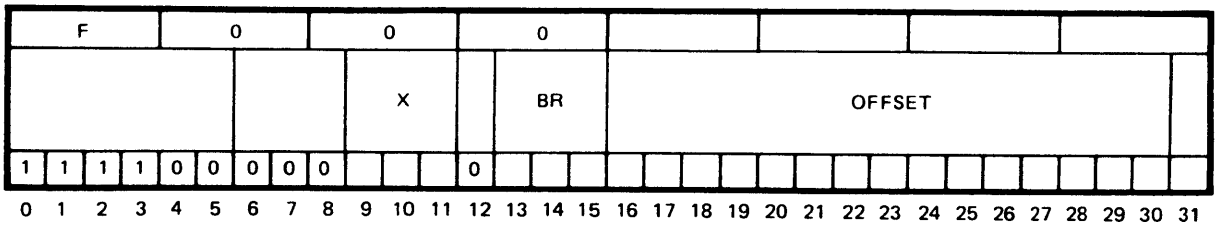
The contents of bits 13-30 of the instruction are transferred to bits 13-30 of the PSD.

Memory Location:	01000
Hexadecimal Instruction:	EC 80 14 14 (D=1, X=0, I=0)
Assembly Language Coding:	BCT 1,X'1414'

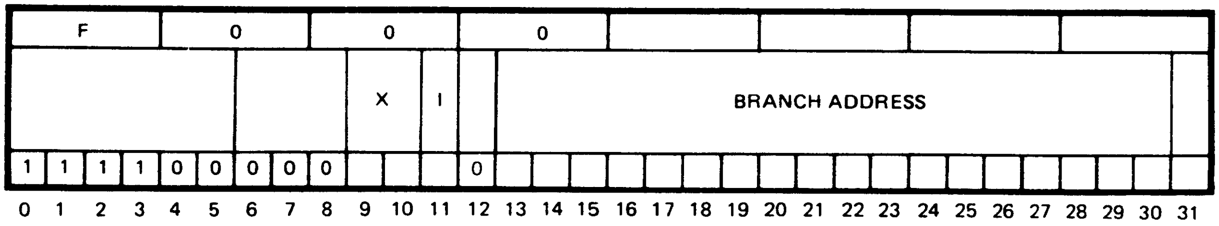
Before	PSD1 50001000	Effective Address 01414 56
After	PSD1 50001414	Effective Address 01414 56

BRANCH FUNCTION TRUE F000

BFT *m,x



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

830190

DEFINITION

The value in the condition code register (0-F_{16}) designates a corresponding bit in the mask register to be tested (refer to the table below). If the appropriate mask register bit is set, the branch is taken. If the bit is not set when tested, the next instruction in sequence is executed. R4 is used as the mask register.

CC Register Value	Mask Register Bit No.
0	16
1	17
2	18
3	19
4	20
5	21
6	22
7	23
8	24
9	25
A	26
B	27
C	28
D	29
E	30
F	31

If the instruction (nonbase register mode) invokes indirect addressing (I=1) and the branch is taken, the contents of the condition codes are set from the corresponding bits in the last memory word in the indirect chain.

SUMMARY EXPRESSION

When the branch is taken,

EA → PSD_{13-30} Nonbase register format

But if I=1,

$(EWL)_{1-4}$ → PSD_{1-4}

EA → PSD_{08-30} Base register format

When the branch is not taken,

PC +1 word → PC

CONDITION CODE RESULTS

When the branch is not taken and indirect addressing is not specified, the condition codes remain unchanged. With indirect addressing and the branch taken, the condition codes are transferred as shown in the definition and the summary expression.

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. Bit 30 of GPR4 defines a function for which CC1= CC2= CC3=1, CC4=0. This function is true, so a branch is effected.

Memory Location:	01000
Hexadecimal Instruction:	F0062000 (X=0, BR=6)
Assembly Language Coding:	BFT X'2000'(6)

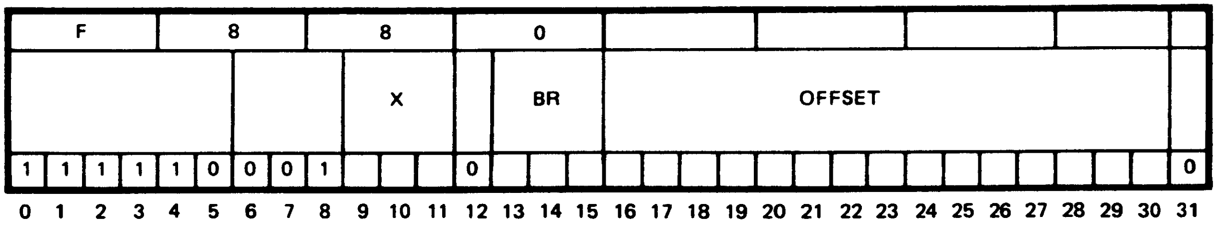
Before	PSD1 72001000	GPR4 00000002	BR6 00000004
After	PSD1 72002004	GPR4 00000002	BR6 00000004

NONBASE REGISTER MODE EXAMPLE

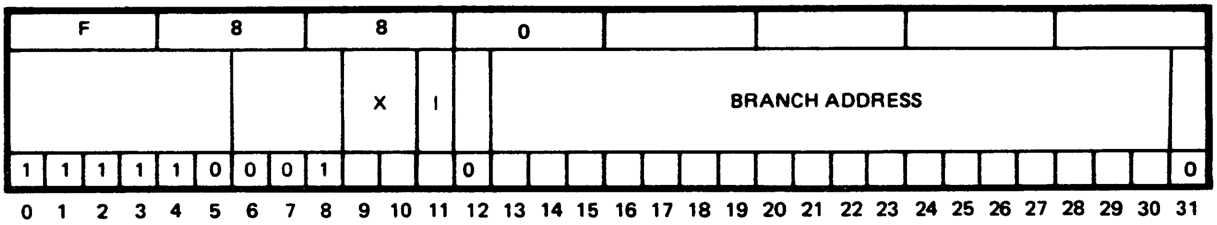
Bit 30 of GPR4 defines a function for which CC1=CC2=CC3=1, CC4=0. This function is true, so a branch is effected.

Memory Location:	01000
Hexadecimal Instruction:	F0 00 20 00 (X=0, I=0)
Assembly Language Coding:	BFT X'2000'

Before	PSD1 70001000	GPR4 00000002
After	PSD1 70002000	GPR4 00000002



BASE REGISTER FORMAT



830191

NONBASE REGISTER FORMAT

DEFINITION

The contents in the first word of the program status doubleword (PSD) are incremented by one word and transferred to general purpose register 0 (GPR0). In base mode, the effective address is transferred to bits 08-30 of PSD1, and bits 0-7 of PSD1 remain unchanged. In non-base mode, if the indirect bit of the instruction word is reset (I=0), the effective address is transferred to bits 13-30 of PSD1, and bits 1-12 of PSD1 remain unchanged.

In non-base mode, if the indirect bit of the instruction word is set (I=1), the effective address is transferred to bits 13-30 of PSD1, and the condition codes are set from the corresponding bits in the last memory word in the indirect chain. Bit 0 (privileged state bit) and bits 5-12 of the PSD1 remain unchanged.

SUMMARY EXPRESSION

$PSD \rightarrow R0$

If I=zero

$EA \rightarrow PSD_{13-30}$ Nonbase register format

If I=one

$EWL_{1-4}, EA \rightarrow PSD_{1-4 \text{ and } 13-30}$

$EA \rightarrow PSD_{08-30}$ Base register format

CONDITION CODE RESULTS

If the indirect bit is reset to zero, the condition code remains unchanged.

CC1: Is set if (I) is equal to one and (EWL_1) is equal to one
CC2: Is set if (I) is equal to one and (EWL_2) is equal to one
CC3: Is set if (I) is equal to one and (EWL_3) is equal to one
CC4: Is set if (I) is equal to one and (EWL_4) is equal to one

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of the incremented PSD are transferred to GPR0. The logical address is transferred to the PSD.

Memory Location:	0894C
Hexadecimal Instruction:	F886A370 (X=0, BR=6)
Assembly Language Coding:	BL X'A370'(6)

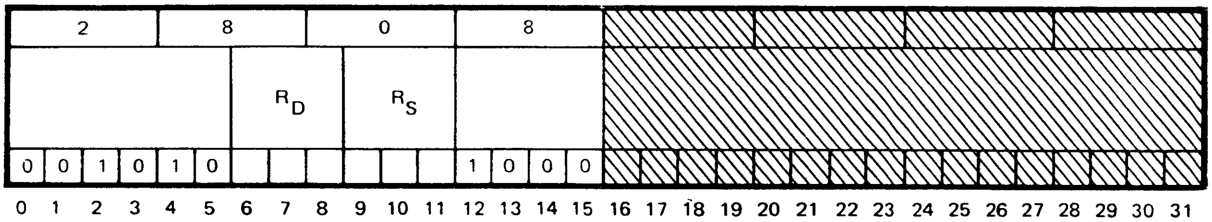
Before	PSD1 1200894C	GPR0 12345678	BR6 00000008
After	PSD1 1200A378	GPR0 12008950	BR6 00000008

NONBASE REGISTER MODE EXAMPLE

The contents of the incremented PSD are transferred to GPR0. The contents of bits 13-30 of the instruction are transferred to bits 13-30 of the PSD.

Memory Location:	0894C
Hexadecimal Instruction:	F8 80 A3 78 (X=0, I=0)
Assembly Language Coding:	BL X'A378'

Before	PSD1 1000894C	GPR0 12345678
After	PSD1 1000A378	GPR0 10008950



830192

DEFINITION

If the RD field is equal to zero, see the BSUB instruction format. The address of the current frame is determined by subtracting 16 words (hex 40) from the contents of Base Register 2 (BR2). This address is aligned to a doubleword boundary by resetting the three LSBs. This address is referred to as the current frame address. The program counter and arithmetic exception bit are stored at the current frame address in main memory (word 0 of the current frame). Word 1 of the current frame is set to '0000 0000'. Base registers 0-7, and general purpose registers 2-7 are stored in words 2-15 of the current frame. The contents of the GPR specified by RD is transferred to base register 3 (BR 3) which is called the argument pointer (AP). Base registers 0 and 2 are set to the current frame address. The contents of the base register specified by RS is transferred to base 1 (BR 1) and the PC portion of the PSD causing control to be transferred to that location in the program.

CONDITION CODE RESULTS

The condition codes remain unchanged.

NOTES

1. Modifications of BR0 will cause unpredictable behavior of the hardware in the call/return sequence.

2. The current frame stack should be bounded on the low end by a write protected or nonexistent page. In this case, a stack overflow will occur on the save of the return PC before the context has been destroyed by the instruction. This will permit error diagnosis to show the cause.

3. If an operand address generated by this instruction crosses a MAP block boundary, an address specification trap will occur. This applies when in either the mapped or unmapped mode.

BASE REGISTER MODE EXAMPLE

```
Memory Location:               4214
Hexadecimal Instruction:       2898
Assembly Language Coding:      Call B1,R1  RS=1; RD=1

Before     PSD1        GPR0  00000000        BR0  000041B2
           8B004214    GPR1  00005100        BR1  00005080
                       GPR2  AAAAAAAA        BR2  000041FC
                       GPR3  BBBBBBBB        BR3  00005000
                       GPR4  CCCCCCCC        BR4  44444444
                       GPR5  DDDDDDDD        BR5  55555555
                       GPR6  EEEEEEEE        BR6  66666666
                       GPR7  FFFFFFFF        BR7  77777777

           Memory Location:      41B8     Don't Care
                                 41BC     Don't Care
                                 41C0     Don't Care
                                   .          .
                                   .          .
                                   .          .
                                 41F4     Don't Care


After      PSD1        GPR0  00000000        BR0  000041B8        CFA
           8A0041B4    GPR1  00005100        BR1  00005080        RS
                       GPR2  AAAAAAAA        BR2  000041B8        CFA
                       GPR3  BBBBBBBB        BR3  00005100        RD
                       GPR4  CCCCCCCC        BR4  44444444        Unchanged
                       GPR5  DDDDDDDD        BR5  55555555        Unchanged
                       GPR6  EEEEEEEE        BR6  66666666        Unchanged
                       GPR7  FFFFFFFF        BR7  77777777        Unchanged

           Memory Location:      41B8     01004216     Next PC
                                 41BC     00000000     Flag Word
                                 41C0     000041B2     BR0
                                 41C4     00005080     BR1
                                 41C8     000041FC     BR2
                                 41CC     50005000     BR3
                                 41D0     44444444     BR4
                                 41D4     55555555     BR5
                                 41D8     66666666     BR6
                                 41DC     77777777     BR7
                                 41E0     AAAAAAAA     GPR2
                                 41E4     BBBBBBBB     GPR3
                                 41E8     CCCCCCCC     GPR4
                                 41EC     DDDDDDDD     GPR5
                                 41F0     EEEEEEEE     GPR6
                                 41F4     FFFFFFFF     GPR7
```

PROCEDURE CALL EXAMPLE EXPLANATION

If the instruction RD field is equal to zero, this is a BSUB instruction. If the instruction RD field is not zero, the processor computes the current frame address (CFA) by: 1. reading the contents of BR2 (41FC-hex); 2. subtracting 40 (hex) resulting in 41BC (hex); 3. doubleword bounding the resulting address to generate 41B8 as the effective current frame address (CFA). Next, the processor computes the address (PC value) of the instruction following the call, and determines the state of the enable/disable arithmetic exception bit (PSD bit 7). The resulting 32 bit word contains zeros in bits 00-06, PSD bit 7 in bit 7, and the PC value in bits 08-31. Bit 31 is not used and is set to zero. The resulting word represents the subroutine return address and is stored in memory addressed by the CFA (Word 0). Next, the processor sets memory location CFA plus one word equal to zero which indicates the call frame was generated by a CALL instruction. Base registers 0 through 7 are stored in the current frame words 2 through 9 and general purpose registers 2 through 7 are stored in the current frame words 10 through 15. The instruction procedure call address, addressed by the instruction RS field (base register 1 in the example) is transferred to base register 1 and to PSD1 PC field (5080-hex). The instruction argument pointer, addressed by the instruction RD field (GPR 1 in the example) is transferred from the GPR (designated in RD) to base register 3 (5100-hex).

This page intentionally left blank

5				C				0				8																			
						R_D			X				BR			OFFSET															
0	1	0	1	1	1	0	0	0				1																		0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

830474

DEFINITION

If the RD field is equal to zero, refer to the BSUBM instruction format. The address of the current frame is determined by subtracting 16 (hex 40) words from the contents of base register 2 (BR2). This address is aligned on a doubleword boundary in main memory by resetting the three LSBs. This address is known as the current frame address. The program counter and arithmetic exception bit are stored at the current frame address (word 0 of the current frame). Word 1 of the current frame is set to '0000 0000'. Base registers 0-7 and general purpose registers 2-7 are stored in words 2-15 of the current frame. The contents of the GPR specified by RD is transferred to BR3, which is known as the argument pointer (AP). BR0 and BR2 are set to the current frame address. The effective word location (EWL) specified by the effective word address is accessed and it's contents transferred to BR1 and the PC portion of the PSD causing control to be transferred to that location in the program.

CONDITION CODE RESULTS

The condition codes remain unchanged.

NOTES

1. Modifications of BR0 will cause unpredictable behavior of the hardware in the call/return sequence.

2. The current frame stack should be bounded on the low end by a write protected or nonexistent page. In this case, a stack overflow will occur on the save of the return PC before the context has been destroyed by the instruction. This will permit error diagnosis to show the cause.

3. An address specification trap will occur if the instruction effective address is not word aligned.

4. If an operand address generated by this instruction crosses a MAP block boundary, an address specification trap will occur. This applies when in either the mapped or unmapped mode.

BASE REGISTER MODE EXAMPLE

Memory Location:	4214
Hexadecimal Instruction:	5C885000
Assembly Language Coding:	CALLM R1, '5000' RD=1

Before	PSD1	GPR0	00000000	BRO	000041B2
	8B004214	GPR1	00005100	BR1	00005000
		GPR2	AAAAAAAA	BR2	000041FC
		GPR3	BBBBBBBB	BR3	00005080
		GRP4	CCCCCCCC	BR4	44444444
		GPR5	DDDDDDDD	BR5	55555555
		GPR6	EEEEEEEE	BR6	66666666
		GPR7	FFFFFFFF	BR7	77777777

Memory Location:	41B8	Don't Care
	41BC	Don't Care
	41C0	Don't Care
	.	.
	.	.
	.	.
	41F4	Don't Care
	5000	00005080 Procedure Add.

After	PSD1	GPR0	00000000	BR0	000041B8	CFA
	8B005080	GPR1	00005100	BR1	00005080	Procedure Add
		GPR2	AAAAAAAA	BR2	000041B8	CFA
		GPR3	BBBBBBBB	BR3	00005100	GPR RD
		GPR4	CCCCCCCC	BR4	44444444	Unchanged
		GPR5	DDDDDDDD	BR5	55555555	Unchanged
		GPR6	EEEEEEEE	BR6	66666666	Unchanged
		GPR7	FFFFFFFF	BR7	77777777	Unchanged

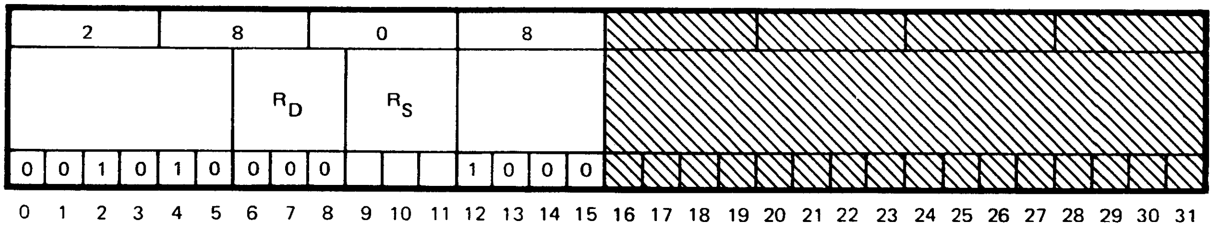
Memory Location:	41B8	01004218	Next PC
	41BC	00000000	Flag Word
	41C0	000041B2	BR0
	41C4	00005000	BR1
	41C8	000041FC	BR2
	41CC	00005080	BR3
	41D0	44444444	BR4
	41D4	55555555	BR5
	41D8	66666666	BR6
	41DC	77777777	BR7
	41E0	AAAAAAAA	GPR2
	41E4	BBBBBBBB	GPR3
	41E8	CCCCCCCC	GPR4
	41EC	DDDDDDDD	GPR5
	41F0	EEEEEEEE	GPR6
	41F4	FFFFFFFF	GPR7
	5000	00005080	

If the instruction RD field is equal to zero, this is a BSUBM instruction. If the instruction RD field is not zero, the processor computes the current frame address (CFA) by: 1. reading the contents of base register 2 (41FC-hex); 2. subtracting 40 (hex) resulting in 41BC (hex); 3. doubleword bounding the resulting address to generate 41B8 as the effective CFA. Next, the processor computes the address (PC value) of the instruction following CALLM, and determines the state of the enable/disable arithmetic exception bit (PSD bit 7). The resulting 32 bit word contains zeros in bits 00-06, PSD bit 7 in bit 7, and the PC value in bits 08-30. Bit 31 is not used and is set to zero. The resulting word represents the subroutine return address and is stored in memory addressed by the CFA (Word 0). Next, the processor sets memory location CFA plus one word equal to zero which indicates the call frame was generated by a CALLM instruction. Base registers 0 through 7 are stored in current frame words 2 through 9 and general purpose registers 2 through 7 are stored current frame words 10 through 15. The processor computes the instruction effective address (5000 in the example) and reads the procedure call address from the effective address location in memory. The procedure call address is loaded into base register 1 and the processor's program counter. The instruction argument pointer, addressed by the instruction RD field (GPR 1 in the example) is transferred from the GPRs to BR3 (5100-hex). The current frame address (41B8-hex) is loaded into BR0 and BR2.

This page intentionally left blank

BRANCH SUBROUTINE 2808

BSUB



830475

DEFINITION

If the RD field is not equal to zero, refer to the CALL instruction format. Base register 2 (BR2) contains a doubleword address that is referred to as the current frame address. The program counter and arithmetic exception bit are stored at the current frame address in main memory (word 0 of the current frame). Word 1 of the current frame is set to '8000 0000'. General purpose register 0 (GPR0) is transferred to BR3 which is called the argument pointer (AP). BRO is set to the current frame address. The contents of the BR specified by RS is transferred to BR1 and the PC portion of the PSD which causes control to be transferred to that location in the program

CONDITION CODE RESULTS

The condition codes remain unchanged.

NOTES

1. Modifications of BR0 will cause unpredictable behavior of the hardware in the sequence.

2. The stack should be bounded on the low end by a write protected or nonexistent page. In this case, a stack overflow will occur on the save of the return PC before the context has been destroyed by the instruction. This will permit error diagnosis to show the cause.

3. Any BR that is to be preserved through the BSUB/RETURN instructions should be stored in the frame at the appropriate location prior to executing the BSUB instruction.

4. An address specification trap will occur if BR2 is not equal to a doubleword address.

Memory Location	4214	
Hexadecimal Instruction:	2818	
Assembly Language Coding:	BSUB B1	RS=1

Before	PSD1	GPR0	00005100	BR0	000041B2
	8B004214	GPR1	FFFFFFFF	BR1	00005000
		GPR2	AAAAAAAA	BR2	000041B8
		GPR3	BBBBBBBB	BR3	33333333
		GPR4	CCCCCCCC	BR4	44444444
		GPR5	DDDDDDDD	BR5	55555555
		GPR6	EEEEEEEE	BR6	66666666
		GPR7	FFFFFFFF	BR7	77777777

Memory Location:	41B8	Don't Care
	41BC	Don't Care
	41C0	FFFFFFFF
	41C4	00000000
	41C8	00000001
	41CC	00000002
	41D0	00000003
	41D4	00000004
	41D8	00000005
	41DC	00000006
	41E0	00000007
	41E4	00000008
	41E8	00000009
	41EC	0000000A
	41F0	0000000B
	41F4	0000000C

After	PSD1	GPR0	00005100	BR0	000041B8	Current Frame
	8B005000	GPR1	FFFFFFFF	BR1	00005000	RS
		GPR2	AAAAAAAA	BR2	000041B8	Current Frame
		GPR3	BBBBBBBB	BR3	00005100	GPR0
		GPR4	CCCCCCCC	BR4	44444444	
		GPR5	DDDDDDDD	BR5	55555555	
		GPR6	EEEEEEEE	BR6	66666666	
		GPR7	FFFFFFFF	BR7	77777777	

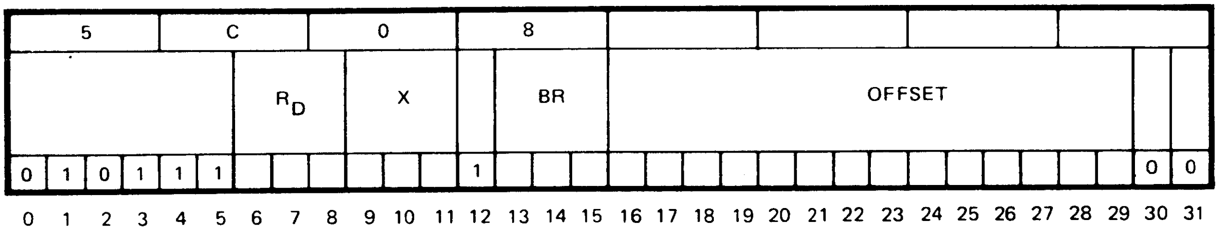
Memory Location:	41B8	01004216	Next PC
	41BC	80000000	Flag Word
	41C0	FFFFFFFF	Unchanged
	41C4	00000000	Unchanged
	41C8	00000001	Unchanged
	41CC	00000002	Unchanged
	41D0	00000003	Unchanged
	41D4	00000004	Unchanged
	41D8	00000005	Unchanged
	41DC	00000006	Unchanged
	41E0	00000007	Unchanged
	41E4	00000008	Unchanged
	41E8	00000009	Unchanged
	41EC	0000000A	Unchanged
	41F0	0000000B	Unchanged
	41F4	0000000C	Unchanged

BSUB EXAMPLE EXPLANATION

If the instruction RD field is not zero, this is a CALL instruction. If the instruction RD field is equal to zero, the processor reads the current frame address (CFA) which must be on a doubleword boundary from BR2. The CFA is copied from BR2 to BR0. Next, the processor computes the address (PC value) of the instruction following the BSUB and the state of the enable/disable arithmetic exception bit (PSD bit 7). The resulting 32 bit word contains zeros in bits 00-06, PSD bit 7 in bit 7, and the PC value in bits 08-30. Bit 31 is not used and is set to zero. The resulting word represents the subroutine return address and is stored in memory which is addressed by the CFA (current frame word 0). Word 1 of the current frame is set to '8000 0000' indicating the call frame was generated by a BSUB instruction. The argument pointer (AP) in GPR0 is transferred to BR3. The instruction procedure call address, addressed by the instruction RS field (BR1 in the example) is transferred to BR1 and the processor's PC counter (5000-hex in the example).

This page intentionally left blank

BRANCH SUBROUTINE MEMORY 5C08 BSUBM



830476

DEFINITION

If the RD field is not equal to zero refer to the CALLM instruction format. Base register 2 (BR2) contains a doubleword address that is referred to as the current frame address (CFA). The program counter (PC) and the enable/disable arithmetic exception bit (PSD bit 7) are stored at the current CFA in main memory (word 0 of the current frame). Word 1 of the current frame is set to '8000 0000'. GPR0 is transferred to BR3 and is called the argument pointer (AP). BR0 is set to the current frame address. The effective word location (EWL) specified by the instruction's effective word address is accessed and its contents transferred to BR1 and the PC portion of the PSD causing control to be transferred to that location in the program.

CONDITION CODE RESULTS

The condition codes remain unchanged.

NOTES

1. Modifications of BR0 will cause unpredictable behavior of the hardware in the sequence.

2. The stack should be bounded on the low end by a write protected or nonexistant page. In this case, a stack overflow will occur on the save of the return PC before the context has been destroyed by the instruction. This will permit error diagnosis to show the cause.

3. Any base register that is to be preserved through the BSUBM/RETURN instructions should be stored in the frame at the appropriate location prior to executing the BSUBM instruction.

4. An address specification trap will occur if BR2 is not equal to a doubleword address.

5. An address specification trap will occur if the instruction effective address is not word aligned.

BASE REGISTER MODE EXAMPLE

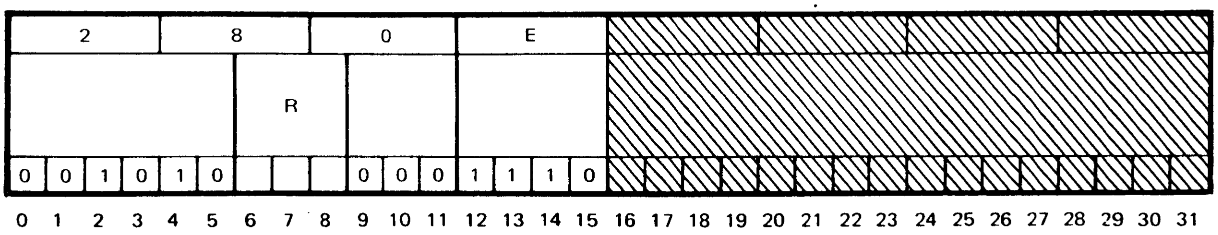
Memory Location:	4214
Hexadecimal Instruction:	5C085000
Assembly Language Coding:	BSUBM X'5000'

Before	PSD1	GPR0 00005100	BR0 000041B2	
	8B004214	GPR1 FFFFFFFF	BR1 00005000	
		GPR2 AAAAAAAA	BR2 000041B8	
		GPR3 BBBBBBBB	BR3 33333333	
		GPR4 CCCCCCCC	BR4 44444444	
		GPR5 DDDDDDDD	BR5 55555555	
		GPR6 EEEEEEEE	BR6 66666666	
		GPR7 FFFFFFFF	BR7 77777777	
	Memory Location:	41B8	Don't Care	
		41BC	Don't Care	
		41C0	Don't Care	
		.	.	
		.	.	
		.	.	
		41F4	Don't Care	
		5000	00005080 Proc. Address	
After	PSD1	GPR0 00005100	BR0 010041B8	CFA
	8B005080	GPR1 FFFFFFFF	BR1 00005080	Proc. Address
		GPR2 AAAAAAAA	BR2 000041B8	CFA
		GPR3 BBBBBBBB	BR3 00005100	GPR0
		GPR4 CCCCCCCC	BR4 44444444	
		GPR5 DDDDDDDD	BR5 55555555	
		GPR6 EEEEEEEE	BR6 66666666	
		GPR7 FFFFFFFF	BR7 77777777	
	Memory Location:	41B8	01004218	Next PC
		41BC	80000000	Flag Word
		41C0	Unchanged	
		.	.	
		.	.	
		.	.	
		41F4	Unchanged	
		5000	00005080	Unchanged

BSUBM EXAMPLE EXPLANATION

If the instruction RD field is not zero, this is a CALLM instruction. If the instruction RD field is equal to zero, the processor reads the current frame address (CFA) which must be on a doubleword boundary from base register 2 (BR2). The CFA is copied from BR2 to BR0. Next, the processor computes the address (PC value) of the instruction following the BSUBM and the state of the enable/disable arithmetic exception bit (PSD bit 7). The resulting 32 bit word contains zeros in bits 00-06, PSD bit 7 in bit 7, and the PC value in bits 08-30. Bit 31 is not used and is set to zero. The resulting word represents the subroutine return address and is stored in memory address by the CFA (current frame word 0). The second word of the current frame is set to '8000 0000' indicating the call frame was generated by a BSUBM instruction. The instruction argument pointer in GPR0 is transferred to BR3. The processor computes the instruction effective address (5000 hex in the example) and reads the procedure call address from the effective address location in memory. The procedure call address is loaded into BR1 and the processor's PC (5080 hex).

This page intentionally left blank



830311

This instruction is used to exit the current procedure and return control to the Caller. The contents of base register 0 (B0) is used as the pointer to the frame. All base registers and general purpose registers 2 to 7 are reloaded from the call frame if bit 0 in the second word of the call frame is zero. If bit 0 is set only the base registers are returned. The return PC, as saved in the call frame, is loaded into the PC portion of the PSD. This includes setting the arithmetic exeception bit as it was at the time of the call. The restoring of the registers has the effect of resetting the stack to the point of call. The contents of general purpose register 0 and 1 (R0, R1) and the condition codes are not changed by this instruction. That is, they are the same as at the point of exit from the procedure.

CONDITION CODE RESULTS

The condition codes are not affected by this instruction.

NOTES

1. If a return is issued with base register 0 containing the address of anything other than a valid call frame, the behavior of this instruction is unpredictable.

2. If the call frame word 1, flag word, is equal to hex 8000 0000, then software must insure the call frame contains valid base registers since the BSUB or BSUBM instructions do not store the base registers.

3. If an operand address generated by this instruction crosses a MAP block boundary, an address specification trap will occur. This applies when in either the mapped or unmapped mode.

BASE REGISTER MODE EXAMPLE 1

```
Memory Location:                   5080
Hexadecimal Instruction:           280E
Assembly Language Coding:          RETURN

Before
PSD1      GPR0 13579BDF   BR0 000041B8   Memory Location
8A005080  GPR1 FDB97531   BR1 00005080   41B8  01004216   Return Address
          GPR2 XXXXXXXX   BR2 000041A0   41BC  00000000   Flag word
          GPR3 XXXXXXXX   BR3 00005100   41C0  000041B2   BR0
          GPR4 XXXXXXXX   BR4 XXXXXXXX   41C4  00005000   BR1
          GPR5 XXXXXXXX   BR5 XXXXXXXX   41C8  000041FC   BR2
          GPR6 XXXXXXXX   BR6 XXXXXXXX   41CC  00005080   BR3
          GPR7 XXXXXXXX   BR7 XXXXXXXX   41D0  44444444   BR4
                                         41D4  55555555   BR5
After     GPR0 13579BDF   BR0 000041B2   41D8  66666666   BR6
PSD1      GPR1 FDB97531   BR1 00005000   41DC  77777777   BR7
8B004216  GPR2 AAAAAAAA   BR2 000041FC   41E0  AAAAAAAA   GPR2
          GPR3 BBBBBBBB   BR3 00005080   41E4  BBBBBBBB   GPR3
          GPR4 CCCCCCCC   BR4 44444444   41E8  CCCCCCCC   GPR4
          GPR5 DDDDDDDD   BR5 55555555   41EC  DDDDDDDD   GPR5
          GPR6 EEEEEEEE   BR6 66666666   41F0  EEEEEEEE   GPR6
          GPR7 FFFFFFFF   BR7 77777777   41F4  FFFFFFFF   GPR7
                                         (These memory locations are
                                         unchanged by the Return
                                         instruction)
```

BASE REGISTER MODE EXAMPLE 2

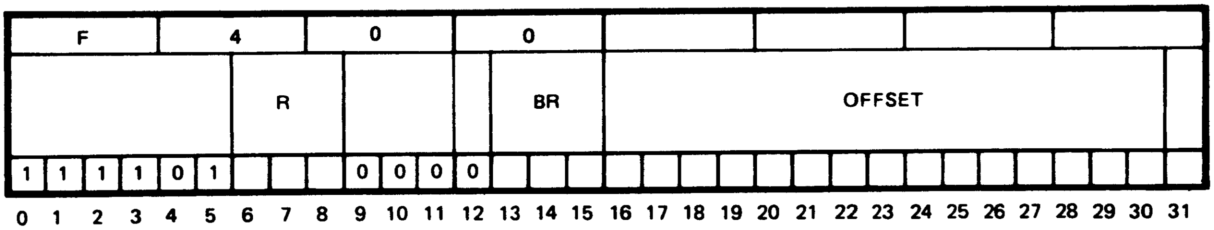
```
Memory Location:                   5080
Hexadecimal Instruction:           280E
Assembly Language Coding:          RETURN

Before
PSD1      GPR0 XXXXXXXX   BR0 000041B8   Memory Location
8A005080  GPR1 XXXXXXXX   BR1 00005080   41B8  01004216   Return Address
          GPR2 XXXXXXXX   BR2 000041B8   41BC  80000000   Flag word
          GPR3 XXXXXXXX   BR3 00005100   41C0  000041B2   BR0
          GPR4 XXXXXXXX   BR4 XXXXXXXX   41C4  00005000   BR1
          GPR5 XXXXXXXX   BR5 XXXXXXXX   41C8  000041FC   BR2
          GPR6 XXXXXXXX   BR6 XXXXXXXX   41CC  33333333   BR3
          GPR7 XXXXXXXX   BR7 XXXXXXXX   41D0  44444444   BR4
                                         41D4  55555555   BR5
After     GPR0 XXXXXXXX   BR0 000041B2   41D8  66666666   BR6
PSD1      GRP1 XXXXXXXX   BR1 00005000   41DC  77777777   BR7
8B004216  GPR2 XXXXXXXX   BR2 000041FC
          GPR3 XXXXXXXX   BR3 33333333   (These memory locations are
          GPR4 XXXXXXXX   BR4 44444444   unchanged by the Return
          GPR5 XXXXXXXX   BR5 55555555   instruction)
          GPR6 XXXXXXXX   BR6 66666666
          GRP7 XXXXXXXX   BR7 77777777
```

PROCEDURE RETURN EXAMPLE EXPLANATION

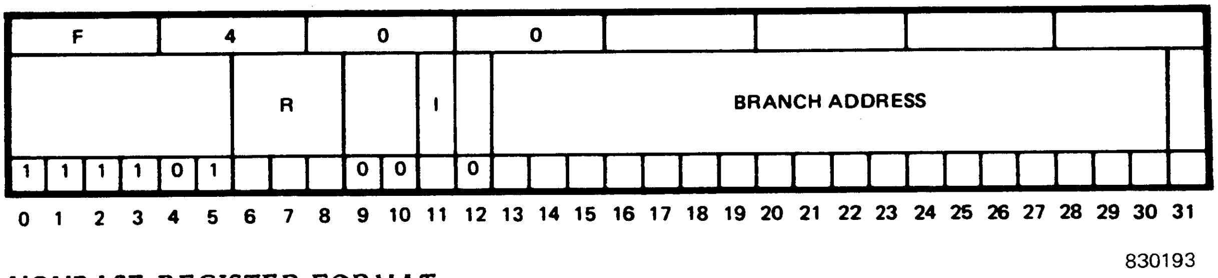
The contents of base register 0 (41B8) are used as the pointer to the current frame address (CFA). The second word of the current frame (41BC) is tested and if bit 0 is zero, as it is in the first example, base registers 0-7 and general purpose registers 2-7 are reloaded from the current frame, words 2-15 (locations 41C0-41F4). If the second word of the current frame (location 41BC) has bit 0 set, as it is in the second example, base registers 0-7 are reloaded from the current frame, words 2-9 (locations 41C0-41DC). The first word of the current frame (location 41B8) is used as the return address, and is loaded into PSD1. This includes setting the arithmetic exception bit as it was at the time of the call. Current condition codes and general purpose registers 1-2 are not changed by this instruction.

This page intentionally left blank

BRANCH AFTER INCREMENTING BY A BYTE — BIB
F400 — d,*m



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

DEFINITION

The contents of the general purpose register (GPR) specified by R are incremented in bit position 31. If the result is nonzero, the effective address (EA) is transferred to the PC field of the program status doubleword (PSD). If the result is equal to zero after incrementing, the next instruction is executed. In either case, the condition codes are unchanged.

SUMMARY EXPRESSION

$(R) + 1 \rightarrow R$

If result $\neq 0$

EA → PSD_{13-30}	Nonbase register format	(Assumes no pre-indexing. Indirect addressing changes condition codes).
EA → PSD_{08-30}	Base register format	

CONDITION CODE RESULTS

CC1:	No change
CC2:	No change
CC3:	No change
CC4:	No change

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of the GPR0 are incremented by one at bit position 31. Since the result is zero, no branch occurs.

Memory Location:	1B204
Hexadecimal Instruction:	F406B100 (R=0, BR=6)
Assembly Language Coding:	BIB 0,X'B100'(6)

Before	PSD1	GPR0	BR6
	2201B204	FFFFFFFF	000000A8
After	PSD1	GPR0	BR6
	2201B208	00000000	000000A8

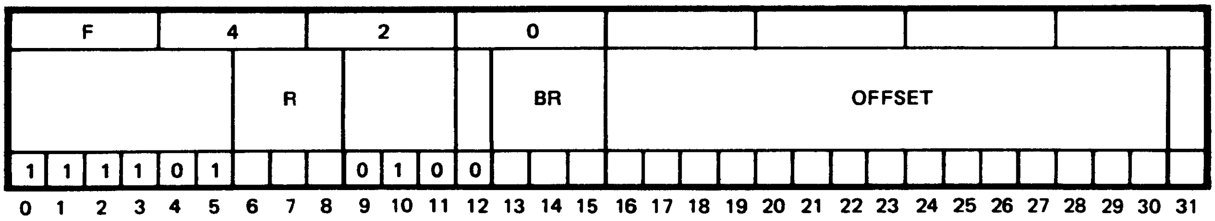
NONBASE REGISTER MODE EXAMPLE

The contents of the GPR0 are incremented by one at bit position 31. Since the result is zero, no branch occurs. Indexing is not allowed.

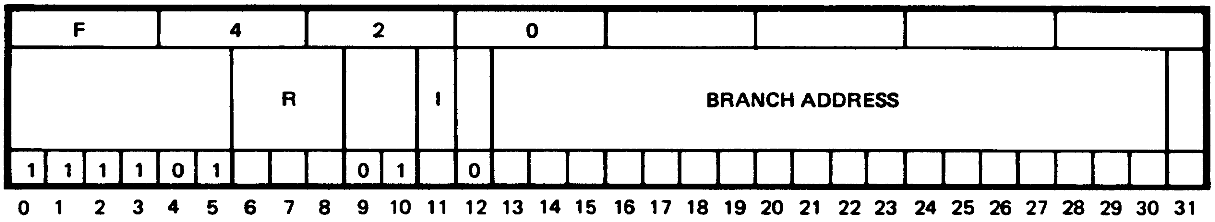
If the indirect bit of the instruction word is set (I=1) when the branch occurs, bit positions 1-4 of the last memory word in the indirect chain are transferred to the corresponding bit positions of the PSD. Bits 0 and 5-12 are unchanged.

Memory Location:	1B204
Hexadecimal Instruction:	F4 01 B1 A8 (R=0, I=0)
Assembly Language Coding:	BIB 0,X'1B1A8'

Before	PSD1	GPR0
	2001B204	FFFFFFFF
After	PSD1	GPR0
	2001B208	00000000



BASE REGISTER FORMAT



830194

NONBASE REGISTER FORMAT

DEFINITION

The contents of the general purpose register (GPR) specified by R are incremented in bit position 30. If the result is nonzero, the effective address (EA) is transferred to the PC field of the program status doubleword (PSD), and the condition codes remain unchanged. If the result is equal to zero after incrementing, the next instruction is executed.

SUMMARY EXPRESSION

$(R) + 2 \rightarrow R$

If result $\neq 0$

$EA \rightarrow PSD_{13-30}$ Nonbase register format
$EA \rightarrow PSD_{08-30}$ Base register format

(Assumes no pre-indexing. Indirect addressing changes condition codes).

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of GPR2 are incremented by one in bit position 30. The result is replaced in GPR2 and a branch occurs to address 003948.

Memory Location:	039A0
Hexadecimal Instruction:	F5263900 (R=2, BR=6)
Assembly Language Coding:	BIH 2, X'3900'(6)

Before	PSD1 120039A0	GPR2 FFFFD72A	BR6 00000048
After	PSD1 12003948	GPR2 FFFFD72C	BR6 00000048

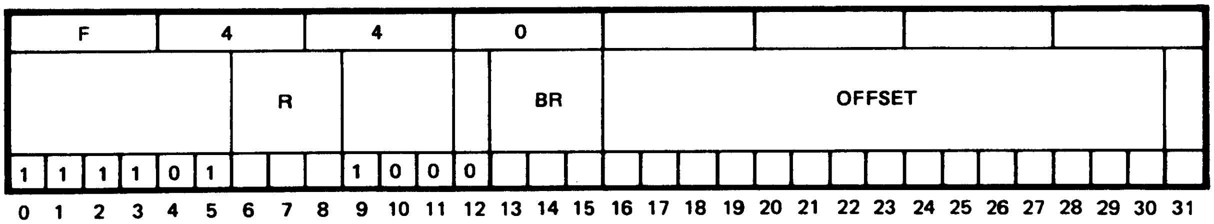
NONBASE REGISTER MODE EXAMPLE

The contents of GPR2 are incremented by one in bit position 30. The result is replaced in GPR2 and a branch occurs to address 03948. Indexing is not allowed.

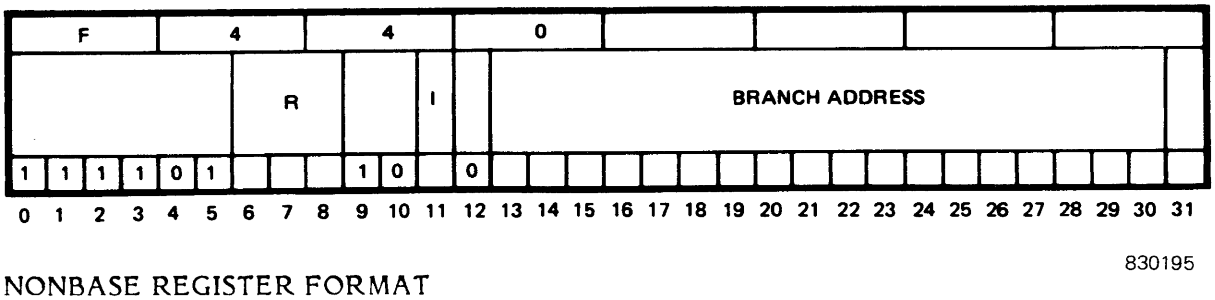
If the indirect bit of the instruction word is equal to one, and the branch occurs, bit positions 1-4 of the last memory word in the indirect chain are transferred to the corresponding bit positions of the PSD. Bits 0 and 5-12 are unchanged.

Memory Location:	039A0
Hexadecimal Instruction:	F5 20 39 48 (R=2, I=0)
Assembly Language Coding:	BIH 2,X'3948'

Before	PSD1 100039A0	GPR2 FFFFD72A
After	PSD1 10003948	GPR2 FFFFD72C



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

DEFINITION

The contents of the general purpose register (GPR) specified by R are incremented in bit position 29. If the result is nonzero, the effective address (EA) is transferred to the PC field of the program status doubleword (PSD), and the condition codes remain unchanged. If the result is equal to zero after incrementing, the next instruction is executed.

SUMMARY EXPRESSION

$(R) + 4 \rightarrow R$

If result $\neq 0$

$EA \rightarrow PSD_{13-30}$	Nonbase register format	(Assumes no pre-indexing. Indirect addressing changes condition codes).
$EA \rightarrow PSD_{08-30}$	Base register format	

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of GPR6 are incremented by one at bit position 29, and the result is transferred to GPR6. The effective address of the BIW instruction (004B2C), replaces the previous contents of the PSD, bits 08-30.

Memory Location:	04A38
Hexadecimal Instruction:	F7464B00 (R=6, BR=6)
Assembly Language Coding:	BIW 6,X'4B00' (6)

Before	PSD1	GPR6	BR6
	62004A38	FFFFDC18	0000002C
After	PSD1	GPR6	BR6
	62004B2C	FFFFDC1C	0000002C

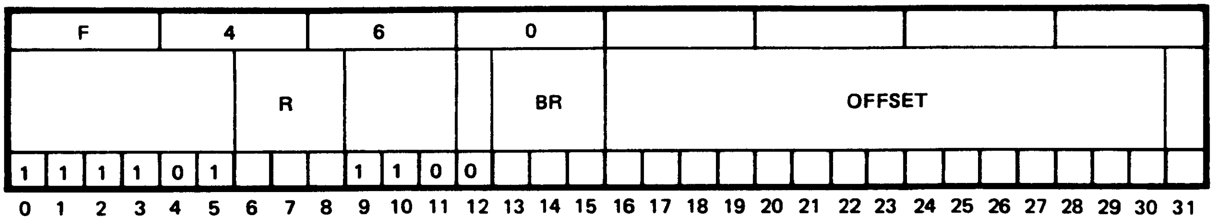
NONBASE REGISTER MODE EXAMPLE

The contents of GPR6 are incremented by one at bit position 29, and the result is transferred to GPR6. The effective address of the BIW instruction (04B2C), replaces the previous contents of the PSD, bits 12-30. Indexing is not allowed.

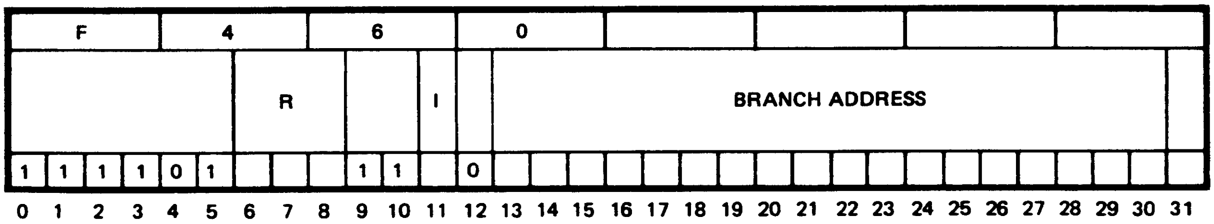
If the indirect bit of the instruction word is set (I=1) when branch occurs, bit positions 1-4 of the last memory word in the direct chain are transferred to the corresponding bit positions of the PSD. Bits 0 and 5-12 are unchanged.

Memory Location:	04A38
Hexadecimal Instruction:	F7 40 4B 2C (R=6, I=0)
Assembly Language Coding:	BIW 6,X'4B2C'

Before	PSD1	GPR6
	6000A438	FFFFDC18
After	PSD1	GPR6
	60004B2C	FFFFDC1C



BASE REGISTER FORMAT



830196

NONBASE REGISTER FORMAT

DEFINITION

The contents of the general purpose register (GPR) specified by R are incremented in bit position 28. If the result is nonzero, the effective address (EA) is transferred to the PC field of the program status doubleword (PSD), and the condition codes remain unchanged. If the result is equal to zero after incrementing, the next instruction is executed.

SUMMARY EXPRESSION

$(R) + 8 \rightarrow R$

If the result $\neq 0$

EA → PSD_{13-30}	Nonbase Register Format	(Assumes no pre-indexing. Indirect addressing change condition codes).
EA → PSD_{08-30}	Base Register Format	

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of GPR3 are incremented by one at bit position 28 and replaced. Since the result is zero, no branch occurs.

Memory Location:	0930C
Hexadecimal Instruction:	F5E69106 (R=3, BR=6)
Assembly Language Coding:	BID 3,'9106'(6)

Before	PSD1 0A00930C	GPR3 FFFFFFF8	BR6 000000A0
After	PSD1 0A009310	GPR3 00000000	BR6 000000A0

NONBASE REGISTER MODE EXAMPLE

The contents of GPR3 are incremented by one at bit position 28 and replaced. Since the result is zero, no branch occurs. Indexing is not allowed.

If the indirect bit of the instruction word is set (I=1) when the branch occurs, bit positions 1-4 of the last memory word in the direct chain are transferred to the corresponding bit positions of the PSD. Bits 0 and 5-12 are unchanged.

Memory Location:	0930C
Hexadecimal Instruction:	F5 E0 91 A6 (R=3, I=0)
Assembly Language Coding:	BID 3,X'91A6'

Before	PSD1 0800930C	GPR3 FFFFFFF8
After	PSD1 08009310	GPR3 00000000

This page intentionally left blank

6.2.5 Compare Instructions

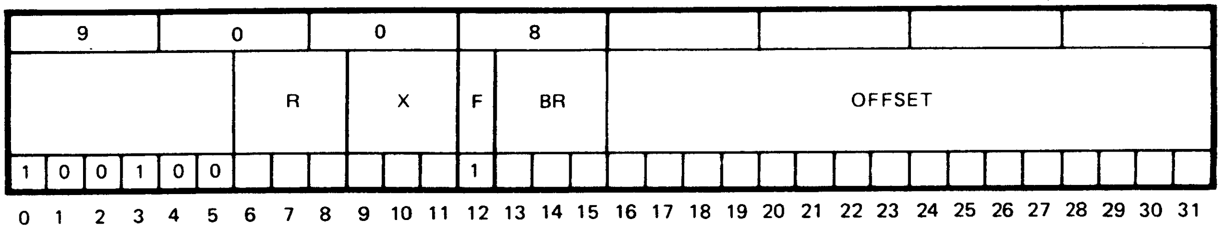
Compare instructions provide the capability of comparing the data in memory and in the general purpose registers. These compare operations can be performed on bytes, halfwords, words, or doublewords. Provisions have been made to allow the result of compare operations to be masked with the contents of the mask register before final testing.

6.2.5.1 Instruction Format

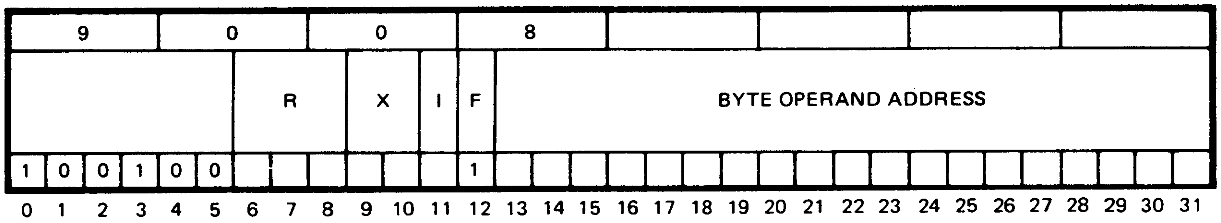
The compare instruction group uses the standard memory reference, immediate, and interregister formats.

6.2.5.2 Condition Code

A condition code is set during most compare instructions to indicate whether the operation produced a result greater than, less than, or equal to zero.



BASE REGISTER FORMAT



830197

NONBASE REGISTER FORMAT

DEFINITION

The contents of the effective byte location (EBL) specified by the effective byte address (EBA) is accessed, right justified, and subtracted algebraically from the word in the general purpose register (GPR) specified by R. The result of the subtraction causes one of the condition code bits (2-4) to be set. The contents of the GPR specified by R and the byte specified by the EBA remain unchanged.

SUMMARY EXPRESSION

$(R) - (EBL) \rightarrow SCC_{2-4}$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R) is greater than (EBL)
CC3: Is set if (R) is less than (EBL)
CC4: Is set if (R) is equal to (EBL)

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. CC3 is set indicating that the contents of GPR1 are less than the contents of memory byte 0010B5.

Memory Location:	01000
Hexadecimal Instruction:	908E1000 (R=1, X=0, BR=6)
Assembly Language Coding:	CAMB 1,X'1000'(6)

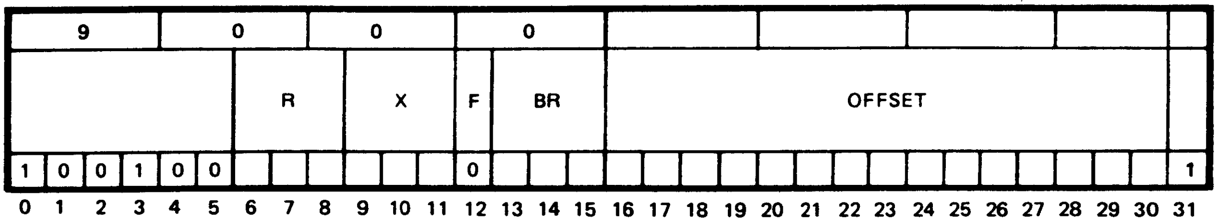
Before	PSD1	GPR1	BR6	Memory Byte 0010B5
	0A001000	000000B6	000000B5	C7
After	PSD1	GPR1	BR6	Memory Byte 0010B5
	12001004	000000B6	000000B5	C7

NONBASE REGISTER MODE EXAMPLE

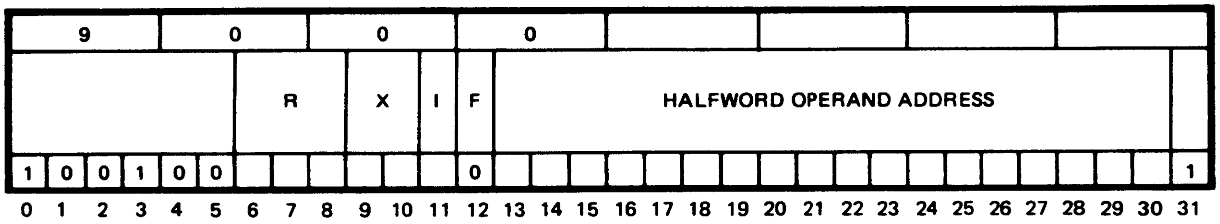
CC3 is set indicating that the contents of GPR1 are less than the contents of memory byte 010B5.

Memory Location:	01000
Hexadecimal Instruction:	90 88 10 B5 (R=1, X=0, I=0)
Assembly Language Coding:	CAMB 1,X'10B5'

Before	PSD1	GPR1	Memory Byte 010B5
	08001000	000000B6	C7
After	PSD1	GPR1	Memory Byte 010B5
	10010004	000000B6	C7



BASE REGISTER FORMAT



830198

NONBASE REGISTER FORMAT

DEFINITION

The contents of the effective halfword location (EHL) specified by the effective halfword address (EHA) is accessed, and the sign bit is extended 16 bits to the left to form a word. The resulting word is subtracted algebraically from the word in the general purpose register (GPR) specified by R. The result of the subtraction causes one of the condition code bits (2-4) to be set. The word in the GPR specified by R and the halfword specified by the EHA remain unchanged.

SUMMARY EXPRESSION

$(R) - (EHL)_{SE} \rightarrow SCC_{2-4}$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R) is greater than $(EHL)_{SE}$
CC3: Is set if (R) is less than $(EHL)_{SE}$
CC4: Is set if (R) is equal to $(EHL)_{SE}$

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. CC2 is set indicating that the contents of GPR4 are greater than the contents of memory halfword 003976 (a negative value).

Memory Location:	0379C
Hexadecimal Instruction:	92063901 (R=4, X=0, BR=6)
Assembly Language Coding:	CAMH 4,X'3900'(6)

Before	PSD1	GPR4	BR6	Memory Halfword 003976
	0A00379C	00008540	00000076	8640
After	PSD1	GPR4	BR6	Memory Halfword 003976
	220037A0	00008540	00000076	8640

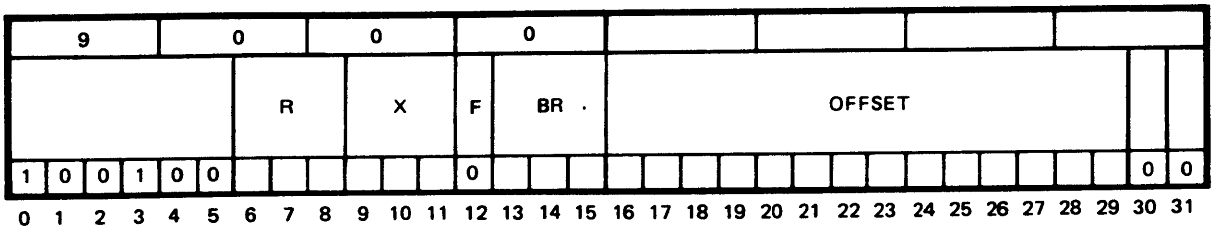
NONBASE REGISTER MODE EXAMPLE

CC2 is set indicating that the contents of GPR4 are greater than the contents of memory halfword 03976 (a negative value).

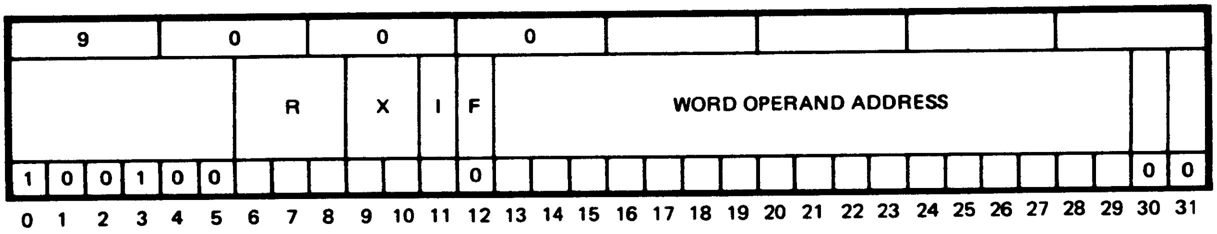
Memory Location:	0379C
Hexadecimal Instruction:	92 00 39 77 (R=4, X=0, I=0)
Assembly Language Coding:	CAMH 4,X'3976'

Before	PSD1	GPR4	Memory Halfword 03976
	0800379C	00008540	8640
After	PSD1	GPR4	Memory Halfword 03976
	200037A0	00008540	8640

COMPARE ARITHMETIC WITH MEMORY WORD 9000 — CAMW d,*m,x



BASE REGISTER FORMAT



830199

NONBASE REGISTER FORMAT

DEFINITION

The contents of the effective word location (EWL) specified by the effective word address (EWA) is accessed and subtracted algebraically from the word in the general purpose register (GPR) specified by R. The result of the subtraction causes one of the condition code bits (2-4) to be set. The word in the GPR specified by R and the word specified by the EWA remain unchanged.

SUMMARY EXPRESSION

$(R) - (EWL) \rightarrow SCC_{2-4}$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R) is greater than (EWL)
CC3: Is set if (R) is less than (EWL)
CC4: Is set if (R) is equal to (EWL)

BASE REGISTER MODE EXAMPLE

The contents of GPR4, BR6 and the instruction offset are added to obtain the logical address. CC3 is set to indicate that the contents of GPR6 are less than the contents of memory word 005C78.

Memory Location:	05820
Hexadecimal Instruction:	93465C00 (R=6, BR=6, X=4)
Assembly Language Coding:	CAMW 6, X '5C00' (6), 4

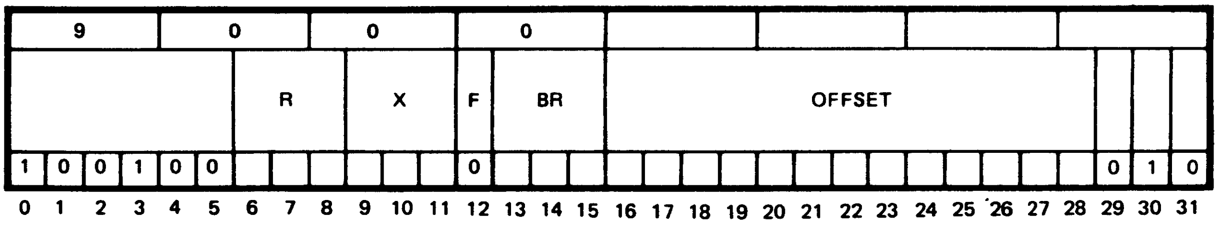
Before	PSD1	GPR6	GPR4	BR6	Memory Word 005C78
	42005B20	9E03B651	00000008	00000070	A184F207
After	PSD1	GPR6	GPR4	BR6	Memory Word 005C78
	12005B24	9E03B651	00000008	00000070	A184F207

NONBASE REGISTER MODE EXAMPLE

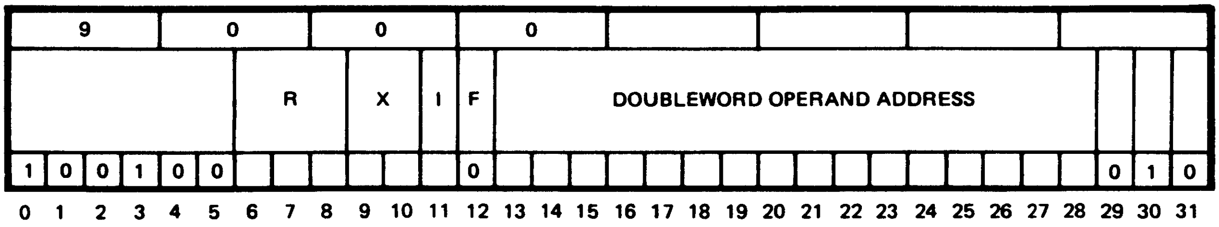
CC3 is set indicating that the contents of the GPR6 are less than the contents of memory word 05C78.

Memory Location:	05B20
Hexadecimal Instruction:	93 00 5C 78 (R=6, X=0, I=0)
Assembly Language Coding:	CAMW 6,X'5C78'

Before	PSD1	GPR6	Memory Word 05C78
	40005B20	9E03B651	A184F207
After	PSD1	GPR6	Memory Word 05C78
	10005B24	9E03B651	A184F207



BASE REGISTER FORMAT



830200

NONBASE REGISTER FORMAT

DEFINITION

The contents of the effective doubleword location (EDL) specified by the effective doubleword address (EDA) is accessed and subtracted algebraically from the doubleword in the general purpose register (GPR) specified by R and R+1. R+1 is the GPR one greater than specified by R. The result of the subtraction causes one of the condition code bits (2-4) to be set. The doubleword in the GPR specified by R and R+1, and the doubleword specified by the EDA, remain unchanged.

NOTE

The GPR specified by R must be an even-numbered register.

SUMMARY EXPRESSION

$(R, R+1) - (EDL) \rightarrow SCC_{2-4}$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R, R+1) is greater than (EDL)
CC3: Is set if (R, R+1) is less than (EDL)
CC4: Is set if (R, R+1) is equal to (EDL)

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. CC4 is set indicating that the doubleword obtained from GPR4 and GPR5 is equal to that obtained from the memory words 007F50 and 007F54.

Memory Location:	27C14
Hexadecimal Instruction:	92067F02 (R=4, X=0, BR=6)
Assembly Language Coding:	CAMD 4,X'7F00'(6)

Before	PSD1	GPR4	GPR5	BR6
	22027C14	7AE0156D	47B39208	00000050
	Memory Word 007F50			Memory Word 007F54
	7AE0156D			47B39208
After	PSD1	GPR4	GPR5	BR6
	0A027C18	7AE0156D	47B39208	00000050
	Memory Word 007F50			Memory Word 007F54
	7AE0156D			47B39208

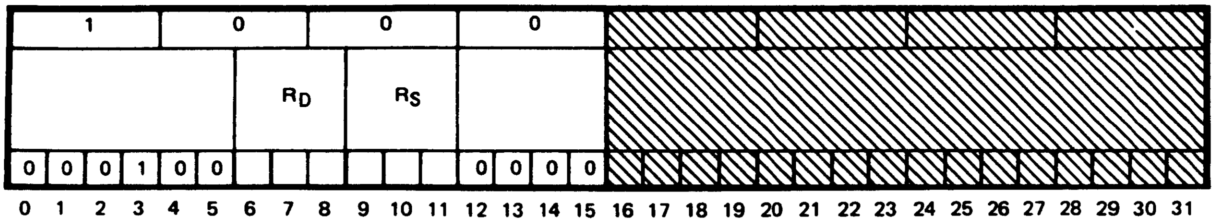
NONBASE REGISTER MODE EXAMPLE

CC4 is set indicating that the doubleword obtained from GPR4 and GPR5 is equal to that obtained from the memory words 27F50 and 27F54.

Memory Location:	27C14
Hexadecimal Instruction:	92 02 7F 52 (R=4, X=0, I=0)
Assembly Language Coding:	CAMD 4,X'27F50'

Before	PSD1	GPR4	GPR5
	20027C14	7AE0156D	47B39208
	Memory Word 27F50		Memory Word 27F54
	7AE0156D		47B39208
After	PSD1	GPR4	GPR5
	08027C18	7AE0156D	47B39208
	Memory Word 27F50		Memory Word 27F54
	7AE0156D		47B39208

COMPARE ARITHMETIC WITH REGISTER CAR
1000 s,d



830201

DEFINITION

The word in the general purpose register (GPR) specified by R_S is subtracted algebraically from the word in the GPR specified by R_D. The result of the subtraction causes one of the condition code bits (2-4) to be set. The words specified by R_S and R_D remain unchanged.

SUMMARY EXPRESSION

$(R_D) - (R_S) \rightarrow SCC_{2-4}$

CONDITION CODE RESULTS

CC1: Always zero

CC2: Is set if (R_D) is greater than (R_S)

CC3: Is set if (R_D) is less than (R_S)

CC4: Is set if (R_D) is equal to (R_S)

NONBASE AND BASE REGISTER MODE EXAMPLE

CC3 is set indicating that the contents of GPR0 are less than the contents of GPR1.

Memory Location:	0B3C2
Hexadecimal Instruction:	10 10 (R_D=0,R_S=1)
Assembly Language Coding:	CAR 1,0

Before	PSD1		GPR0	GPR1
	0800B3C2	(Nonbase)	58DF620A	6A92B730
	0A00B3C2	(Base)		
After	PSD1		GPR0	GPR1
	1000B3C4	(Nonbase)	58DF620A	6A92B730
	1200B3C4	(Base)		

<table>
<tr><td colspan="4">C</td><td colspan="4">8</td><td colspan="4">0</td><td colspan="4">5</td><td colspan="4"></td><td colspan="4"></td><td colspan="4"></td><td colspan="4"></td></tr>
<tr><td colspan="6"></td><td colspan="3">R</td><td colspan="3"></td><td colspan="4"></td><td colspan="16">IMMEDIATE OPERAND</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td></td><td></td><td></td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
</table>

830202

DEFINITION

The sign bit (bit 16) of the immediate operand is extended 16 bit positions to the left to form a word. This word is subtracted from the word in the general purpose register (GPR) specified by R. The result of the subtraction causes one of the condition code bits (2-4) to be set. The word in the GPR specified by R and the immediate operand (bit 16-31) remain unchanged.

SUMMARY EXPRESSION

$$(R) - (IW_{16-31})_{SE} \rightarrow SCC_{2-4}$$

CONDITION CODE RESULTS

CC1: Always zero

CC2: Is set if (R) is greater than $(IW_{16-31})_{SE}$

CC3: Is set if (R) is less than $(IW_{16-31})_{SE}$

CC4: Is set if (R) is equal to $(IW_{16-31})_{SE}$

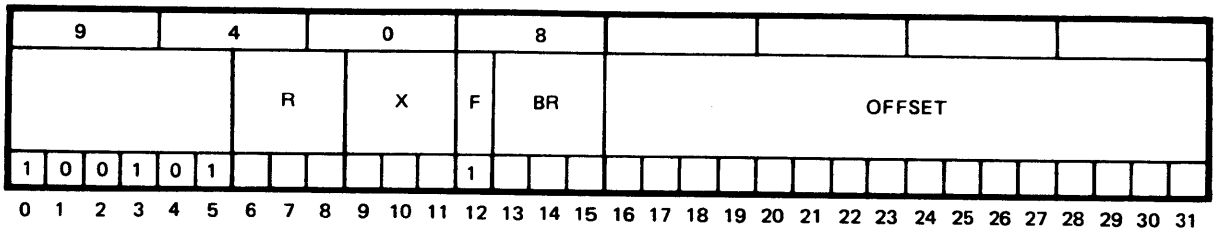
NONBASE AND BASE REGISTER MODE EXAMPLE

CC3 is set indicating that the contents of GPR1 are less than the immediate operand.

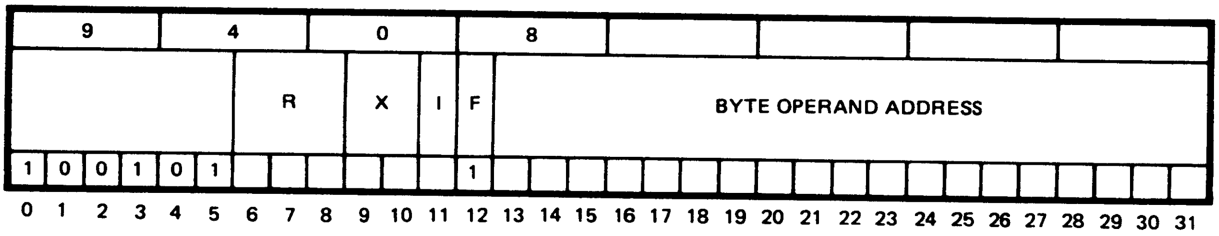
Memory Location:	0A794
Hexadecimal Instruction:	C8 85 71 A2 (R=1)
Assembly Language Coding:	CI 1,X'71A2'

Before	PSD1		GPR1
	4000A794	(Nonbase)	00005719
	4200A794	(Base)	
After	PSD1		GPR1
	1000A798	(Nonbase)	00005719
	1200A798	(Base)	

COMPARE MASKED WITH MEMORY BYTE 9408 — CMMB d,*m,x



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

830203

DEFINITION

The contents of the effective byte location (EBL) specified by the effective byte address (EBA) is accessed, and 24 zeros are appended to the most-significant end to form a word. This word is logically compared (exclusive OR function) with the word in the general purpose register (GPR) specified by R. The resulting word is then masked (logical AND function) with the contents of the mask register (R4). The masked result is tested and condition code bit 4 is set if all 32 bits equal zero. The word in the GPR specified by R and the byte specified by the EBA remain unchanged.

SUMMARY EXPRESSION

$(R) \oplus \text{Zeros}_{0\text{-}23}, (EBL)\ \&\ (R4) \rightarrow SCC_4$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Always zero
CC3: Always zero
CC4: Is set if the result is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of GPR0 and memory byte 000917 are identical in those bit positions specified by the contents of GPR4. CC4 is set.

Memory Location:	00800
Hexadecimal Instruction:	940E0900 (R=0, X=0, BR=6)
Assembly Language Coding:	CMMB 0,X'0900'(6)

	PSD1	GPR0	GPR4	BR6	Memory Byte 000917
Before	12000800	000000A1	000000F0	00000017	A9
After	0A000804	000000A1	000000F0	00000017	A9

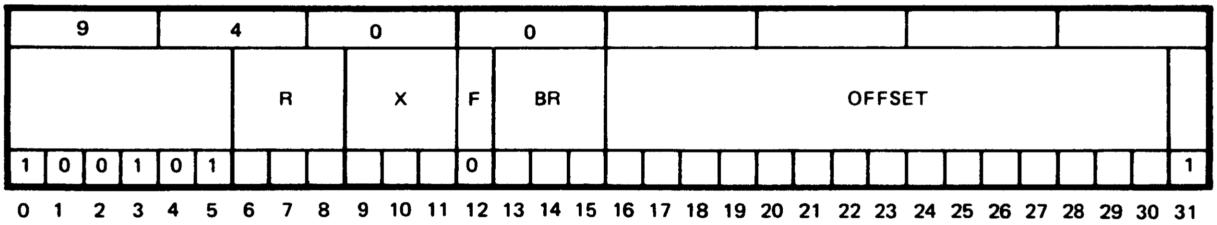
NONBASE REGISTER MODE EXAMPLE

The contents of GPR0 and memory byte 00917 are identical in those bit positions specified by the contents of GPR4. CC4 is set.

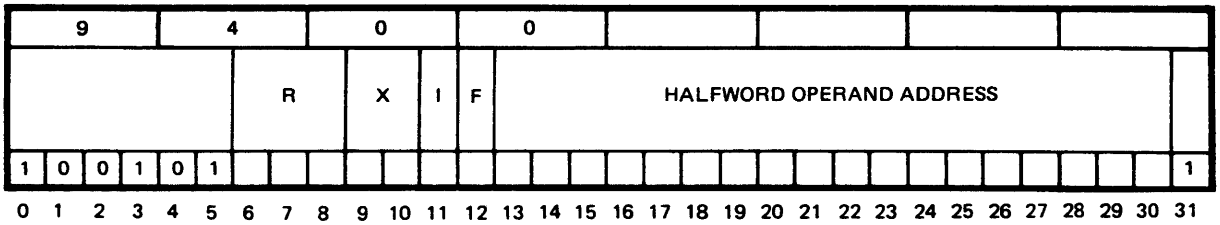
Memory Location:	00800
Hexadecimal Instruction:	94 08 09 17 (R=0, X=0, I=0)
Assembly Language Coding:	CMMB 0,X'917'

	PSD1	GPR0	GPR4	Memory Byte 00917
Before	10000800	000000A1	000000F0	A9
After	08000804	000000A1	000000F0	A9

COMPARE MASKED WITH MEMORY HALFWORD 9400 — CMMH d,*m,x



BASE REGISTER FORMAT



830204

NONBASE REGISTER FORMAT

DEFINITION

The contents of the effective halfword location (EHL) specified by the effective halfword address (EHA) is accessed, and the sign (bit 16) is extended 16 bits to the left to form a word. The resulting word is logically compared (exclusive OR function) with the word in the general purpose register (GPR) specified by R. The resulting word is then masked (logical AND function) with the contents of the mask register (R4). The masked result is tested and condition code bit 4 is set if all 32 bits equal zero. The word in the GPR specified by R and the halfword specified by the EHA remain unchanged.

SUMMARY EXPRESSION

$(R) \oplus (EHL)_{SE} \ \& \ (R4) \rightarrow SCC_4$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Always zero
CC3: Always zero
CC4: Is set if the result is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of GPR2 and memory halfword 006292 are identical in those bit positions specified by the contents of GPR4. CC4 is set.

Memory Location:	061B8
Hexadecimal Instruction:	95066201 (R=2, X=0, BR=6)
Assembly Language Coding:	CMMH 2,X'6200'(6)

Before	PSD1	GPR2	GPR4	BR6	Memory Halfword 006292
	120061B8	09A043B6	00004284	00000092	46FC
After	PSD1	GPR2	GPR4	BR6	Memory Halfword 006292
	0A0061BC	09A043B6	00004284	00000092	46FC

NONBASE REGISTER MODE EXAMPLE

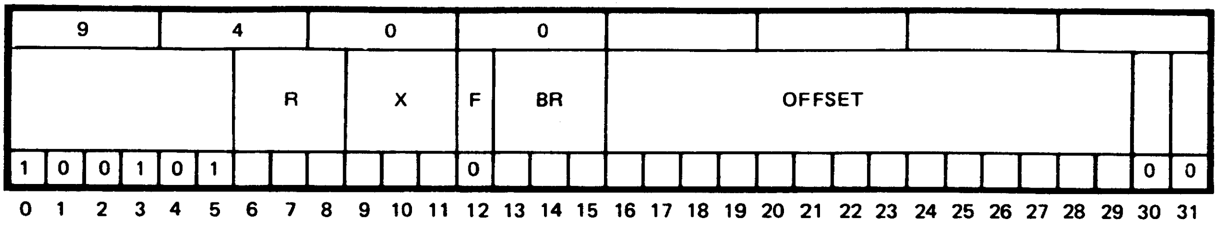
The contents of GPR2 and memory halfword 06292 are identical in those bit positions specified by the contents of GPR4. CC4 is set.

Memory Location:	061B8
Hexadecimal Instruction:	95 00 62 93 (R=2, X=0, I=0)
Assembly Language Coding:	CMMH 2,X'6292'

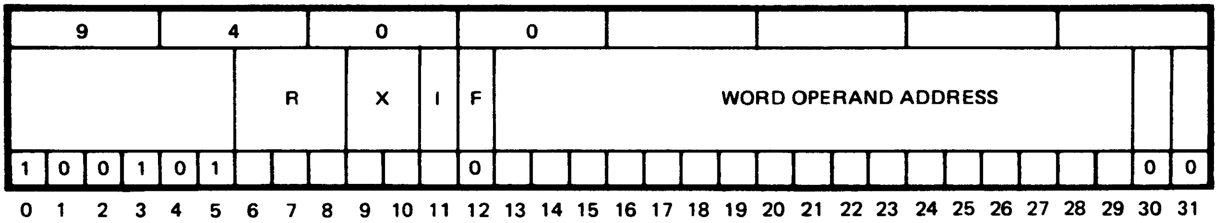
Before	PSD1	GPR2	GPR4	Memory Halfword 06292
	100061B8	09A043B6	00004284	46FC
After	PSD1	GPR2	GPR4	Memory Halfword 06292
	080061BC	09A043B6	00004284	46FC

COMPARE MASKED WITH MEMORY WORD 9400

CMMW d,*m,x



BASE REGISTER FORMAT



830205

NONBASE REGISTER FORMAT

DEFINITION

The contents of the effective word location (EWL) specified by the effective word address (EWA) is accessed and logically compared (exclusive OR function) with the word in the general purpose register (GPR) specified by R. The result of the comparison is then masked (logical AND function) with the contents of the mask register (R4). The masked result is tested and condition code bit 4 is set if all 32 bits equal zero. The word in the GPR specified by R and the word specified by the EWA remain unchanged.

SUMMARY EXPRESSION

$(R) \oplus (EWL) \ \& \ (R4) \rightarrow SCC_4$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Always zero
CC3: Always zero
CC4: Is set if the result is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of GPR6 and memory word 003C94 are not equal within the bits specified by the contents of GPR4.

Memory Location:	13A74
Hexadecimal Instruction:	97063C00 (R=6, X=0, BR=6)
Assembly Language Coding:	CMMW 6,X'3C00'(6)

Before	PSD1	GPR4	GPR6	BR6	Memory Word 003C94
	0A013A74	00FFFF00	132A1C04	00000094	472A3D04
After	PSD1	GPR4	GPR6	BR6	Memory Word 003C94
	02013A78	00FFFF00	132A1C04	00000094	472A3D04

NONBASE REGISTER MODE EXAMPLE

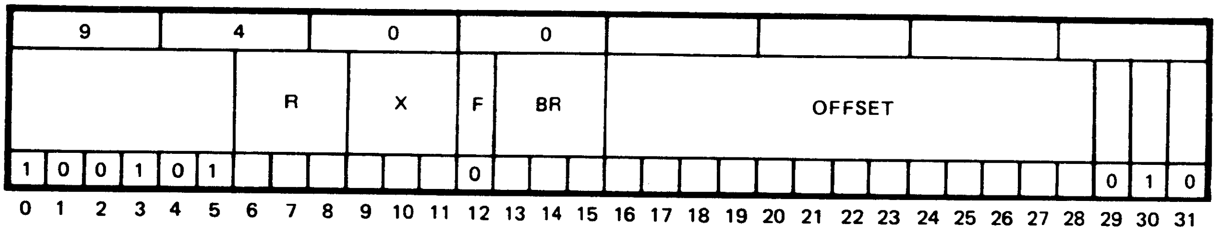
The contents of GPR6 and memory word 13C94 are not equal within the bits specified by the contents of GPR4.

Memory Location:	13A74
Hexadecimal Instruction:	97 01 3C 94 (R=6, X=0, I=0)
Assembly Language Coding:	CMMW 6,X'3C94'

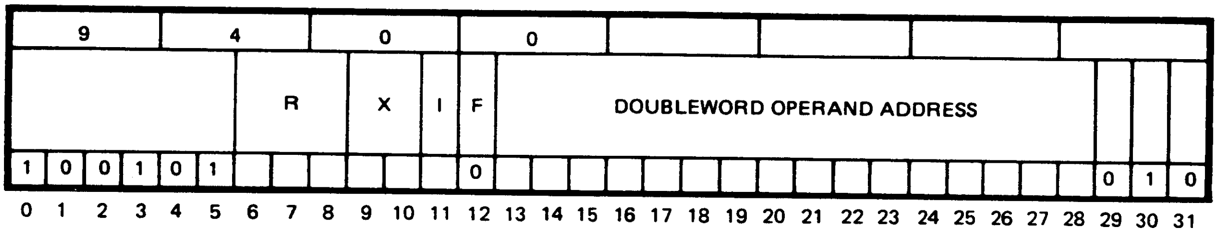
Before	PSD1	GPR4	GPR6	Memory Word 13C94
	08013A74	00FFFF00	132A1C04	472A3D04
After	PSD1	GPR4	GPR6	Memory Word 13C94
	00013A78	00FFFF00	132A1C04	472A3D04

COMPARE MASKED WITH MEMORY DOUBLEWORD 9400

CMMD
d,*m,x



BASE REGISTER FORMAT



830206

NONBASE REGISTER FORMAT

DEFINITION

The contents of the effective doubleword locations (EWL, EWL+1) specified by the effective doubleword address (EDA) is accessed and compared (exclusive OR function) with the doubleword in the general purpose register (GPR) specified by R and R+1. R+1 is the GPR one greater than specified by R. Each result from the comparison is then masked (logical AND function) with the contents of the mask register (R4). The doubleword masked result is tested and condition code bit 4 is set if all 64 bits equal zero. The doubleword in the GPR specified by R and R+1 and the doubleword specified by the EDA remain unchanged.

NOTE

The GPR specified by R must be an even-numbered register.

SUMMARY EXPRESSION

$(R) \oplus (EWL) \& (R4), (R+1) \oplus (EWL+1) \& (R4) \rightarrow SCC_4$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Always zero
CC3: Always zero
CC4: Is set if the result is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of GPR7 and memory word 0031BC differ within the bit positions specified by the contents of GPR4.

Memory Location:	03000
Hexadecimal Instruction:	9706300A (R=6, X=0, BR=6)
Assembly Language Coding:	CMMD 6,X'3008'(6)

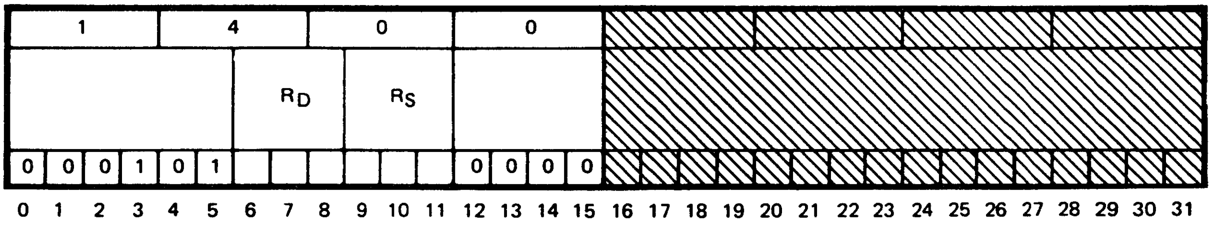
Before	PSD1	GPR4	GPR6	GPR7	BR6
	12003000	000FFFFF	FFF3791B	890A45D6	000001B0
	Memory Word 0031B8		Memory Word 0031BC		
	0003791B		890A45C2		
After	PSD1	GPR4	GPR6	GPR7	BR6
	02003004	000FFFFF	FFF3791B	890A45D6	000001B0
	Memory Word 0031B8		Memory Word 0031BC		
	0003791B		890A45C2		

NONBASE REGISTER MODE EXAMPLE

The contents of GPR7 and memory word 031BC differ within the bit positions specified by the contents of GPR4.

Memory Location:	03000
Hexadecimal Instruction:	97 00 31 BA (R=6, X=0, I=0)
Assembly Language Coding:	CMMD 6,X'31B8'

Before	PSD1	GPR4	GPR6	GPR7
	10003000	000FFFFF	FFF3791B	890A45D6
	Memory Word 031B8		Memory Word 031BC	
	0003791B		890A45C2	
After	PSD1	GPR4	GPR6	GPR7
	00003004	000FFFFF	FFF3791B	890A45D6
	Memory Word 031B8		Memory Word 031BC	
	0003791B		890A45C2	



DEFINITION

830207

The word in the general purpose register (GPR) specified by R_D is logically compared (exclusive OR function) with the word in the GPR specified by R_S. The result of the comparison is then masked (logical AND function) with the contents of the mask register (R4). The result is tested and condition code bit 4 is set if all 32 bits equal zero. The words specified by R_S and R_D remain unchanged.

SUMMARY EXPRESSION

$$(R_D) \oplus (R_S) \ \& \ (R4) \rightarrow SCC_4$$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Always zero
CC3: Always zero
CC4: Is set if the result is equal to zero

NONBASE AND BASE REGISTER MODE EXAMPLE

The contents of GPR1 and GPR2 are identical within the bit positions specified by the contents of GPR4. CC4 is set.

Memory Location:	050D2
Hexadecimal Instruction:	14 A0 (R_D=1, R_S=2)
Assembly Language Coding:	CMR 2,1

Before	PSD1		GPR1	GPR2	GPR4
	100050D2	(Nonbase)	583C94A2	0C68C5F6	AAAAAAAA
	120050D2	(Base)			
After	PSD1		GPR1	GPR2	GPR4
	080050D5	(Nonbase)	583C94A2	0C68C5F6	AAAAAAAA
	0A0050D5	(Base)			

This page intentionally left blank

6.2.6 Logical Instructions

The logical instruction group provides the capability of performing AND, OR, and exclusive OR operations on bytes, halfwords, words, and doublewords in memory and general purpose registers. Provisions have been made to allow the result of register-to-register OR and exclusive OR operations to be masked with the contents of the mask register (R4) before final storage.

6.2.6.1 Instruction Format

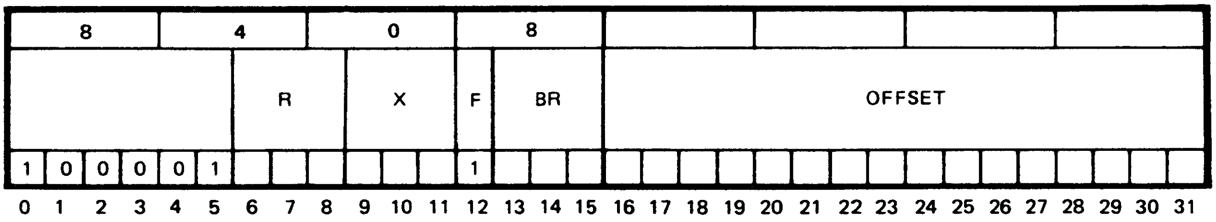
The logical instruction group uses the standard memory reference and interregister formats.

6.2.6.2 Condition Code

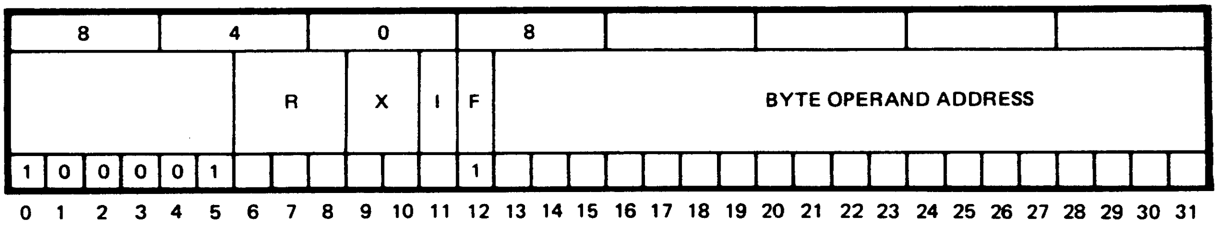
A condition code is set during execution of most logical instructions to indicate whether the result of that operation was greater than, less than, or equal to zero.

AND MEMORY BYTE 8408

ANMB d,*m,x



BASE REGISTER FORMAT



830208

NONBASE REGISTER FORMAT

DEFINITION

The contents of the effective byte location (EBL) specified by the effective byte address (EBA) is accessed and logically ANDed with the least-significant byte (bits 24-31) of the general purpose register (GPR) specified by R. The result is transferred to bit positions 24-31 of the GPR specified by R. Bit positions 0-23 of the GPR specified by R remain unchanged.

SUMMARY EXPRESSION

$(EBL)\&(R_{24-31}) \rightarrow R_{24-31}$

R_{0-23} unchanged

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R_{24-31}) is greater than zero
CC3: Always zero
CC4: Is set if (R_{24-31}) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of memory byte 000373 are ANDed with the least-significant byte of GPR1; the result replaces the byte in GPR1. CC2 is set.

Memory Location:	00200
Hexadecimal Instruction:	848E0300 (R=1, X=0, BR=6)
Assembly Language Coding:	ANMB 1,X'0300'(6)

Before:	PSD1	GPR1	BR6	Memory Byte 000373
	02000200	36AC718F	00000073	C7
After	PSD1	GPR1	BR6	Memory Byte 000373
	22000204	36AC7187	00000073	C7

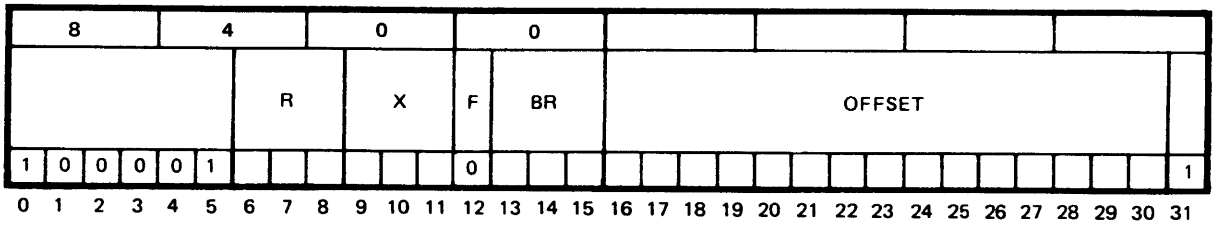
NONBASE REGISTER MODE EXAMPLE

The contents of memory byte 00373 are ANDed with the least-significant byte of GPR1; the result replaces the byte in GPR1. CC2 is set.

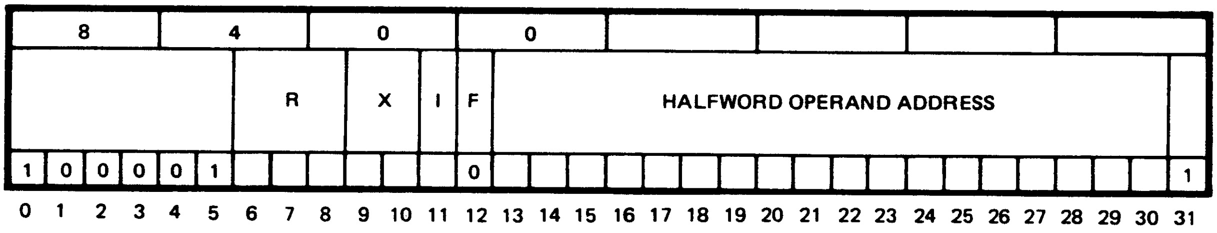
Memory Location:	00200
Hexadecimal Instruction:	84 88 03 73 (R=1, X=0, I=0)
Assembly Language Coding:	ANMB 1,X'373'

Before	PSD1	GPR1	Memory Byte 00373
	00000200	36AC718F	C7
After	PSD1	GPR1	Memory Byte 00373
	20000204	36AC7187	C7

AND MEMORY HALFWORD 8400 ANMH d,*m,x



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

DEFINITION

The contents of the effective halfword location (EHL) specified by the effective halfword address (EHA) is accessed and logically ANDed with the least-significant halfword (bits 16-31) of the general purpose register (GPR) specified by R. The result is transferred to bit positions 16-31 of the GPR specified by R. Bit positions 0-15 of the GPR specified by R remain unchanged.

SUMMARY EXPRESSION

$(\mathrm{EHL})\&(R_{16-31}) \rightarrow R_{16-31}$

R_{0-15} unchanged

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R_{16-31}) is greater than zero
CC3: Always zero
CC4: Is set if (R_{16-31}) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of memory halfword 0012A2 are ANDed with the right halfword of GPR6; the result replaces the halfword in GPR6. CC4 is set.

Memory Location:	01000
Hexadecimal Instruction:	87061203 (R=6, X=0, BR=6)
Assembly Language Coding:	ANMH 6,X'1202'(6)

Before	PSD1	GPR6	BR6	Memory Halfword 0012A2
	42001000	4F638301	000000A0	70F6
After	PSD1	GPR6	BR6	Memory Halfword 0012A2
	0A001004	4F630000	000000A0	70F6

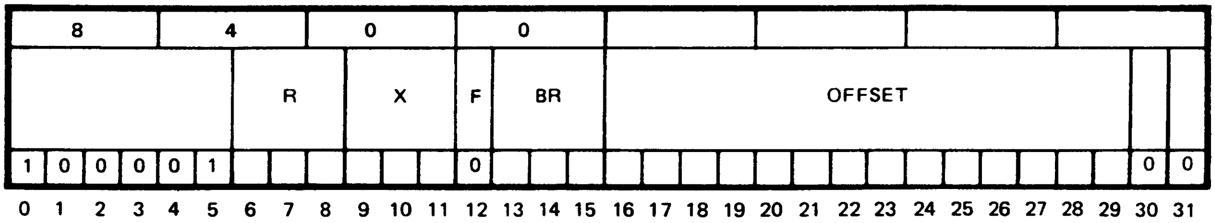
NONBASE REGISTER MODE EXAMPLE

The contents of memory halfword 012A2 are ANDed with the right halfword of GPR6; the result replaces the halfword in GPR6. CC4 is set.

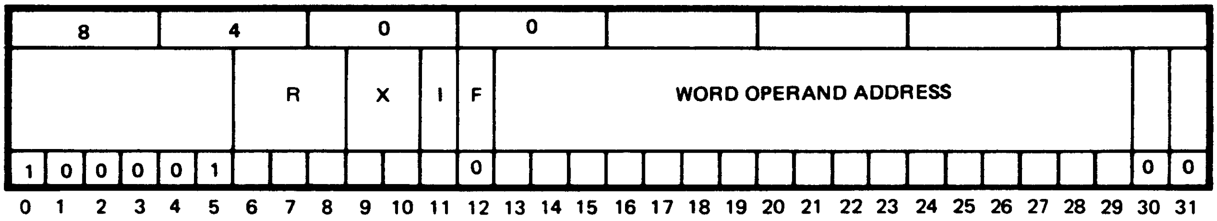
Memory Location:	01000
Hexadecimal Instruction:	87 00 12 A3 (R=6, X=0, I=0)
Assembly Language Coding:	ANMH 6,X'12A2'

Before	PSD1	GPR6	Memory Halfword 012A2
	40001000	4F638301	70F6
After	PSD1	GPR6	Memory Halfword 012A2
	08001004	4F630000	70F6

AND MEMORY WORD 8400 — ANMW d,*m,x



BASE REGISTER FORMAT



830210

NONBASE REGISTER FORMAT

DEFINITION

The contents of the effective word location (EWL) specified by the effective word address (EWA) is accessed and logically ANDed with the word located in the GPR specified by R.

SUMMARY EXPRESSION

(EWL)&(R) → R

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R_{0-31}) is greater than zero
CC3: Is set if (R_{0-31}) is less than zero
CC4: Is set if (R_{0-31}) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of GPR4, BR6 and the instruction offset are added to obtain the logical address. The contents of memory word 000FD0 are ANDed with the contents of GPR6, and the result replaces the contents of that register. CC3 is set.

Memory Location:	00 F1C
Hexadecimal Instruction:	87460E00 (R=6, BR=6, X=4)
Assembly Language Coding:	ANMW 6, X'0E00' (6), 4

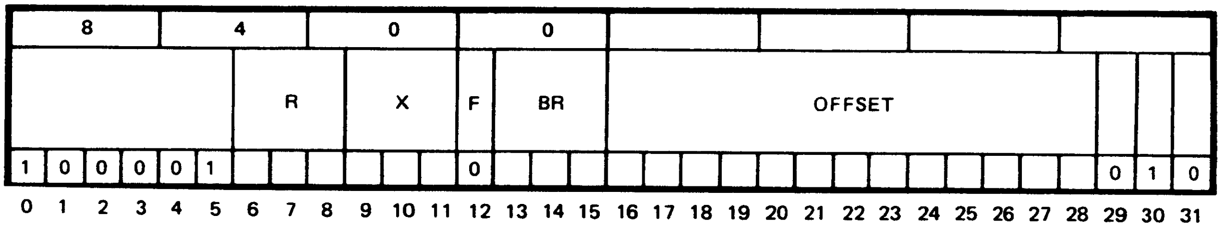
Before	PSD1	GPR6	BR6	GPR4	Memory Word 000FD0
	0A000F1C	F0F0F0F0	00000100	000000D0	9ED13854
After	PSD1	GPR6	BR6	GPR4	Memory Word 000FD0
	12000F20	90D03050	00000100	000000D0	9ED13854

NONBASE REGISTER MODE EXAMPLE

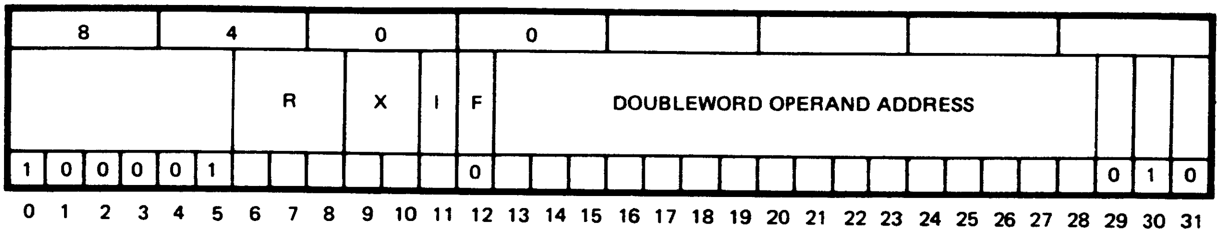
The contents of memory word 00FD0 are ANDed with the contents of GPR7, and the result replaces the contents of that register. CC3 is set.

Memory Location:	00F1C
Hexadecimal Instruction:	87 80 0F D0 (R=7, X=0, I=0)
Assembly Language Coding:	ANMW 7,X'FD0'

Before	PSD1	GPR7	Memory Word 00FD0
	08000F1C	F0F0F0F0	9ED13854
After	PSD1	GPR7	Memory Word 00FD0
	10000F20	90D03050	9ED13854



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

830211

DEFINITION

The contents of the effective word locations (EWL, EWL+1) specified by the effective doubleword address (EDA) is accessed and logically ANDed with the doubleword in the general purpose register (GPR) specified by R and R+1. R+1 is the GPR one greater than specified by R. The resulting doubleword is transferred to the GPR specified by R and R+1.

NOTE

The GPR specified by R must be an even-numbered register.

SUMMARY EXPRESSION

(EWL+1)&(R+1) → R+1

(EWL)&(R) → R

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R,R+1) is greater than zero
CC3: Is set if (R,R+1) is less than zero
CC4: Is set if (R,R+1) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of memory word 00081C are ANDed with the contents of GPR5; the result replaces the contents of GPR5. The contents of memory word 000818 are ANDed with the contents of GPR4; the result replaces the contents of GPR4. CC2 is set.

Memory Location:	00674
Hexadecimal Instruction:	8606080A (R=4, X=0, BR=6)
Assembly Language Coding:	ANMD 4,X'808'(6)

Before	PSD1	GPR4	GPR5	BR6
	02000674	9045C64A	32B08F00	00000010
	Memory Word 000818		Memory Word 00081C	
	684A711C		8104A2BC	
After	PSD1	GPR4	GPR5	BR6
	22000678	00404008	00008200	00000010
	Memory Word 000818		Memory Word 00081C	
	684A711C		8104A2BC	

NONBASE REGISTER MODE EXAMPLE

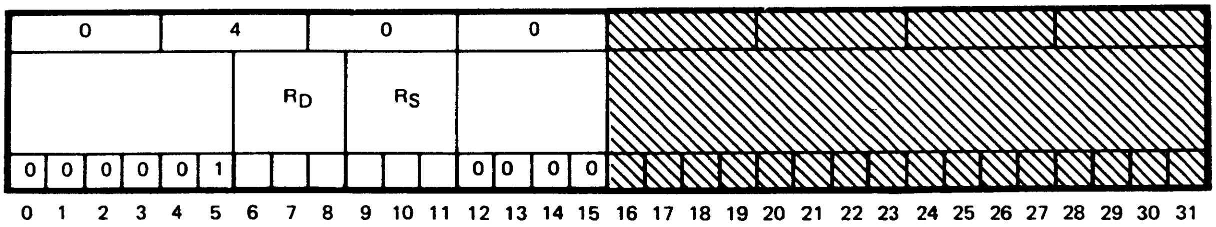
The contents of memory word 0081C are ANDed with the contents of GPR5; the result replaces the contents of GPR5. The contents of memory word 00818 are ANDed with the contents of GPR4; the result replaces the contents of GPR4. CC2 is set

Memory Location:	00674
Hexadecimal Instruction:	86 00 08 1A (R=4, X=0, I=0)
Assembly Language Coding:	ANMD 4,X'818'

Before	PSD1	GPR4	GPR5
	00000674	9045C64A	32B08F00
	Memory Word 00818	Memory Word 0081C	
	684A711C	8104A2BC	
After	PSD1	GPR4	GPR5
	20000678	00404008	00008200
	Memory Word 00818	Memory Word 0081C	
	684A711C	8104A2BC	

AND REGISTER AND REGISTER
0400

ANR
s,d



Bits 0–3	Bits 4–7		Bits 8–11	Bits 12–15	Bits 16–31
0	4		0	0	▨
		RD (6–8)	RS (9–11)		▨
0 0 0 0	0 1			0 0 0 0	▨

830212

DEFINITION

The word in the general purpose register (GPR) specified by R_D is logically ANDed with the word in the GPR specified by R_S. The resulting word is transferred to the GPR specified by R_D.

SUMMARY EXPRESSION

$(R_S)\&(R_D) \rightarrow R_D$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R_D) is greater than zero
CC3: Is set if (R_D) is less than zero
CC4: Is set if (R_D) is equal to zero

NONBASE AND BASE REGISTER MODE EXAMPLE

The contents of GPR1 and GPR7 are ANDed and the result is transferred to GPR1. CC2 is set.

Memory Location:	03812
Hexadecimal Instruction:	04 F0 (R_D=1, R_S=7)
Assembly Language Coding:	ANR 7,1

Before	PSD1		GPR1	GPR7
	40003812	(Nonbase)	AC881101	000FFFFF
	42003812	(Base)		
After	PSD1		GPR1	GPR7
	20003815	(Nonbase)	00081101	000FFFFF
	22003815	(Base)		

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
8				8				0				8																			
						R			X			F	BR			OFFSET															
1	0	0	0	1	0							1																			

BASE REGISTER FORMAT

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
8				8				0				8																			
						R			X		I	F	BYTE OPERAND ADDRESS																		
1	0	0	0	1	0							1																			

830213

NONBASE REGISTER FORMAT

DEFINITION

The contents of the effective byte location (EBL) specified by the effective byte address (EBA) is accessed and logically ORed with the least-significant byte (bits 24-31) of the general purpose register (GPR) specified by R. The resulting byte is transferred to bit positions 24-31 of the GPR specified by R. Bit positions 0-23 of the GPR specified by R remain unchanged.

SUMMARY EXPRESSION

$(EBL) \vee (R_{24-31}) \rightarrow R_{24-31}$

R_{0-23} unchanged

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R_{0-31}) is greater than zero
CC3: Is set if (R_{0-31}) is less than zero
CC4: Is set if (R_{0-31}) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of memory byte 0008A3 are logically ORed with the least-significant byte of GPR1; the result replaces that byte in GPR1. CC2 is set.

Memory Location:	00600
Hexadecimal Instruction:	888E0800 (R=1, X=0, BR=6)
Assembly Language Coding:	ORMB 1,X'800'(6)

Before	PSD1	GPR1	BR6	Memory Byte 0008A3
	02000600	40404040	000000A3	3C
After	PSD1	GPR1	BR6	Memory Byte 0008A2
	22000604	4040407C	000000A3	3C

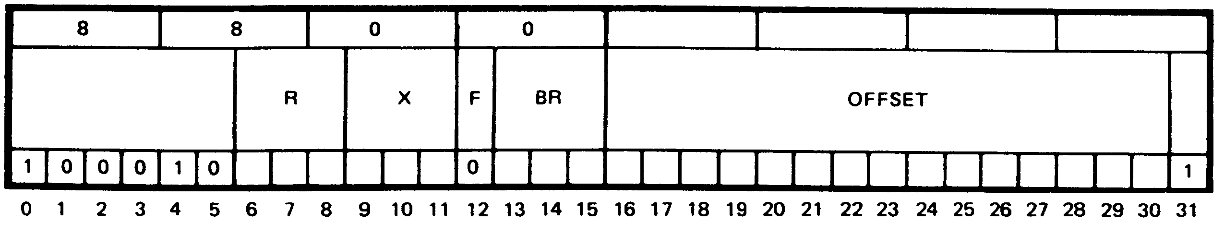
NONBASE REGISTER MODE EXAMPLE

The contents of memory byte 8A3 are logically ORed with the least-significant byte of GPR1; the result replaces that byte in GPR1. CC2 is set.

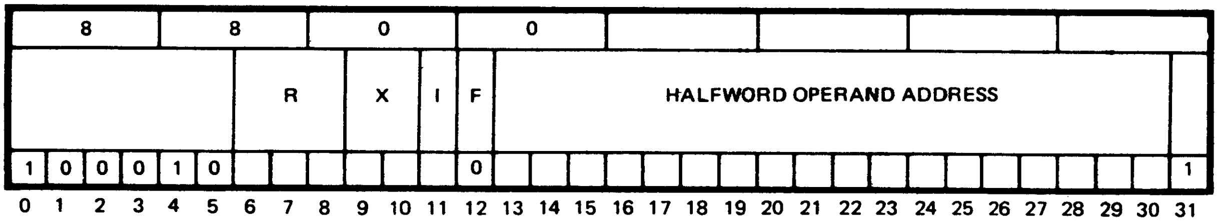
Memory Location:	00600
Hexadecimal Instruction:	88 88 08 A3 (R=1, X=0, I=0)
Assembly Language Coding:	ORMB 1,X'8A3'

Before	PSD1	GPR1	Memory Byte 8A3
	00000600	40404040	3C
After	PSD1	GPR1	Memory Byte 8A3
	20000604	4040407C	3C

OR MEMORY HALFWORD 8800 — ORMH d,*m,x



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

830214

DEFINITION

The contents of the effective halfword location (EHL) specified by the effective halfword address (EHA) is accessed and logically ORed with the least-significant halfword (bits 16-31) of the general purpose register (GPR) specified by R. The resulting halfword is transferred to bit positions 16-31 of the GPR specified by R. Bit positions 0-15 of the GPR specified by R remain unchanged.

SUMMARY EXPRESSION

$(EHL)v(R_{16-31}) \rightarrow R_{16-31}$

R_{0-15} unchanged

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R_{0-31}) is greater than zero
CC3: Is set if (R_{0-31}) is less than zero
CC4: Is set if (R_{0-31}) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of memory halfword 001942 are ORed with the right halfword of GPR6; the result replaces the halfword in GPR6. CC3 is set.

Memory Location:	018AC
Hexadecimal Instruction:	8B061903 (R=6, X=0, BR=6)
Assembly Language Coding:	ORMH 6,X'1902'(6)

Before	PSD1	GPR6	BR6	Memory Halfword 001942
	020018AC	BD71A4C6	00000040	45F3
After	PSD1	GPR6	BR6	Memory Halfword 001942
	120018B0	BD71E5F7	00000040	45F3

NONBASE REGISTER MODE EXAMPLE

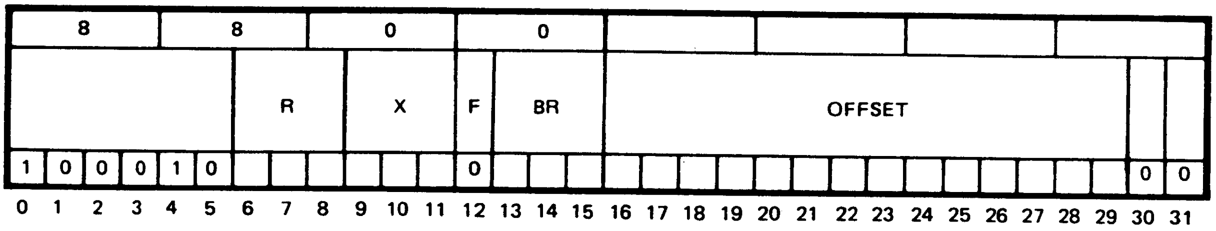
The contents of memory halfword 01946 are ORed with the right halfword of GPR6; the result replaces that halfword in GPR6. CC3 is set.

Memory Location:	018AC
Hexadecimal Instruction:	8B 00 19 47 (R=6, X=0, I=0)
Assembly Language Coding:	ORMH 6,X'1946'

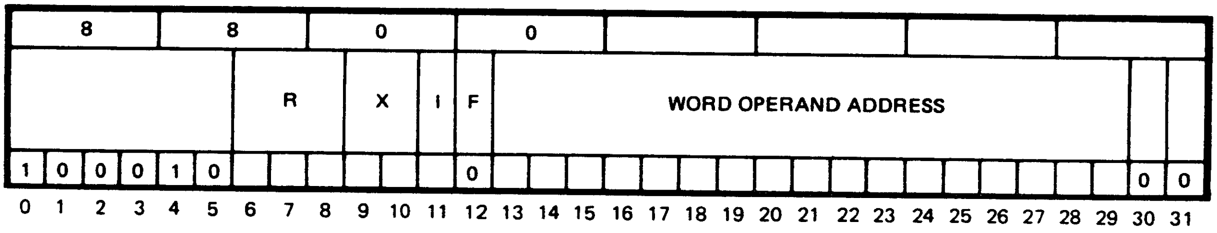
Before	PSD1	GPR6	Memory Halfword 01946
	000018AC	BD71A4C6	45F3
After	PSD1	GPR6	Memory Halfword 01946
	100018B0	BD71E5F7	45F3

OR MEMORY WORD
8800

ORMW
d,*m,x



BASE REGISTER FORMAT



830215

NONBASE REGISTER FORMAT

DEFINITION

The contents of the effective word location (EWL) specified by the effective word address (EWA) is accessed and logically ORed with the word in the general purpose register (GPR) specified by R. The result is transferred to the GPR specified by R.

SUMMARY EXPRESSION

(EWL)v(R) → R

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R_{0-31}) is greater than zero
CC3: Is set if (R_{0-31}) is less than zero
CC4: Is set of (R_{0-31}) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of GPR4, BR6 and the instruction offset are added to obtain the logical address. The contents of memory word 00520C are ORed with the contents of GPR2, and the result is transferred to GPR2. CC3 is set.

Memory Location:	05000
Hexadecimal Instruction:	89465000 (R=2, BR=6, X=4)
Assembly Language Coding:	ORMW 2, X '5000' (6), 4

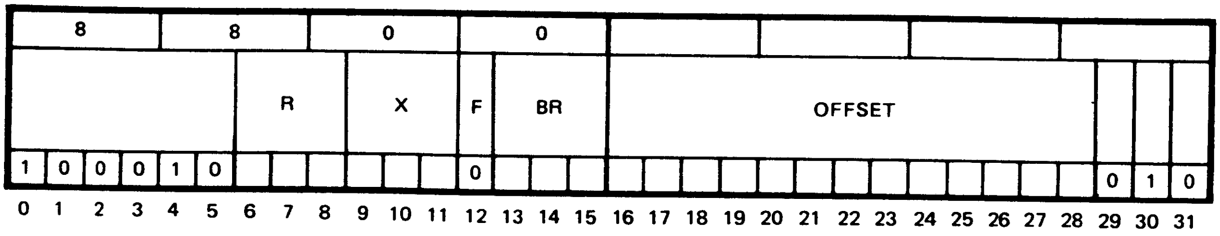
Before	PSD1	GPR2	BR6	GPR4	Memory Word 00520C
	42005000	88888888	0000000C	00000200	0EDC4657
After	PSD1	GPR2	BR6	GPR4	Memory Word 005200
	12005004	8EDCCEDF	0000000C	00000200	0EDC4657

NONBASE REGISTER MODE EXAMPLE

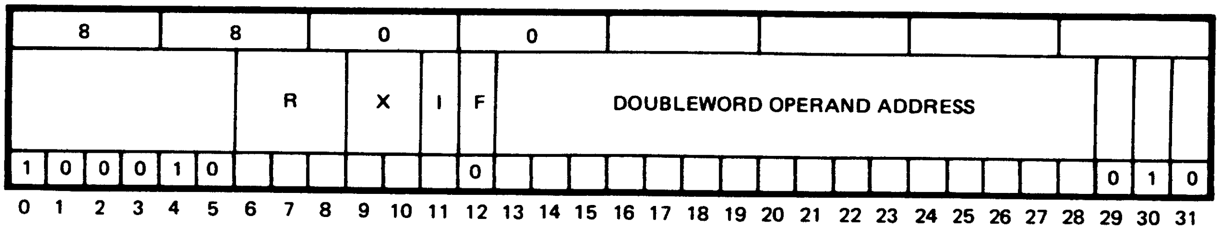
The contents of memory word 0520C are ORed with the contents of GPR3, and the result is transferred to GPR3. CC3 is set.

Memory Location:	05000
Hexadecimal Instruction:	89 80 52 0C (R=3, X=0, I=0)
Assembly Language Coding:	ORMW 3,X'520C'

Before	PSD1	GPR3	Memory Word 0520C
	40005000	88888888	0EDC4657
After	PSD1	GPR3	Memory Word 0520C
	40005000	8EDCCEDF	0EDC4657



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

DEFINITION

The contents of the effective doubleword locations (EWL, EWL+1) specified by the effective doubleword address (EDA) is accessed and logically ORed with the doubleword in the general purpose register (GPR) specified by R and R+1. R+1 is the GPR one greater than specified by R. The result is transferred to the GPR specified by R and R+1.

NOTE

The GPR specified by R must be an even-numbered register.

SUMMARY EXPRESSION

(EWL+1)v(R+1) → R+1

(EWL)v(R) → R

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R,R+1) is greater than zero
CC3: Is set if (R,R+1) is less than zero
CC4: Is set if (R,R+1) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of memory word 000C34 are ORed with the contents of GPR7; the result is transferred to GPR7. The contents of memory word 000C30 are ORed with the contents of GPR6; the result is transferred to GPR6. CC2 is set.

Memory Location:	00B68
Hexadecimal Instruction:	8B060C02 (R=6, X=0, BR=6)
Assembly Language Coding:	ORMD 6,X'C00'(6)

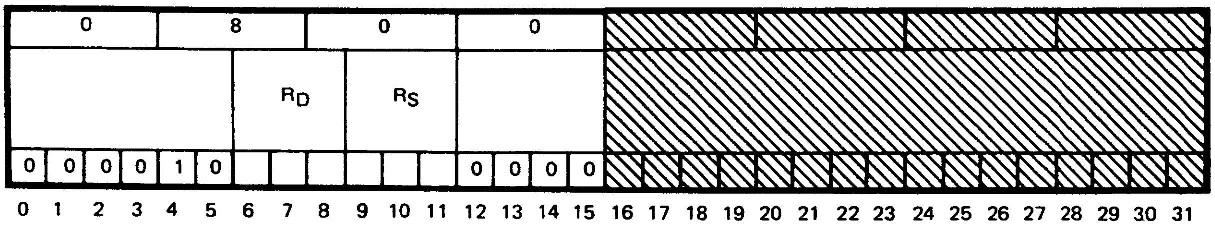
Before	PSD1	GPR6	GPR7	BR6
	12000B68	002A0031	001D0039	00000030
	Memory Word 000C30		Memory Word 000C34	
	18004C00		09002400	
After	PSD1	GPR6	GPR7	BR6
	22000B6C	182A4C31	091D2439	00000030
	Memory Word 000C30		Memory Word 000C34	
	18004C00		09002400	

NONBASE REGISTER MODE EXAMPLE

The contents of memory word 00C34 are ORed with the contents of GPR7, and the result is transferred to GPR7. The contents of memory word 00C30 are ORed with the contents of GPR6, and the result is transferred to GPR6. CC2 is set.

Memory Location:	00B68
Hexadecimal Instruction:	8B 00 0C 32 (R=6, X=0, I=0)
Assembly Language Coding:	ORMD 6,X'C30'

Before	PSD1	GPR6	GPR7
	10000B68	002A0031	001D0039
	Memory Word 00C30	Memory Word 00C34	
	18004C00	09002400	
After	PSD1	GPR6	GPR7
	20000B6C	182A4C31	091D2439
	Memory Word 00C30	Memory Word 00C34	
	18004C00	09002400	



830217

DEFINITION

The word in the general purpose register (GPR) specified by R_D is logically ORed with the word in the GPR specified by R_S. The result is transferred to the GPR specified by R_D.

SUMMARY EXPRESSION

$(R_S)v(R_D) \rightarrow R_D$

CONDITION CODE RESULTS

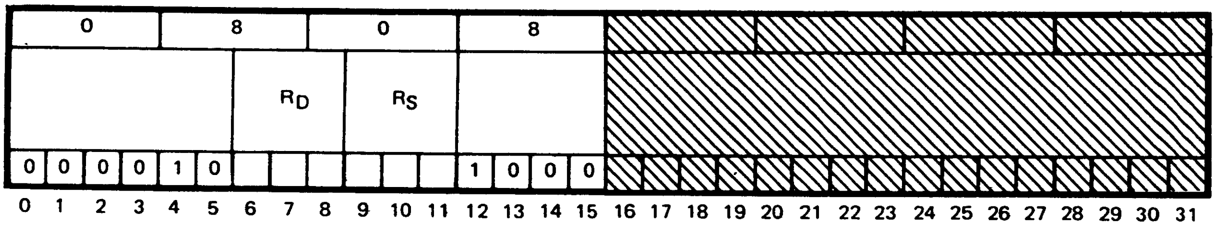
CC1: Always zero
CC2: Is set if (R_D) is greater than zero
CC3: Is set if (R_D) is less than zero
CC4: Is set if (R_D) is equal to zero

NONBASE AND BASE REGISTER MODE EXAMPLE

The contents of GPR1 and GPR2 are ORed, and the result is transferred to GPR1. CC3 is set.

Memory Location:	00F8A
Hexadecimal Instruction:	08 A0 (R_D=1, R_S=2)
Assembly Language Coding:	ORR 2,1

Before	PSD1	GPR1	GPR2
	40000F8A (Nonbase)	0001D63F	88880000
	42000F8A (Base)		
After	PSD1	GPR1	GPR2
	10000F8D (Nonbase)	8889D63F	88880000
	12000F8D (Base)		



830218

DEFINITION

The word in the general purpose register (GPR) specified by R_D is logically ORed with the word in the GPR specified by R_S. The resulting word is then masked (logical AND function) with the contents of the mask register (R4). The result is then transferred to the GPR specified by R_D.

SUMMARY EXPRESSION

$(R_S)v(R_D)\ \&(R4) \rightarrow R_D$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R_D) is greater than zero
CC3: Is set if (R_D) is less than zero
CC4: Is set if (R_D) is equal to zero

NONBASE AND BASE REGISTER MODE EXAMPLE

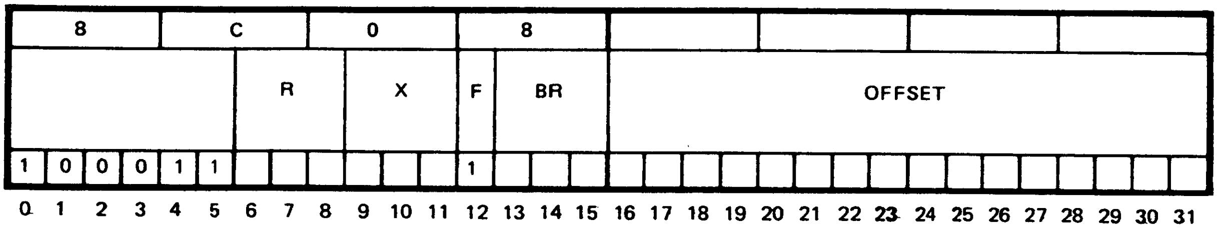
The contents of GPR5 and GPR6 are ORed; the result is ANDed with the contents of GPR4 and transferred to GPR6. CC2 is set.

Memory Location:	03956
Hexadecimal Instruction:	0B 58 (R_D=6, R_S=5)
Assembly Language Coding:	ORRM 5,6

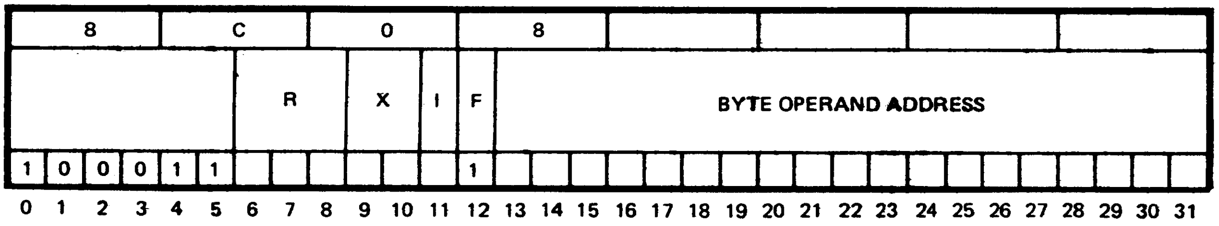
Before	PSD1	GPR4	GPR5	GPR6
	08003956 (Nonbase)	EEEEEEEE	37735814	2561CA95
	0A003956 (Base)			
After	PSD1	GPR4	GPR5	GPR6
	10003959 (Nonbase)	EEEEEEEE	37735814	2662CA84
	12003959 (Base)			

EXCLUSIVE OR MEMORY BYTE 8C08

EOMB d,*m,x



BASE REGISTER FORMAT



830219

NONBASE REGISTER FORMAT

DEFINITION

The contents of the effective byte location (EBL) specified by the effective byte address (EBA) is accessed and logically exclusive ORed with the least-significant byte (bits 24-31) of the GPR specified by R. The result is transferred to bit positions 24-31 of the GPR specified by R. Bits 0-23 of the GPR specified by R remain unchanged.

SUMMARY EXPRESSION

$$(\text{EBL}) \oplus (R_{24-31}) \rightarrow R_{24-31}$$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R_{0-31}) is greater than zero
CC3: Is set if (R_{0-31}) is less than zero
CC4: Is set if (R_{0-31}) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of memory byte 0013A3 are exclusive ORed with the least-significant byte of GPR0; the result replaces that byte in GPR0. CC3 is set.

Memory Location:	012F8
Hexadecimal Instruction:	8C0E1300 (R=0, X=0, BR=6)
Assembly Language Coding:	EOMB 0,X'1300'(6)

Before	PSD1	GPR0	BR6	Memory Byte 0013A3
	020012F8	D396F458	000000A3	A9
After	PSD1	GPR0	BR6	Memory Byte 0013A3
	120012FC	D396F4F1	000000A3	A9

NONBASE REGISTER MODE EXAMPLE

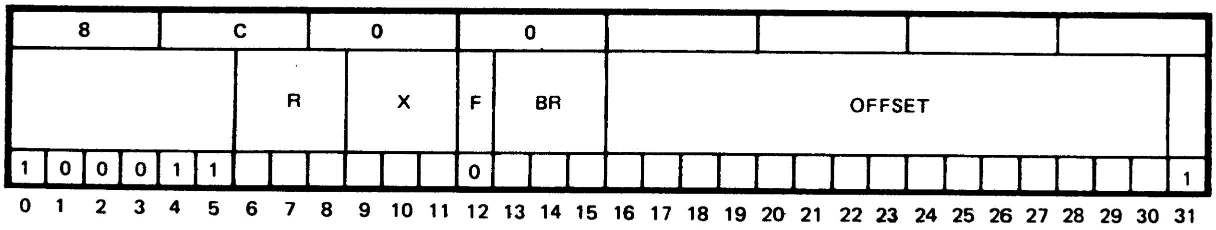
The contents of memory byte 013A3 are exlcusive ORed with the least-significant byte of GPR0; the result replaces that byte in GPR0. CC3 is set.

Memory Location:	012F8
Hexadecimal Instruction:	8C 08 13 A3 (R=0, X=0, I=0)
Assembly Language Coding:	EOMB 0,X'13A3'

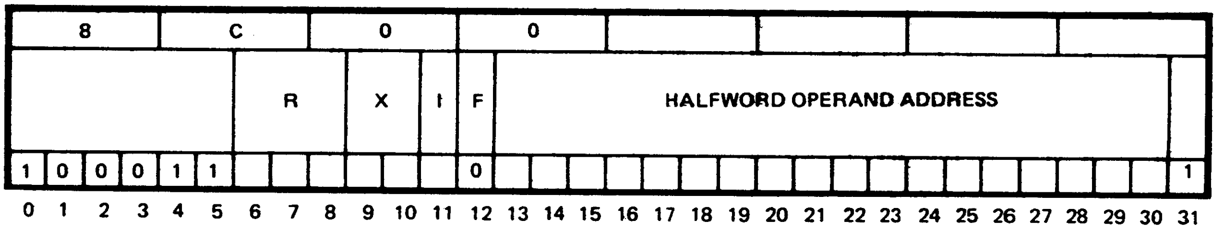
Before	PSD1	GPR0	Memory Byte 013A3
	000012F8	D396F458	A9
After	PSD1	GPR0	Memory Byte 013A3
	100012FC	D396F4F1	A9

EXCLUSIVE OR MEMORY HALFWORD 8C00

EOMH
d,*m,x



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

830220

DEFINITION

The contents of the effective halfword location (EHL) specified by the effective halfword address (EHA) is accessed and logically exclusive ORed with the least-significant halfword (bits 16-31) of the general purpose register (GPR) specified by R. The result is transferred to bit positions 16-31 of the GPR specified by R. Bit positions 0-15 of the GPR specified by R remain unchanged.

SUMMARY EXPRESSION

$$(EHL) \oplus (R_{16-31}) \rightarrow R_{16-31}$$

R_{0-15} unchanged

CONDITION CODE RESULTS

CC1:Always zero
CC2:Is set if (R_{0-31}) is greater than zero
CC3:Is set if (R_{0-31}) is less than zero
CC4:Is set if (R_{0-31}) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of memory halfword 000A42 are exclusive ORed with the right halfword of GPR5. The result replaces the halfword in GPR5. CC3 is set.

Memory Location:	00958
Hexadecimal Instruction:	8E860A03 (R=5, X=0, BR=6)
Assembly Language Coding:	EOMH 5,X'A02'(6)

Before	PSD1	GPR5	BR6	Memory Halfword 000A42
	42000958	96969696	00000040	5CAB
After	PSD1	GPR5	BR6	Memory Halfword 000A42
	1200095C	9696CA3D	00000040	5CAB

NONBASE REGISTER MODE EXAMPLE

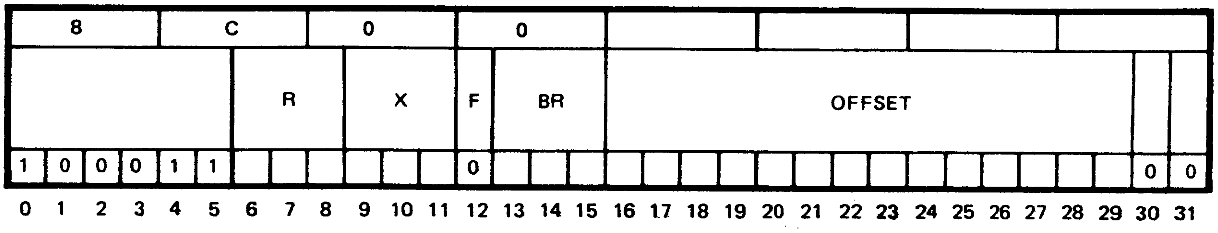
The contents of memory halfword 00A40 are exclusive ORed with the right halfword of GPR5; the result replaces that halfword in GPR5. CC3 is set.

Memory Location:	00958
Hexadecimal Instruction:	8E 80 0A 41 (R=5, X=0, I=0)
Assembly Language Coding:	EOMH 5,X'A40'

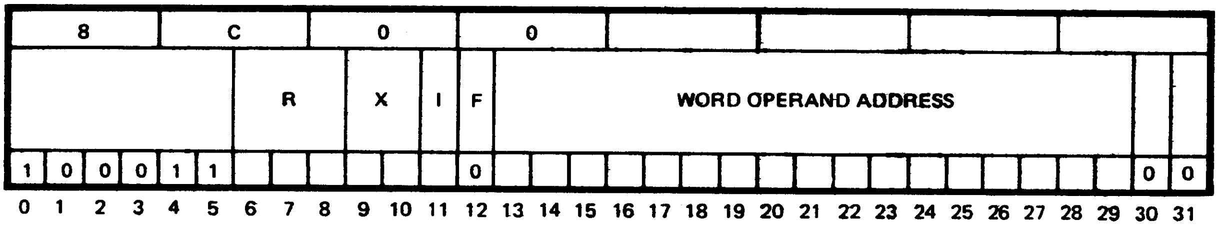
Before	PSD1	GPR5	Memory Halfword 00A40
	40000958	96969696	5CAB
After	PSD1	GPR5	Memory Halfword 00A40
	1000095C	9696CA3D	5CAB

EXCLUSIVE OR MEMORY WORD 8C00

EOMW d,*m,x



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

DEFINITION

The contents of the effective word location (EWL) specified by the effective word address (EWA) is accessed and logically exclusive ORed with the word in the general purpose register (GPR) specified by R. The result is transferred to the GPR specified by R.

SUMMARY EXPRESSION

$(EWL) \oplus (R) \rightarrow R$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R_{0-31}) is greater than zero
CC3: Is set if (R_{0-31}) is less than zero
CC4: Is set if (R_{0-31}) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of GPR4, BR6 and the instruction offset are added to obtain the logical address. The contents of memory word 008694 are exclusive ORed with the contents of GPR6; the result replaces the contents of GPR6. CC2 is set.

Memory Location:	185BC
Hexadecimal Instruction:	8F468600 (R=6, BR=6, X=4)
Assembly Language Coding:	EOMW 6, X '8600' (6), 4

Before	PSD1	GPR6	BR6	GPR4	Memory Word 008694
	020185BC	13579BDF	00000090	00000004	22222222
After	PSD1	GPR6	BR6	GPR4	Memory Word 008694
	220185C0	3175B9FD	00000090	00000004	22222222

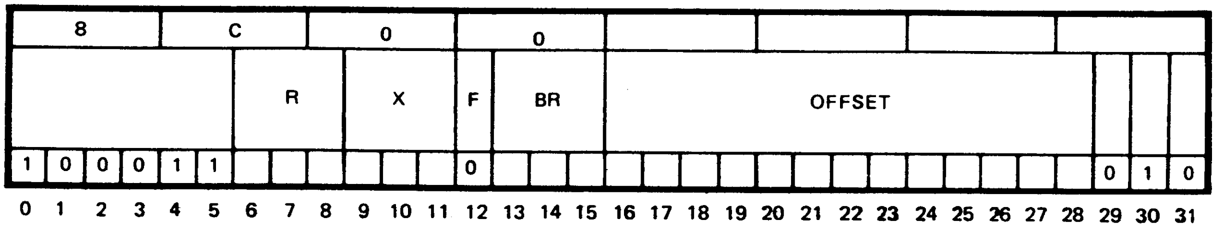
NONBASE REGISTER MODE EXAMPLE

The contents of memory word 18694 are exclusive ORed with the contents of GPR7; the result replaces the contents of GPR7. CC2 is set.

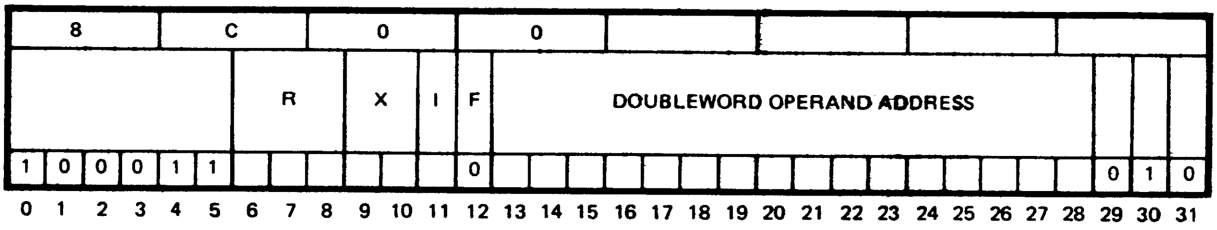
Memory Location:	185BC
Hexadecimal Instruction:	8F 81 86 94 (R=7, X=0, I=0)
Assembly Language Coding:	EOMW 7,X'18694'

Before	PSD1	GPR7	Memory Word 18694
	010185BC	13579BDF	22222222
After	PSD1	GPR7	Memory Word 18694
	200185C0	3175B9FD	22222222

EXCLUSIVE OR MEMORY DOUBLEWORD 8C00 — EOMD d,*m,x



BASE REGISTER FORMAT



830222

NONBASE REGISTER FORMAT

DEFINITION

The contents of the effective doubleword locations (EWL, EWL+1) specified by the effective doubleword address (EDA) is accessed and logically exclusive ORed with the doubleword in the general purpose register (GPR) specified by R and R+1. R+1 is the GPR one greater than specified by R. The result is transferred to the GPR specified by R and R+1.

NOTE

The GPR specified by R must be an even-numbered register.

SUMMARY EXPRESSION

$(EWL+1) \oplus (R+1) \rightarrow R+1$

$(EWL) \oplus (R) \rightarrow R$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R,R+1) is greater than zero
CC3: Is set if (R,R+1) is less than zero
CC4: Is set if (R,R+1) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of memory word 00053C and GPR7 are exclusive ORed and the result is transferred to GPR7. The contents of memory word 000538 are GPR6 are exclusive ORed and the result is transferred to GPR6. CC2 is set.

Memory Location:	00448
Hexadecimal Instruction:	8F06050A (R=6, X=0, BR=6)
Assembly Language Coding:	EOMD 6,X'508'(6)

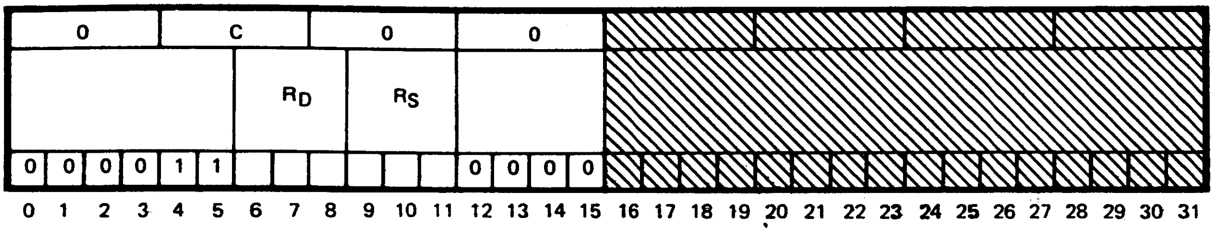
Before	PSD1	GPR6	GPR7	BR6
	02000448	00FFFF00	00FFF000	00000030
	Memory Word 000538		Memory Word 00053C	
	482144C0		2881433A	
After	PSD1	GPR6	GPR7	BR6
	2200044C	48DEBBC0	287EB33A	00000030
	Memory Word 000538		Memory Word 00053C	
	482144C0		2881433A	

NONBASE REGISTER MODE EXAMPLE

The contents of memory word 0053C and GPR7 are exclusive ORed and the result is transferred to GPR7. The contents of memory word 00538 and GPR6 are exclusive ORed and the result is transferred to GPR6. CC2 is set.

Memory Location:	00448
Hexadecimal Instruction:	8F 00 05 3A (R=6, X=0, I=0)
Assembly Language Coding:	EOMD 6,X'538'

Before	PSD1	GPR6	GPR7
	00000448	00FFFF00	00FFF000
	Memory Word 00538		Memory Word 0053C
	482144C0		2881433A
After	PSD1	GPR6	GPR7
	2000044C	48DEBBC0	287EB33A
	Memory Word 00538		Memory Word 0053C
	482144C0		2881433A



DEFINITION

The word in the general purpose register (GPR) specified by R_D is logically exclusive ORed with the word in the GPR specified by R_S. The result is transferred to the GPR specified by R_D.

SUMMARY EXPRESSION

$(R_S) \oplus (R_D) \rightarrow R_D$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R_D) is greater than zero
CC3: Is set if (R_D) is less than zero
CC4: Is set if (R_D) is equal to zero

NONBASE AND BASE REGISTER MODE EXAMPLE

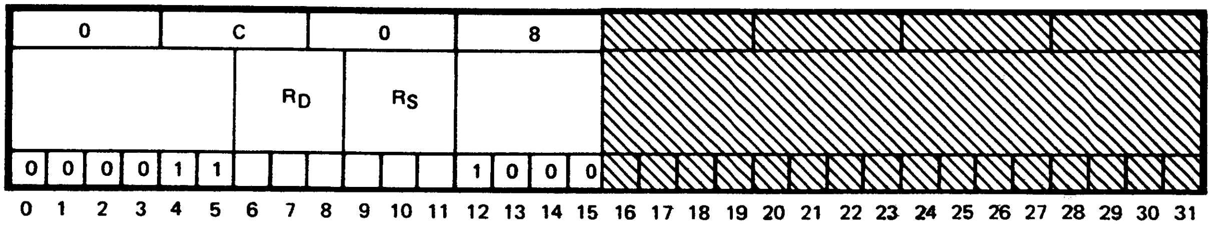
The contents of GPR6 and GPR7 are exclusive ORed, and the result is transferred to GPR7. CC2 is set.

Memory Location:	0139E
Hexadecimal Instruction:	0F E0 (R_D=7, R_S=6)
Assembly Language Coding:	EOR 6,7

Before	PSD1	GPR6	GPR7
	0100139E (Nonbase)	33333333	55555555
	0300139E (Base)		
After	PSD1	GPR6	GPR7
	200013A1 (Nonbase)	33333333	66666666
	220013A1 (Base)		

EXCLUSIVE OR REGISTER AND REGISTER MASKED 0C08

EORM s,d



830224

DEFINITION

The word in the general purpose register (GPR) specified by R_D is exclusive ORed with the word in the GPR specified by R_S. The resulting word is then masked (logical AND function) with the contents of the mask register (R4). The result is transferred to the GPR specified by R_D.

SUMMARY EXPRESSION

$(R_S) \oplus (R_D)\ \&\ (R4) \rightarrow R_D$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R_D) is greater than zero
CC3: Is set if (R_D) is less than zero
CC4: Is set if (R_D) is equal to zero

NONBASE AND BASE REGISTER MODE EXAMPLE

The contents of GPR6 and GPR7 are exclusive ORed. The result is ANDed with the contents of GPR4 and transferred to GPR7. CC4 is set.

Memory Location:	25A32
Hexadecimal Instruction:	0F E8 (R_D=7, R_S=6)
Assembly Language Coding:	EORM 6,7

Before	PSD1	GPR4	GPR6	GPR7
	00025A32 (Nonbase)	00FEDF00	9725A2C8	6C248237
	02025A32 (Base)			
After	PSD1	GPR4	GPR6	GPR7
	08025A34 (Nonbase)	00FEDF00	9725A2C8	00000000
	0A025A34 (Base)			

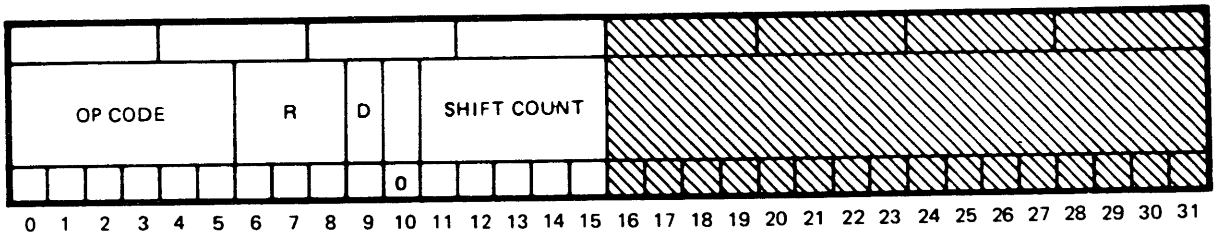
This page intentionally left blank

6.2.7 Shift Operation Instructions

This group of instructions provide the capability to perform arithmetic, logical, and circular, left or right, shift operations on the contents of words or doublewords in general purpose registers. Provisions have been made to allow normalize operations to be performed on the contents of words or doublewords in general purpose registers.

6.2.7.1 Instruction Format

Most of the shift operation instructions use the halfword format described below. The normalize, normalize double, and shift and count zeros instructions, which involve two registers specified by R_S and R_D, adapt to the standard interregister format but with the roles of source and destination interchanged.



830350

Bits 0-5 Define the operation code.

Bits 6-8 Designate a general purpose register address (0-7)

Bit 9 Designates direction.

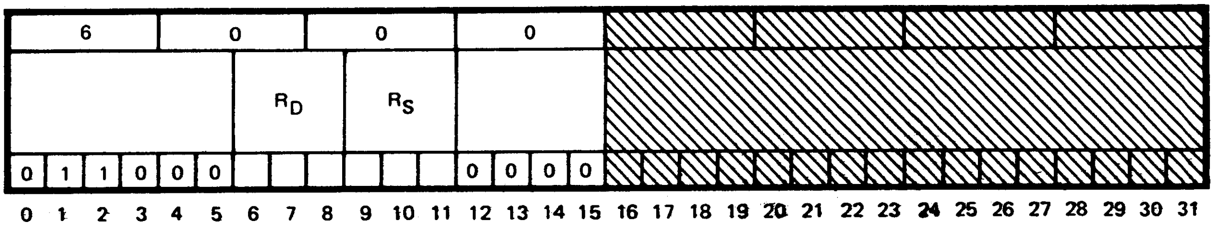
D=1 Designates shift left
D=0 Designates shift right

Bit 10 Unassigned.

Bits 11-15 Define the number of shifts to be made.

6.2.7.2 Condition Code

Most shift instructions leave the condition code unchanged. Those exceptions which alter the condition code contain comments to explain the changes incurred.



830225

DEFINITION

The word in the general purpose register (GPR) specified by R_D is shifted left, four bits at a time, until the five leftmost bits (bits 0-4) in R_D are neither all zeros nor all ones. The number of four-bit shifts required to do this is subtracted from 40_{16}, and stored in R_S. If R_D is initially zero, then no shifts are performed but R_S is set to zero.

NOTE

1. The normalized result must be further converted to the floating-point operand format prior to use by the floating-point arithmetic unit or standard FORTRAN floating-point subroutines. In addition, a test must be made for minus full scale (1XXX XXXX 0000 0000 --- 0000) and a conversion made to (1YYY YYYY 1111 0000 ---- 0000), where YYY YYYY is one less than XXX XXXX.

2. This instruction is used for the nonbase register mode only.

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

NONBASE REGISTER MODE EXAMPLE

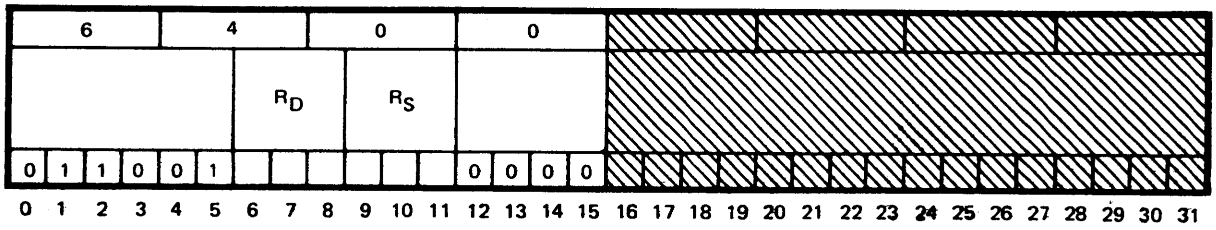
The content of GPR6 (R_D) is normalized by shifting three hexadecimal digits to the left. The exponent is determined by subtracting 40_{16} minus 3. The result is transferred to GPR1 (R_S).

Memory Location:	00D32
Hexadecimal Instruction:	63 10 (R_D = 6, R_S = 1)
Assembly Language Coding:	NOR 6,1

Before	PSD1	GPR1	GPR6
	20000D32	12345678	0002E915
After	PSD1	GPR1	GPR6
	20000D35	0000003D	2E915000

NORMALIZE DOUBLE 6400

NORD d,s



830226

DEFINITION

The doubleword in the general purpose registers (GPRs) specified by R_D and R_D+1 is shifted left, four bits at a time, until the five leftmost bits (bits 0-4) in R_D are neither all zeros nor all ones. The number of four bits shifts required to do this is subtracted from 40_{16}, and stored in R_S. If R_D and R_D+1 are initially zero, then no shifts are performed, but R_S is set to zero.

NOTE

1. The normalized result must be further converted to the floating-point operand format prior to use by the floating-point arithmetic unit or standard FORTRAN floating-point subroutines. In addition, a test must be made for minus full scale (1XXX XXXX 0000 0000 --- 0000) and a conversion made to (1YYY YYYY 1111 0000 --- 0000), where YYY YYYY is one less than XXX XXXX.

2. The instruction is used for the nonbase register mode only

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

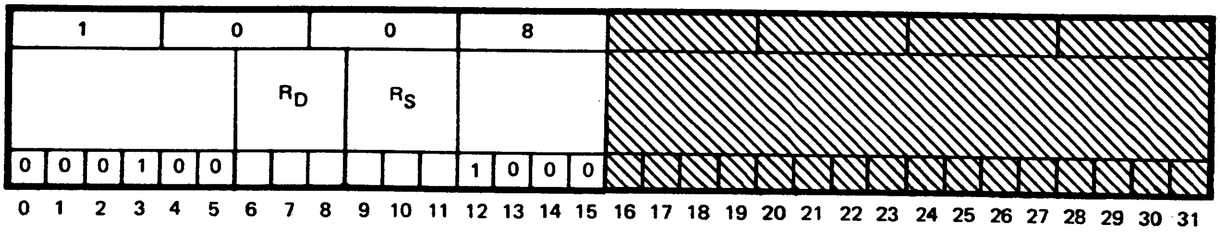
NONBASE REGISTER MODE EXAMPLE

The doubleword obtained from the contents of GPR6 and GPR7 is normalized by shifting nine hexadecimal digits to the left. The result is returned to GPR6 and GPR7, and the exponent (40_{16} minus 9) is transferred to GPR1.

Memory Location:	0046E
Hexadecimal Instruction:	67 10 (R_S=1, R_D=6)
Assembly Language Coding:	NORD 6,1

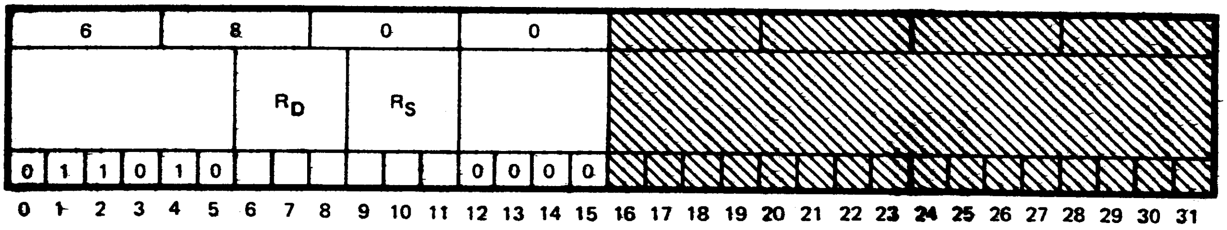
Before	PSD1	GPR1	GPR6	GPR7
	1000046E	9ABCDEF0	FFFFFFF	FF3AD915
After	PSD1	GPR1	GPR6	GPR7
	10000471	00000037	F3AD9150	00000000

SHIFT AND COUNT ZEROS



BASE REGISTER FORMAT (1008)

SACZ
s,d

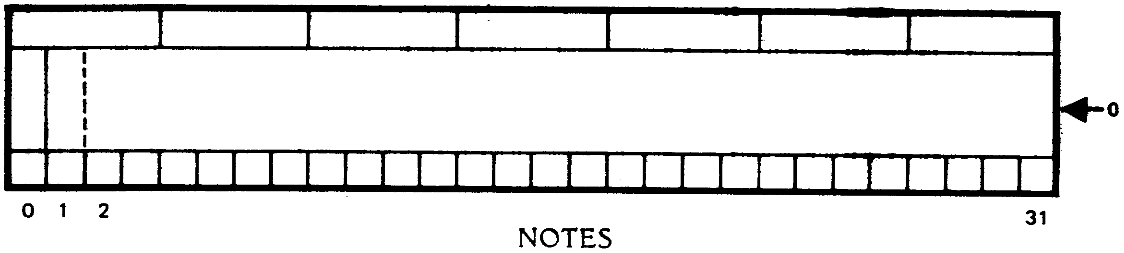


NONBASE REGISTER FORMAT (6800)

SCZ
d,s

DEFINITION

The word in the general purpose register (GPR) specified by R_D is shifted left, one bit position at a time, until the sign (bit 0) changes from zero to one. The contents are then shifted left one more bit position, and the total number of shifts minus one is placed in bit positions 27-31 of the GPR specified by R_S. Bit positions 0-26 of the GPR specified by R_S are set to zeros. The shift count specifies the most-significant bit position (0-31) of R_D that was equal to one.



NOTES

830227

1. If the contents of the GPR specified by R_D are equal to zero, the shift count placed in bit positions 27-31 of the GPR specified by R_S is zero, and condition code bit 4 is set to one.

2. If the sign (bit 0) of the GPR specified by R_D is equal to one, the shift count placed in bit positions 27-31 of the GPR specified by R_S is zero, and condition code bit 4 is cleared to zero.

CONDITION CODE RESULTS

CC1: Always zero
CC2: Always zero
CC3: Always zero
CC4: Is set if original R_D 0-31 is equal to zero (also refer to note 2)

BASE REGISTER MODE EXAMPLE

SACZ
s,d

The contents of GPR4 are left shifted 10 bits at which point bit 0 becomes equal to one. The contents are then shifted one more bit position, and the zero count of 10_{10} (A_{16}) is transferred to GPR2 bits 27-31, bits 0-26 reset to zero.

Memory Location:	0399E
Hexadecimal Instruction:	12 28 (R_S = 2, R_D = 4)
Assembly Language Coding:	SACZ 2,4

Before	PSD1	GPR2	GPR4
	2200399E	12345678	00300611
After	PSD1	GPR2	GPR4
	020039A0	0000000A	80308800

NONBASE REGISTER MODE EXAMPLE

SCZ
d,s

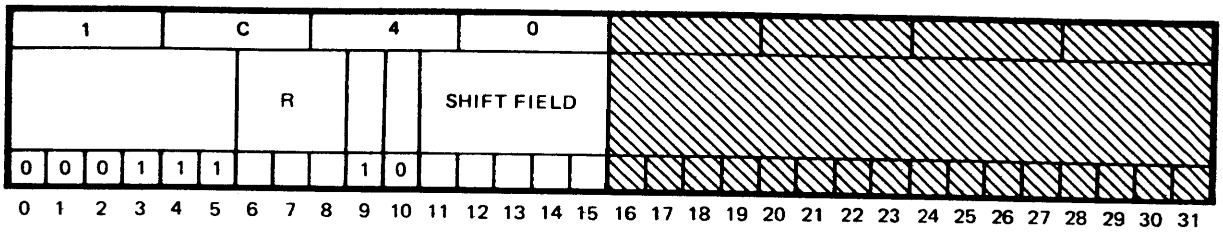
The contents of GPR4 are left shifted 10 bits at which point bit 0 becomes equal to one. The contents are then shifted one more bit position, and the zero count of 10_{10} (A_{16}) is transferred to GPR2 bits 27-31, bits 0-26 reset to zero.

Memory Location:	0399E
Hexadecimal Instruction:	6A 20 (R_D=4, R_S=2)
Assembly Language Coding:	SCZ 4, 2

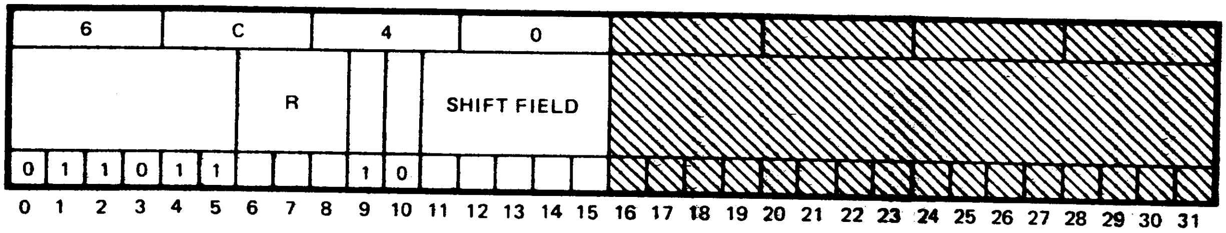
Before	PSD1	GPR2	GPR4
	2000399E	12345678	00300611
After	PSD1	GPR2	GPR4
	000039A0	0000000A	80308800

SHIFT LEFT ARITHMETIC

SLA
d,v



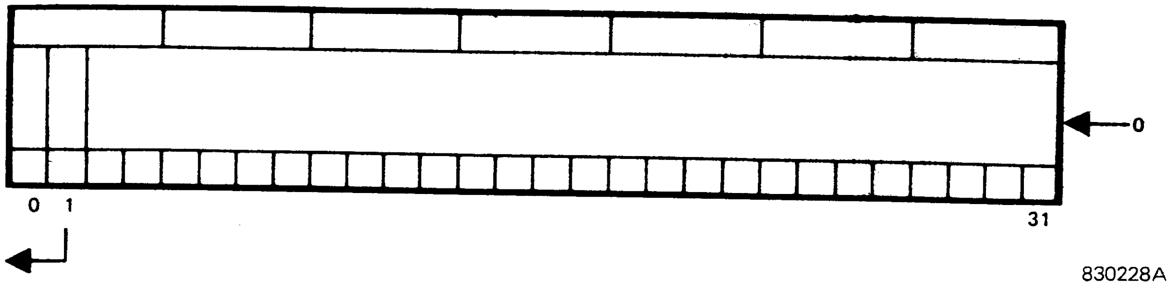
BASE REGISTER FORMAT (1C40)



NONBASE REGISTER FORMAT (6C40)

DEFINITION

Bit positions 1-31 of the general purpose register (GPR) specified by R are shifted left the number of bit positions specified by the shift field (bits 11-15) in the instruction word. As bits are shifted to the left, the word is zero-filled from the right. Bit position 0 (sign bit) of the GPR specified by R remains unchanged. Condition code bit 1 is set to one if any bit shifted out of position 1 differs from the sign bit.



CONDITION CODE RESULTS

CC1: Is set if arithmetic exception
CC2: Always zero
CC3: Always zero
CC4: Always zero

BASE REGISTER MODE EXAMPLE

The contents of GPR6 are left shifted 12_{10} bit positions and zero filled from the right.

Memory Location:	00106
Hexadecimal Instruction:	1F4C (R=6, shift field = 12_{10})
Assembly Language Coding:	SLA 6,12

Before	PSD1	GPR6
	12000106	000013AD
After	PSD1	GPR6
	02000109	013AD000

NONBASE REGISTER MODE EXAMPLE

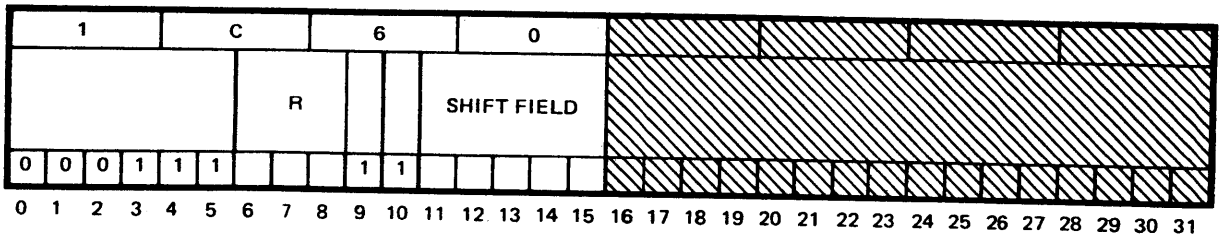
The contents of GPR6 are left shifted 12_{10} bit positions and zero filled from the right.

Memory Location:	00106
Hexadecimal Instruction:	6F 4C (R=6, shift field = 12_{10})
Assembly Language Coding:	SLA 6,12

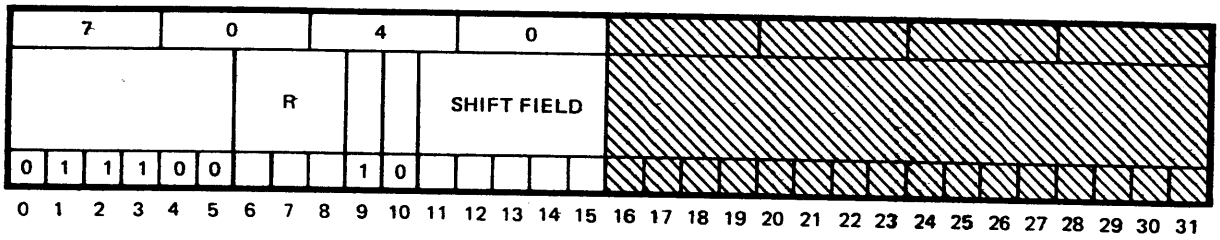
Before	PSD1	GPR6
	10000106	000013AD
After	PSD1	GPR6
	00000109	013AD000

SHIFT LEFT LOGICAL

SLL
d,v



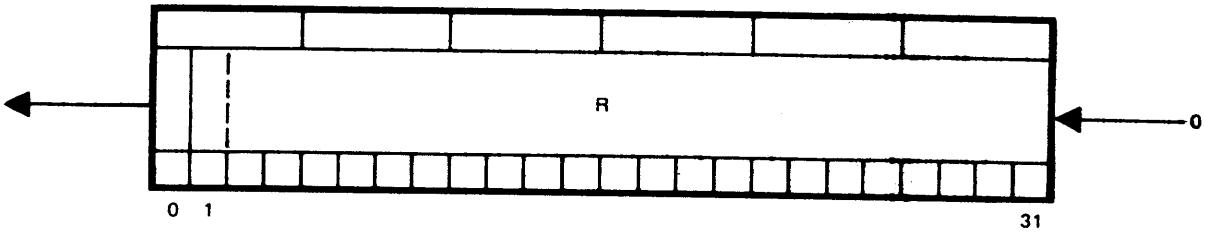
BASE REGISTER FORMAT (1C60)



NONBASE REGISTER FORMAT (7040)

DEFINITION

The word in the general purpose register (GPR) specified by R is left shifted the number of bit positions specified by the shift field (bits 11-15) in the instruction word. As bits are shifted to the left, the word is zero filled from the right.



830229

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

BASE REGISTER MODE EXAMPLE

The contents of GPR7 are shifted to the left 20_{10} bit positions, and zero filled from the right.

Memory Location:		00812
Hexadecimal Instruction:		1FF4 (R=7, shift field = 20_{10})
Assembly Language Coding:		SLL 7,20
Before	PSD1 A2000812	GPR7 12345678
After	PSD1 A2000815	GPR7 67800000

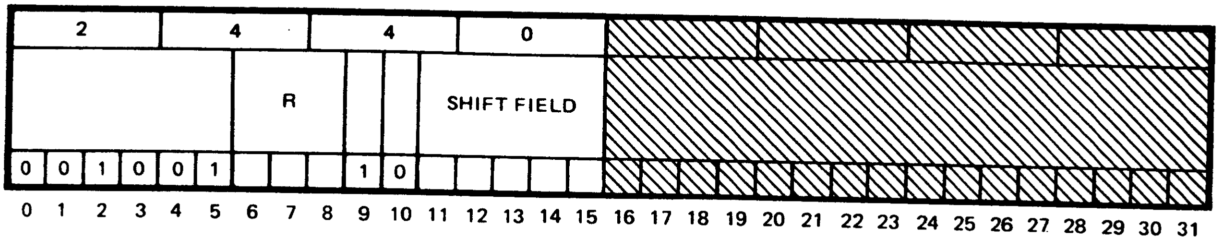
NONBASE REGISTER MODE EXAMPLE

The contents of GPR7 are shifted to the left 20_{10} bit positions, and zero filled from the right.

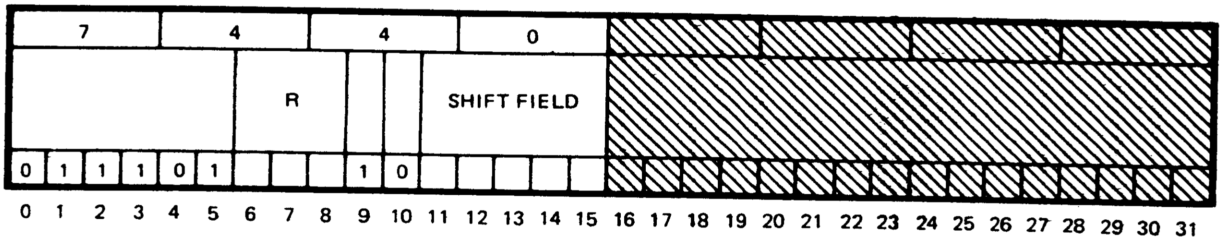
Memory Location:		00812
Hexadecimal Instruction:		73 D4 (R=7, shift field = 20_{10})
Assembly Language Coding:		SLL 7,20
Before	PSD1 A0000812	GPR7 12345678
After	PSD1 A0000815	GPR7 67800000

SHIFT LEFT CIRCULAR

SLC
d,v



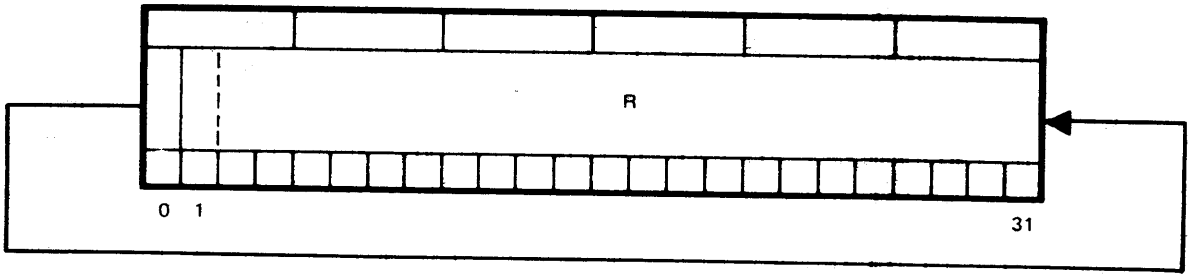
BASE REGISTER FORMAT (2440)



NONBASE REGISTER FORMAT (7440)

DEFINITION

The word in the general purpose register (GPR) specified by R is shifted left the number of bit positions specified by the shift field (bits 11-15) in the instruction word. Bits shifted out of bit position 0 are shifted into bit position 31.



CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

BASE REGISTER MODE EXAMPLE

The contents of GPR7 are shifted left circular for 16_{10} bit positions.

Memory Location:	001FA
Hexadecimal Instruction:	27D0 (R=7, shift field = 16_{10})
Assembly Language Coding:	SLC 7,16

Before	PSD1 020001FA	GPR7 12345678
After	PSD1 020001FD	GPR7 56781234

NONBASE REGISTER MODE EXAMPLE

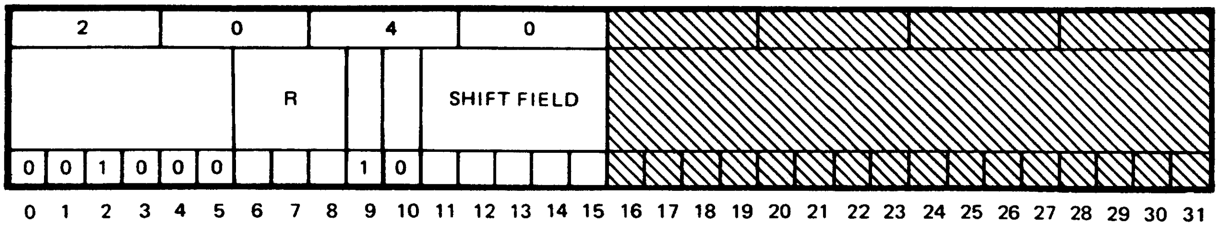
The contents of GPR7 are shifted left circular for 16_{10} bit positions.

Memory Location:	001FA
Hexadecimal Instruction :	77DO (R=7, shift field = 16_{10})
Assembly Language Coding:	SLC 7,16

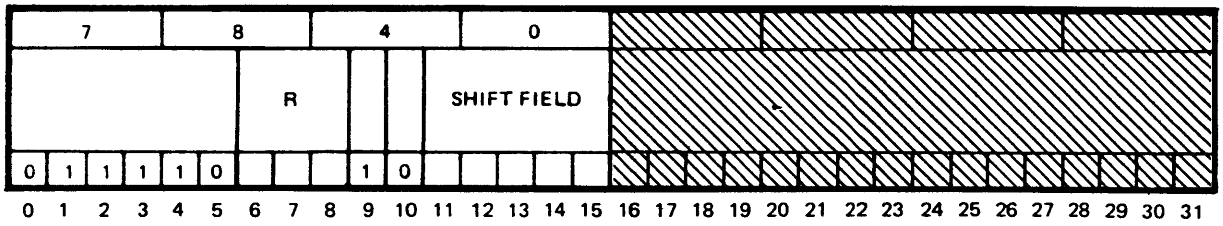
Before	PSD1 000001FA	GPR7 12345678
After	PSD1 000001FD	GPR7 56781234

SHIFT LEFT ARITHMETIC DOUBLE

SLAD
d,v



BASE REGISTER FORMAT (2040)



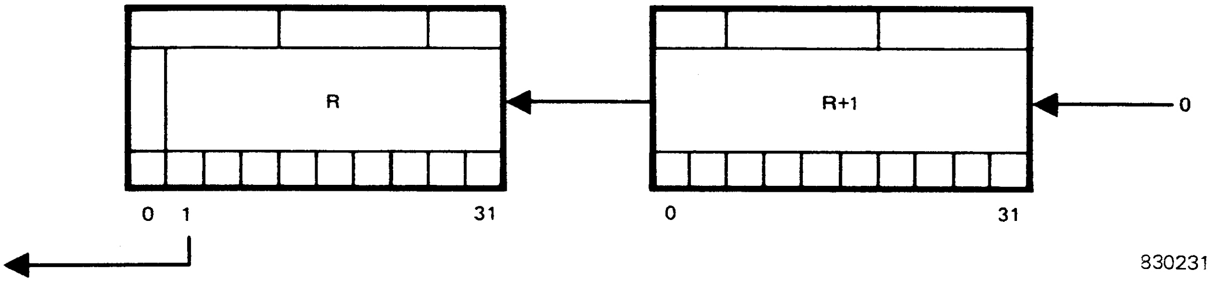
NONBASE REGISTER FORMAT (7840)

DEFINITION

The doubleword in the general purpose registers (GPR) specified by R and R+1 is shifted left the number of bit positions specified by the shift field (bits 11-15) in the instruction word. The doubleword is zero filled from the right as bits are shifted to the left. R+1 is the GPR one greater than specified by R. The sign (bit 0) of the GPR specified by R remains unchanged. Condition code bit 1 is set to one if any bit shifted out of position 1 differs from the sign bit, position 0.

NOTE

The GPR specified by R must be an even-numbered register.



CONDITION CODE RESULTS

CC1: Is set if arithmetic exception occurs
CC2: Always zero
CC3: Always zero
CC4: Always zero

BASE REGISTER MODE EXAMPLE

The doubleword contents of GPR4 and GPR5 are left shifted 24_{10} bit positions, and zero filled from the right.

Memory Location:	002DF6
Hexadecimal Instruction:	2258 (R=4, shift field =24_{10})
Assembly Language Coding:	SLAD 4,24

Before	PSD1 82002DF6	GPR4 FFFFFFA3	GPR5 9A178802
After	PSD1 82002DF9	GPR4 A39A1788	GPR5 02000000

NONBASE REGISTER MODE EXAMPLE

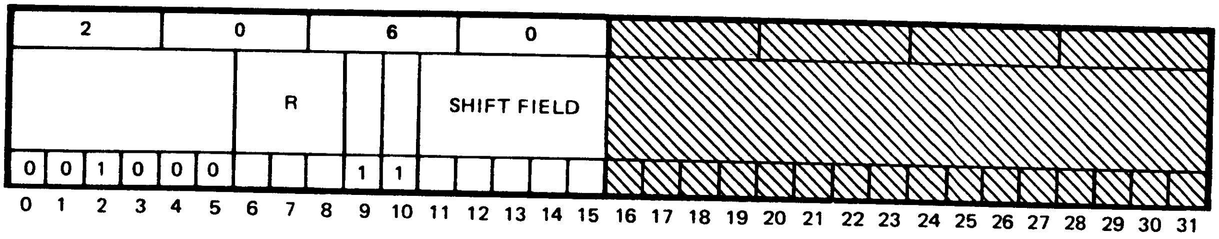
The doubleword contents of GPR4 and GPR5 are left-shifted 24_{10} bit positions, and zero filled from the right.

Memory Location:	02DF6
Hexadecimal Instruction:	7A 58 (R=4, shift field = 24_{10})
Assembly Language Coding:	SLAD 4,24

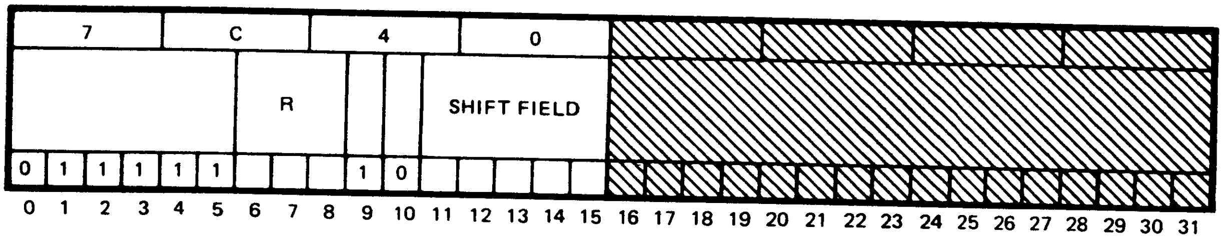
Before	PSD1 80002DF6	GPR4 FFFFFFA3	GPR5 9A178802
After	PSD1 80002DF9	GPR4 A39A1788	GPR5 02000000

SHIFT LEFT LOGICAL DOUBLE

SLLD
d,v



BASE REGISTER FORMAT (2060)



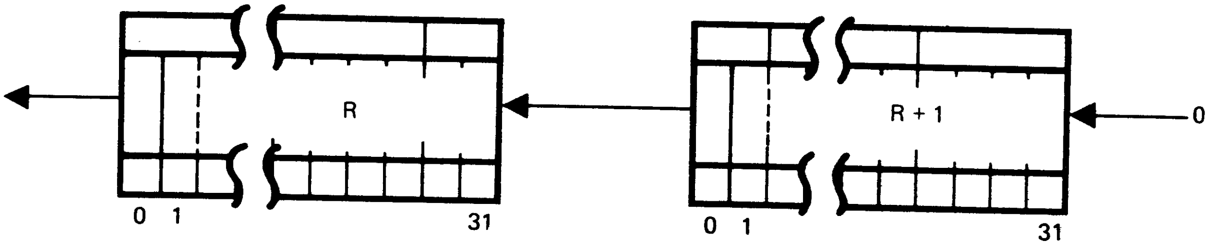
NONBASE REGISTER FORMAT (7C40)

DEFINITION

The doubleword in the general purpose register (GPR) specified by R and R+1 is shifted left the number of bit positions specified by the shift field (bits 11-15) in the instruction word. The doubleword is zero filled from the right as it is shifted to the left. R+1 is the GPR one greater than specified by R.

NOTE

The GPR specified by R must be an even-numbered register.



830232A

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

BASE REGISTER MODE EXAMPLE

The doubleword contents of GPR6 and GPR7 are left shifted 24_{10} bit positions, and zero filled from the right.

Memory Location:	001FE
Hexadecimal Instruction:	2378 (R=6, shift field = 24_{10})
Assembly Language Coding:	SLLD 6,24

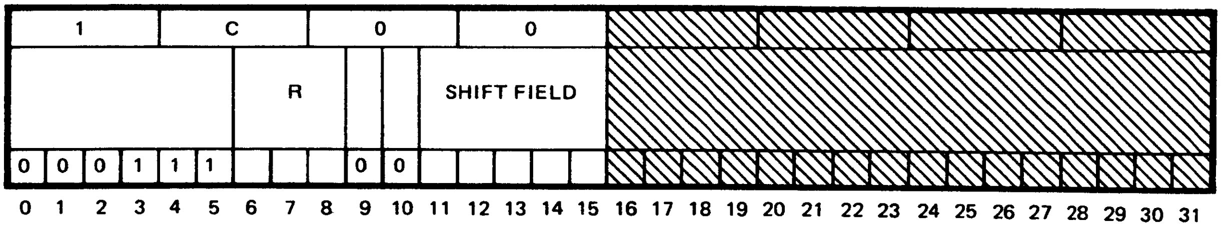
Before	PSD1 120001FE	GPR6 01234567	GPR7 89ABCDEF
After	PSD1 12000201	GPR6 6789ABCD	GPR7 EF000000

NONBASE REGISTER MODE EXAMPLE

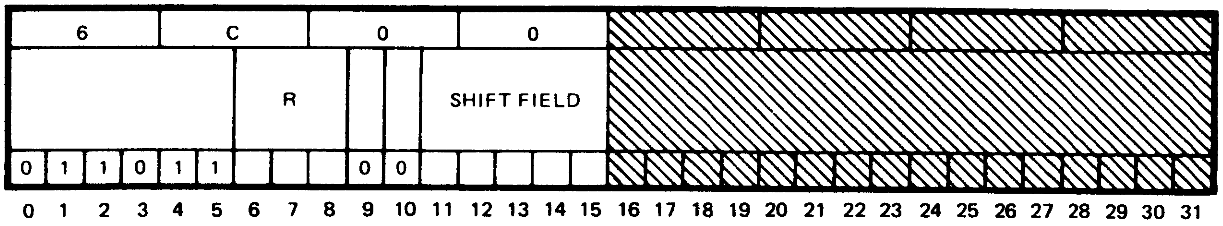
The doubleword contents of GPR6 and GPR7 are left-shifted 24_{10} bit positions, and zero filled from the right.

Memory Location:	001FE
Hexadecimal Instruction:	7F 58 (R=6, shift field=24_{10})
Assembly Language Coding:	SLLD 6,24

Before	PSD1 100001FE	GPR4 01234567	GPR7 89ABCDEF
After	PSD1 10000201	GPR6 6789ABCD	GPR7 EF000000



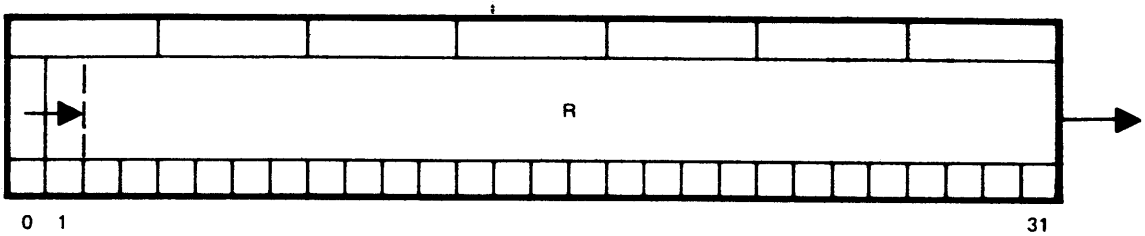
BASE REGISTER FORMAT (1C00)



NONBASE REGISTER FORMAT (6C00)

DEFINITION

The word in the general purpose register (GPR) specified by R is shifted right the number of bit positions specified by the shift field (bits 11-15) in the instruction word. The contents of bit position 0 (sign bit) is shifted into bit position 1 on each shift. The sign bit remains unchanged.



CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

BASE REGISTER MODE EXAMPLE

The contents of GPR4 are shifted right 10_{10} bit positions. Since that value is negative, a one is entered into bit position 1 with each shift.

Memory Location:		00372
Hexadecimal Instruction:		1E0A (R=4, shift field = 10_{10})
Assembly Language Coding:		SRA 4,10
Before	PSD1 12000372	GPR4 B69825F1
After	PSD1 12000375	GPR4 FFEDA609

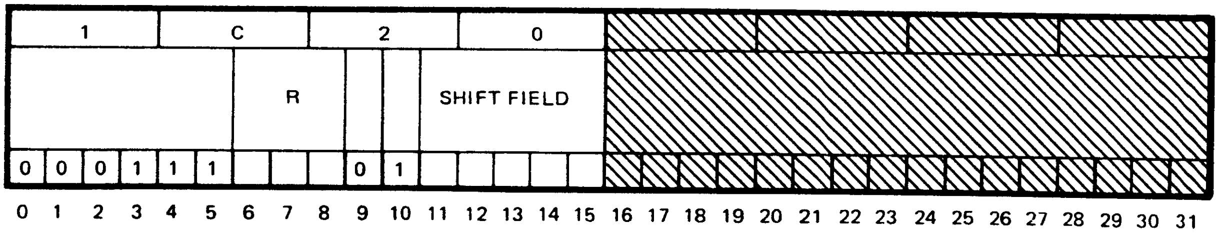
NONBASE REGISTER MODE EXAMPLE

The contents of GPR4 are shifted right 10_{10} bit positions. Since that value is negative, a one is entered into bit position 1 with each shift.

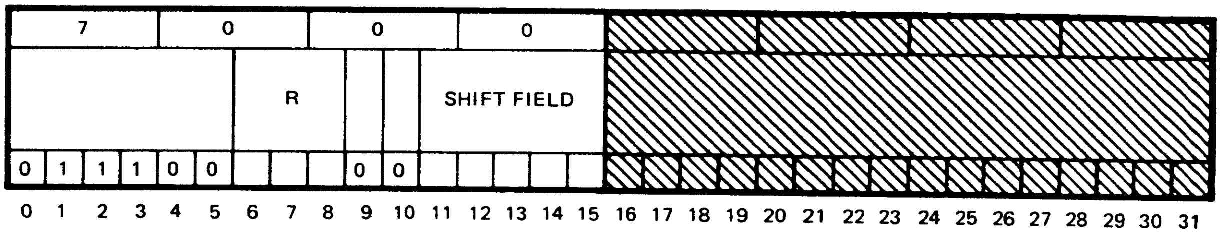
Memory Location:		00372
Hexadecimal Instruction:		6E0A (R=4, shift field=10_{10})
Assembly Language Coding:		SRA 4,10
Before	PSD1 10000372	GPR4 B69825F1
After	PSD1 10000375	GPR4 FFEDA609

SHIFT RIGHT LOGICAL

SRL
d,v



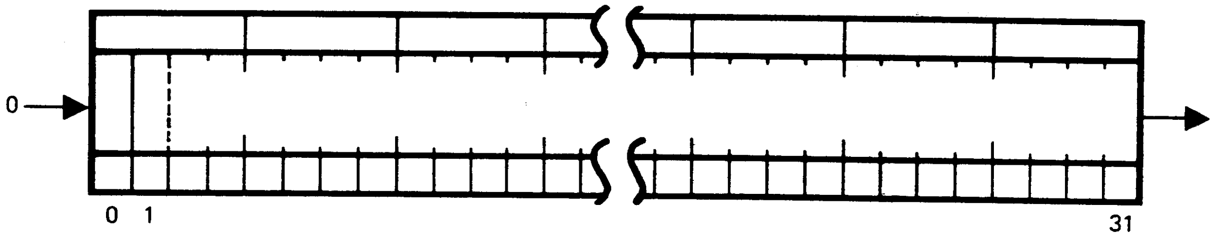
BASE REGISTER FORMAT (1C20)



NONBASE REGISTER FORMAT (7000)

DEFINITION

The word in the general purpose register (GPR) specified by R is shifted right the number of bit positions specified by the shift field (bits 11-15) in the instruction word. The resultant word is zero filled from the left as bits are shifted to the right.



320234A

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

BASE REGISTER MODE EXAMPLE

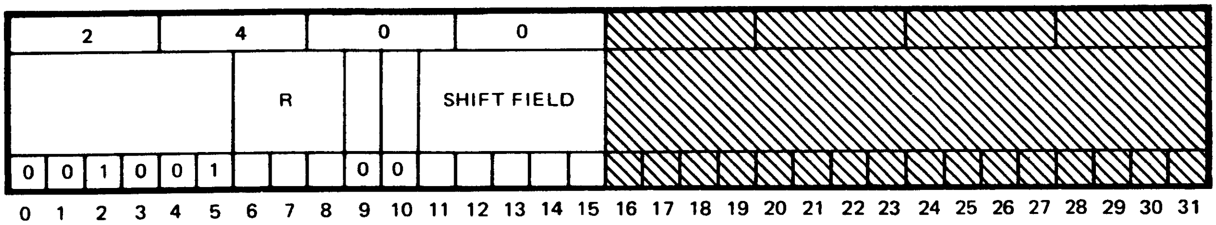
The contents of GPR4 are shifted right 10_{10} bit positions, and zero filled from the left.

Memory Location:		00372
Hexadecimal Instruction:		1E2A (R=4, shift field = 10_{10})
Assembly Language Coding:		SRL 4,10
Before	PSD1 12000372	GPR4 B69825F1
After	PSD1 12000375	GPR4 002DA609

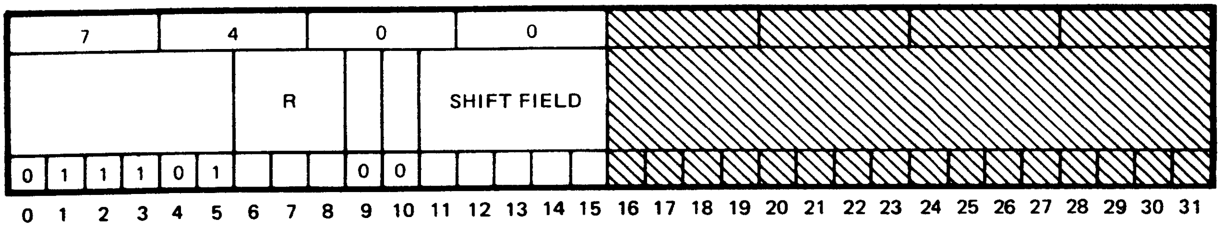
NONBASE REGISTER MODE EXAMPLE

The contents of GPR4 are shifted right 10_{10} bit positions, and zero filled from the left.

Memory Location:		00372
Hexadecimal Instruction:		72 0A (R=4, shift field=10_{10})
Assembly Language Coding:		SRL 4,10
Before	PSD1 10000372	GPR4 B69825F1
After	PSD1 10000375	GPR4 002DA609



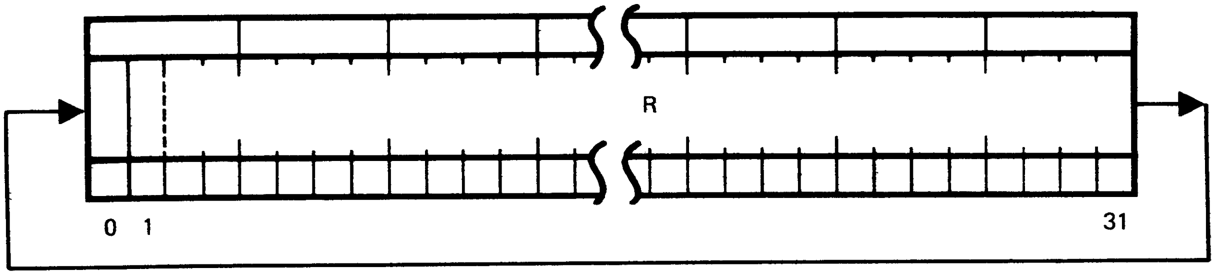
BASE REGISTER FORMAT (2400)



NONBASE REGISTER FORMAT (7400)

DEFINITION

The word in the general purpose register (GPR) specified by R is shifted right the number of bit positions specified by the shift field (bits 11-15) in the instruction word. Bits shifted out of bit position 31 are shifted into bit position 0.



830235A

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

BASE REGISTER MODE EXAMPLE

The contents of GPR4 are shifted right circular 12_{10} bit positions.

Memory Location:	00372
Hexadecimal Instruction:	260C (R=4, shift field = 12_{10})
Assembly Language Coding:	SRC 4,12

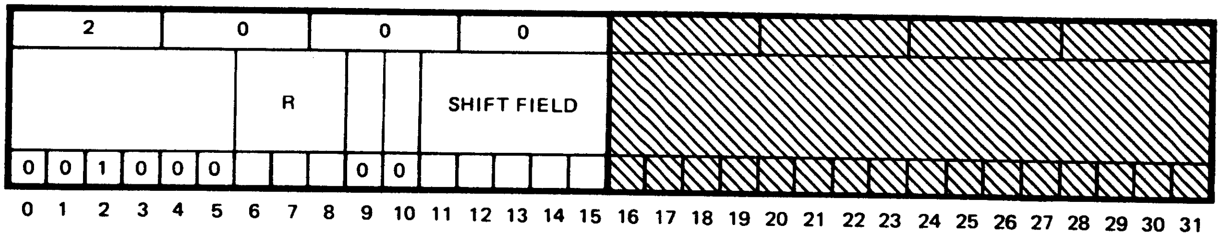
Before	PSD1 22000372	GPR4 01234567
After	PSD1 22000375	GPR4 56701234

NONBASE REGISTER MODE EXAMPLE

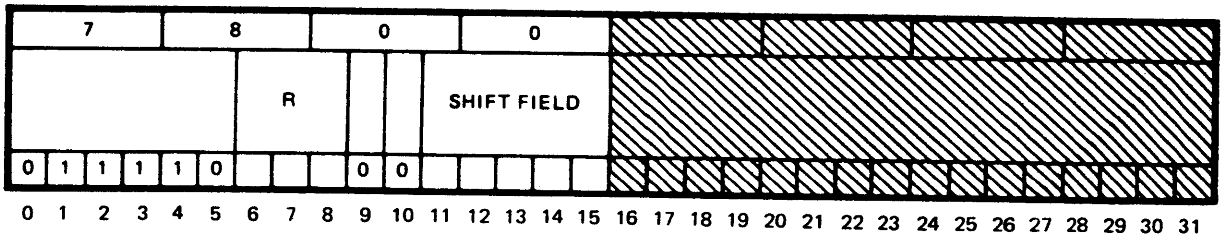
The contents of GPR4 are shifted right circular 12_{10} bit positions.

Memory Location:	00372
Hexadecimal Instruction:	76 0C (R=4, shift field=12_{10})
Assembly Language Coding:	SRC 4,12

Before	PSD1 20000372	GPR4 01234567
After	PSD1 20000375	GPR4 56701234



BASE REGISTER FORMAT (2000)



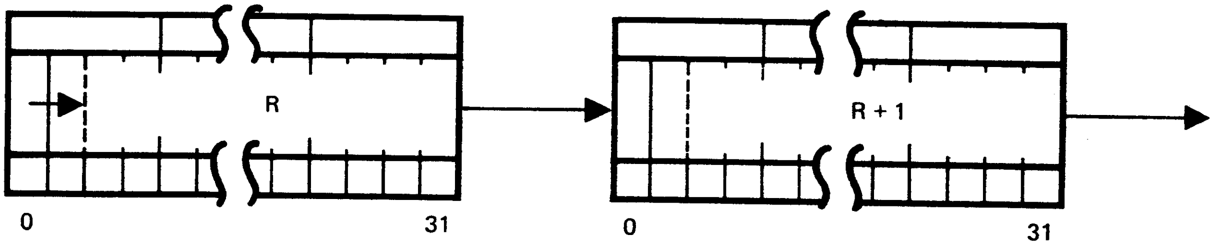
NONBASE REGISTER FORMAT (7800)

DEFINITION

The doubleword in the general purpose register (GPR) specified by R and R+1 is shifted right the number of bit positions specified by the shift field (bits 11-15) in the instruction word. R+1 is the GPR one greater than specified by R. The contents of bit position 0 of the GPR specified by R (sign bit) is shifted into bit position 1 on each shift. The sign (bit 0) of the GPR specified by R remains unchanged.

NOTE

The GPR specified by R must be an even-numbered register.



830236A

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

BASE REGISTER MODE EXAMPLE

The doubleword contents of GPR6 and GPR7 are shifted right 24_{10} bit positions, and the sign bit is extended 24_{10} bits from the left.

Memory Location:	02B46
Hexadecimal Instruction:	2318 (R=6, shift field = 24_{10})
Assembly Language Coding:	SRAD 6,24

Before	PSD1 22002B46	GPR6 8E2A379B	GPR7 58C1964D
After	PSD1 22002B49	GPR6 FFFFFF8E	GPR7 2A379B58

NONBASE REGISTER MODE EXAMPLE

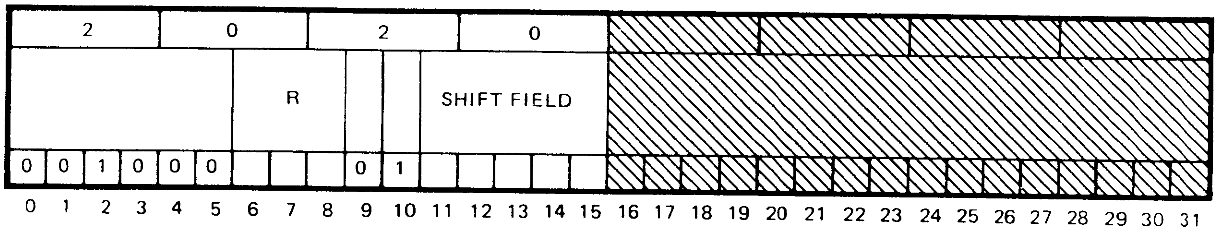
The doubleword contents of GPR6 and GPR7 are shifted right 24_{10} bit positions, and the sign bit is extended 24_{10} bits from the left.

Memory Location:	02B46
Hexadecimal Instruction:	7B 18 (R=6, shift field=24_{10})
Assembly Language Coding:	SRAD 6,24

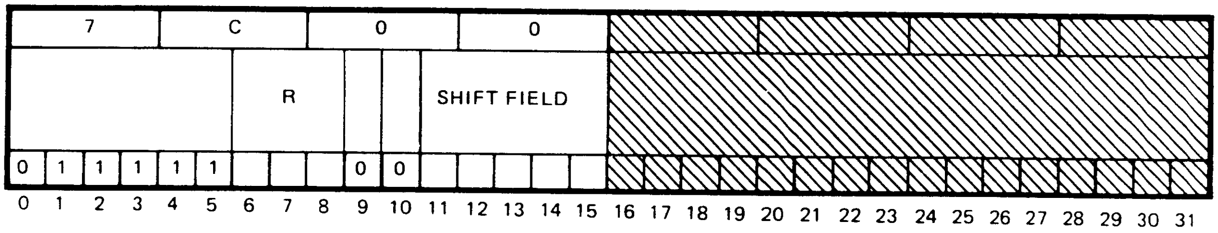
Before	PSD1 20002B46	GPR6 8E2A379B	GPR7 58C1964D
After	PSD1 20002B49	GPR6 FFFFFF8E	GPR7 2A379B58

SHIFT RIGHT LOGICAL DOUBLE

SRLD
d,v



BASE REGISTER FORMAT (2020)



830237

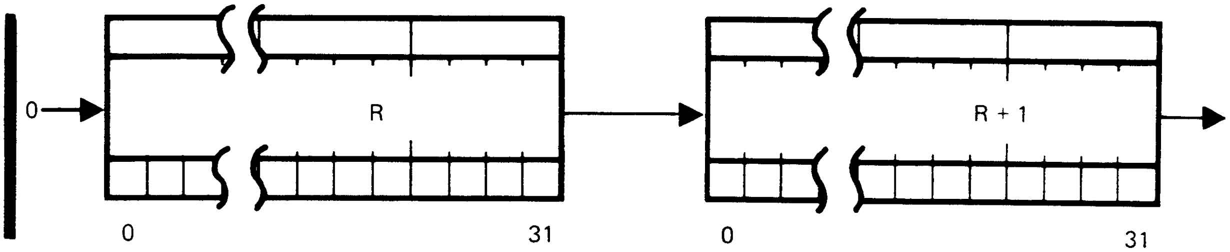
NONBASE REGISTER FORMAT (7C00)

DEFINITION

The doubleword in the general purpose register (GPR) specified by R and R+1 is shifted right the number of bit positions specified by the shift field (bits 11-15) in the instruction word. R+1 is the GPR one greater than specified by R. The resultant word is zero filled from the left.

NOTE

The GPR specified by R must be an even-numbered register.



830237-1

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

BASE REGISTER MODE EXAMPLE

The doubleword contents of GPR6 and GPR7 are shifted right 24_{10} bit positions, and zero filled from the left.

Memory Location:	02B46
Hexadecimal Instruction:	2338 (R=6, shift field = 24_{10})
Assembly Language Coding:	SRLD 6,24

Before	PSD1 22002B46	GPR6 8E2A379B	GPR7 58C1964D
After	PSD1 22002B49	GPR6 0000008E	GPR7 2A379B58

NONBASE REGISTER MODE EXAMPLE

The doubleword contents of GPR6 and GPR7 are shifted right 24_{10} bit positions, and zero filled from the left.

Memory Location:	02B46
Hexadecimal Instruction:	7F 18 (R=6, shift field=24_{10})
Assembly Language Coding:	SRLD 6,24

Before	PSD1 20002B46	GPR6 8E2A379B	GPR7 58C1964D
After	PSD1 20002B49	GPR6 0000008E	GPR7 2A379B58

6.2.8 Bit Manipulation Instructions

The bit manipulation instruction group provides the capability to set, reset, or add to a bit at a specified bit location within a specified byte of a memory location or general purpose register. Provision is made to test a bit in memory or in a general purpose register by shifting the contents of that bit position into the condition code field of the PSD.

6.2.8.1 Instruction Format

The bit manipulation instruction group uses the standard memory reference and interregister formats.

6.2.8.2 Condition Code

If the bit on which the operation is being performed is set to one before the execution of set bit, zero bit, and test bit operations, then a condition code is set. During add bit operations, a condition code is set to indicate whether the execution of the instruction caused a result greater than zero, less than zero, equal to zero, or an arithmetic exception.

6.2.8.3 Shared Memory Configurations

The shared memory environment of multiple processor systems includes those features which allow the individual processors of a multiprocessor system to communicate through a common block of memory. Specifically, these are the Read and Lock feature and the shared memory boundary feature.

Two types of shared memory configurations are common with multiprocessor systems. The first shared memory configuration is a CPU and IPU system where the multiprocessors and shared memory reside on the same bus. In this configuration each processor can monitor the SelBUS memory write activity and keep its cache up to date. The only shared memory feature that needs to be used for a CPU/IPU shared memory is the read and lock feature which is required to implement the interprocessor bit semaphores (flags) as described later. The second shared memory configuration consists of two or more SelBUSes with each bus having its own processor(s), where the multiple SelBUSes communicate to a remote multiport memory. In this configuration the processor(s) on one SelBUS operate independent of the memory activity on another SelBUS. Therefore, the processor(s) on one SelBUS cannot keep its cache(s) updated with respect to the memory write activity on another SelBUS and vice versa. To make this configuration work the processors cannot use their caches when making memory transactions within the shared memory region. To implement this function each processor is equipped with shared memory boundary registers which describe the upper and lower boundaries of the shared memory region. Any memory access within these boundaries forces a cache miss causing the processor to access the multiport shared memory device. In the remote shared memory configuration the read and lock feature must be used to implement interprocessor bit semaphores.

The shared memory features are invoked by software through the SMC instruction. The SMC instruction format provides notes and examples for implementing the shared memory features. SMC is valid for V6 only.

6.2.8.4 Interprocessor Bit Semaphores

Interprocessor semaphores are software defined memory bit flags that may enable or disable a processor access to a defined region within shared memory. Frequently, a bit flag is defined to indicate when one processor is modifying a memory region, thus prohibiting access to that region by other processors until the memory modification is complete and the bit flag cleared.

Since the bit flags may be used to control processor access to memory, it is possible that multiple processors will attempt to modify a bit flag or a different flag bit within a flag word simultaneously. If a standard read operation (without lock) of the bit manipulation instructions were allowed, confusion may result between processors, their caches, and memory. The read and lock feature eliminates this confusion. The zero bit in memory (ZBM) and the set bit in memory (SBM) instructions generate the read and lock function in a shared memory configuration. The add bit in memory (ABM) is a standard read operation because ABM should never be used for semaphore manipulation.

In a multiprocessor environment, use of the read and lock feature with the SBM and ZBM instructions prevent inadvertant destruction of bit flags. The read and lock feature functions by replacing the normal SelBUS read transaction of the SMB and ZBM transaction with a read and lock SelBUS transaction, which causes the memory module to reject any subsequent transaction (except write and unlock). For SBM and ZBM instructions, the normal write transaction is replaced by a SelBUS write and unlock transaction which unlocks the memory module and allows subsequent memory transaction to proceed.

The read and lock transaction operates differently among the different memory modules. However, the end result of the read and lock is to prevent the inadvertant destruction of flag bits on concurrent SBM or ZBM sequences in a multiprocessor environment.

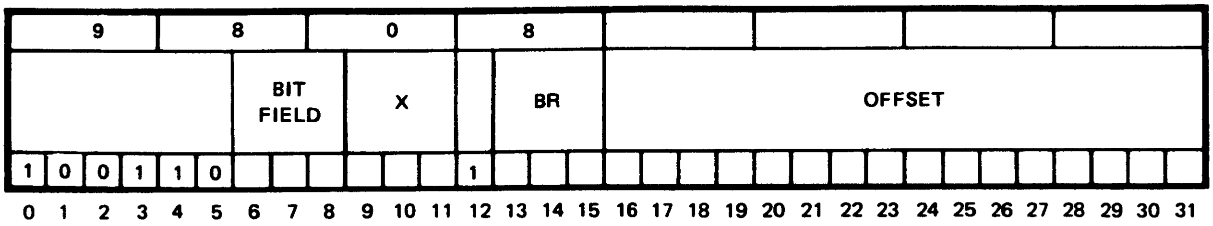
6.2.8.5 Interprocessor Semaphore Considerations

For an interprocessor semaphore scheme to be effective, the following rules and recommendations should be observed:

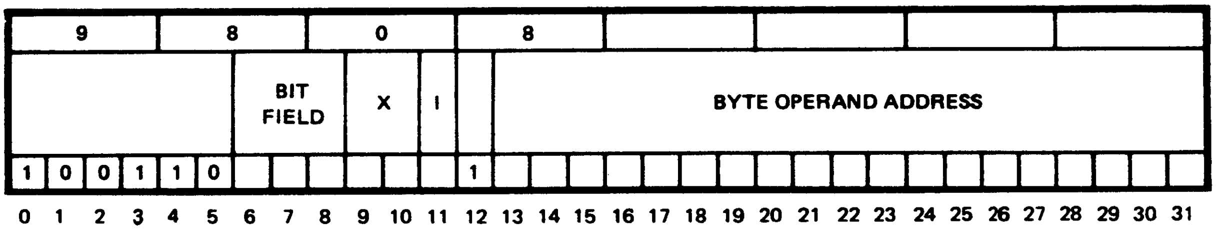
1. All processors in a multiprocessor system must use the read and lock feature.
2. A memory word containing semaphores (bit variables) must not contain other types of variables (byte, halfword etc.). Semaphore words may contain constants as long as the constant is not modified.
3. Only the SBM and ZBM instructions should be used to modify memory words containing semaphores.
4. Software must avoid the use of tight read and lock loops. If a holding loop on a semaphore state is required, the test bit in memory (TBM) instruction should be used to detect the semaphore change of state.

SET BIT IN MEMORY

SET BIT IN MEMORY **SBM**
9808 **c,*m,x**



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

830238

DEFINITION

The effective word location (EWL) containing the byte specified by the effective byte address (EBA) is fetched. The specified bit within the byte is set to one. All other bits within the byte remain unchanged. The resulting byte is replaced in the location specified by the EBA. Condition code bit 3 (CC3) is transferred to CC4, CC2 is transferred to CC3, CC1 is transferred to CC2, and the previous status of the specified bit of the byte specified by the EBA is transferred to CC1.

NOTE

> Since the contents of the condition code field of the PSD are shifted to the next highest position before the specified bit is loaded into CC1, any four bits in memory or the GPRs can be loaded into the condition code register for a combined conditional branch test.

SUMMARY EXPRESSION

$(CC3) \rightarrow CC4$
$(CC2) \rightarrow CC3$
$(CC1) \rightarrow CC2$
$(EBL_{SBL}) \rightarrow CC1$
$1 \rightarrow EBL_{SBL}$

CONDITION CODE RESULTS

CC1: Is set if EBL_{SBL} is equal to one
CC2: Is set if CC1 was one
CC3: Is set if CC2 was one
CC4: Is set if CC3 was one

BASE REGISTER MODE EXAMPLE

The contents of GPR4, BR6 and the instruction offset are added to obtain the logical address. Bit 1 of memory byte 001403 is set to one.

Memory Location:	01000
Hexadecimal Instruction:	98CE1000 (bit field = 1, BR=6, X=4)
Assembly Language Coding:	SBM 1, X '1000' (6), 4

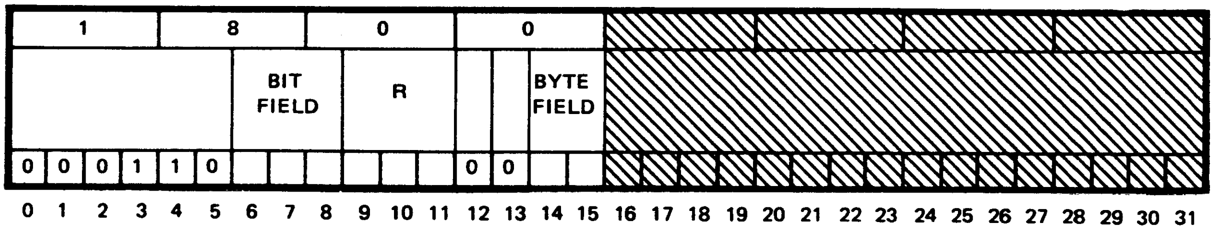
Before	PSD1	GPR4	BR6	Memory Byte 001403
	22001000	00000003	00000400	1A
After	PSD1	GPR4	BR6	Memory Byte 001403
	12001004	00000003	00000400	5A

NONBASE REGISTER MODE EXAMPLE

Bit 1 of memory byte 01403 is set to one.

Memory Location:	01000
Hexadecimal Instruction:	98 88 14 03 (bit field = 1, X=0, I=0)
Assembly Language Coding:	SBM 1,X'1403'

Before	PSD1	Memory Byte 01403
	20001000	1A
After	PSD1	Memory Byte 01403
	10001004	5A



830239

DEFINITION

The specified bit (bit field) of the specified byte (byte field) in the general purpose register (GPR) specified by R is set to one. All other bits within the GPR specified by R remain unchanged. Condition code bit 3 (CC3) is transferred to CC4, CC2 is transferred to CC3, CC1 is transferred to CC2, and the previous status of the specified bit in register R is transferred to CC1.

NOTE

> Since the contents of the condition code field of the PSD are shifted to the next highest position before the specified bit is loaded into CC1, any four bits in memory or the GPRs can be loaded into the condition code register for a combined conditional branch test.

SUMMARY EXPRESSION

$(CC3) \rightarrow CC4$
$(CC2) \rightarrow CC3$
$(CC1) \rightarrow CC2$
$(R_{SBL}) \rightarrow CC1$
$1 \rightarrow EBL_{SBL}$

CONDITON CODE RESULTS

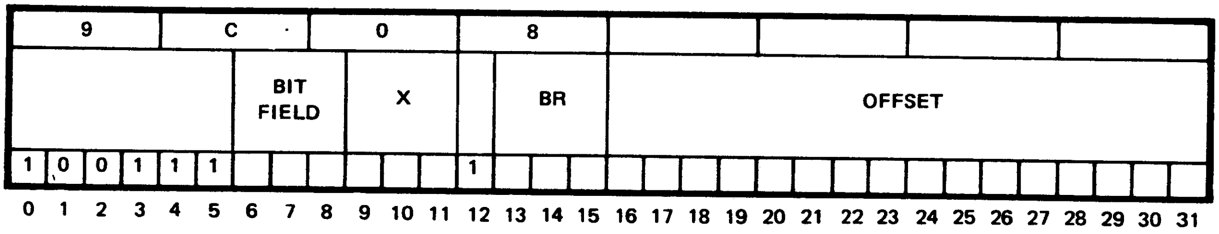
CC1: Is set if R_{SBL} is equal to one
CC2: Is set if CC1 was one
CC3: Is set if CC2 was one
CC4: Is set if CC3 was one

NONBASE AND BASE REGISTER MODE EXAMPLE

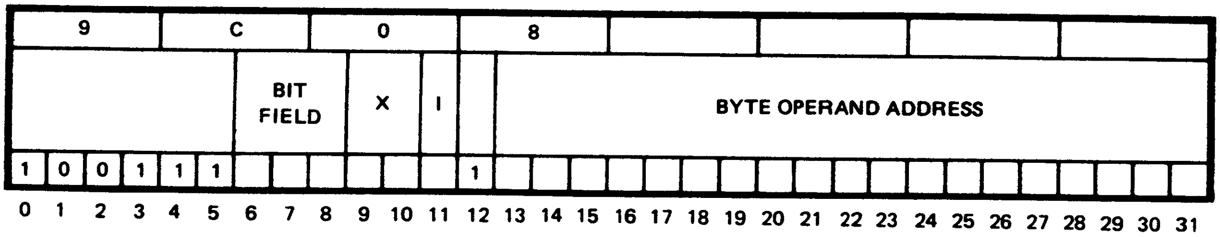
Bit 7 of byte 2 of GPR0 is set to one.

Memory Location:	01002
Hexadecimal Instruction:	1B82 (bit field=7, R=0, byte field=2)
Assembly Language Coding:	SBR 0, 23

Before	PSD1 10001002 (Nonbase) 12001002 (Base)	GPR0 0374B891
After	PSD1 08001005 (Nonbase) 0A001005 (Base)	GPR0 0374B991



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

830240

DEFINITION

The effective word location (EWL) containing the byte specified by the effective byte address (EBA) is fetched. The specified bit within the byte is reset to zero. All other bits within the byte remain unchanged. The resulting byte is transferred to the location specified by the EBA. Condition code bit 3 (CC3) is transferred to CC4, CC2 is transferred to CC3, CC1 is transferred to CC2, and the previous status of the specified bit of the byte specified by the EBA is transferred to CC1.

NOTE

> Since the contents of the condition code field of the PSD are shifted to the next highest position before the specified bit is loaded into CC1, any four bits in memory or the GPRs can be loaded into the condition code register for a combined conditional branch test.

SUMMARY EXPRESSION

$(CC3) \rightarrow CC4$
$(CC2) \rightarrow CC3$
$(CC1) \rightarrow CC2$
$(EBL_{SBL}) \rightarrow CC1$
$\text{Zero} \rightarrow EBL_{SBL}$

CONDITION CODE RESULTS

CC1: Is set if EBL_{SBL} is equal to one
CC2: Is set if CC1 was one
CC3: Is set if CC2 was one
CC4: Is set if CC3 was one

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address.

Memory Location:	1F684
Hexadecimal Instruction:	9E8E0123 (bit field = 5, X=0, BR=6)
Assembly Language Coding:	ZBM 5,X'0123'

Before	PSD1	BR6	Memory Byte 002123
	1201F684	00002000	34
After	PSD1	BR6	Memory Byte 002123
	4A01F688	00002000	30

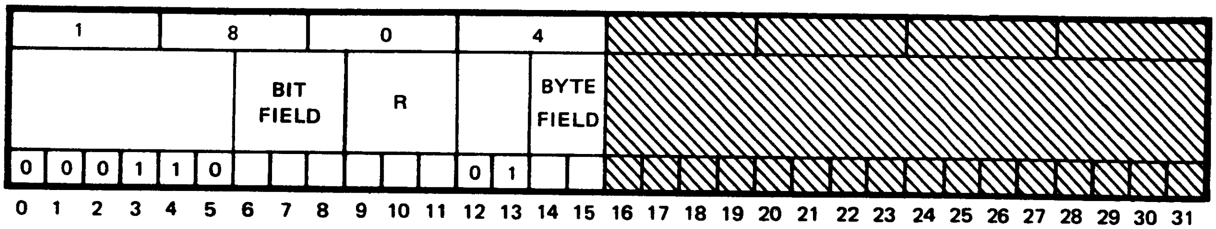
NONBASE REGISTER MODE EXAMPLE

Memory Location:	1F684
Hexadecimal Instruction:	9E 8A 01 23 (bit field = 5, X=0, I=0)
Assembly Language Coding:	ZBM 5,X'20123'

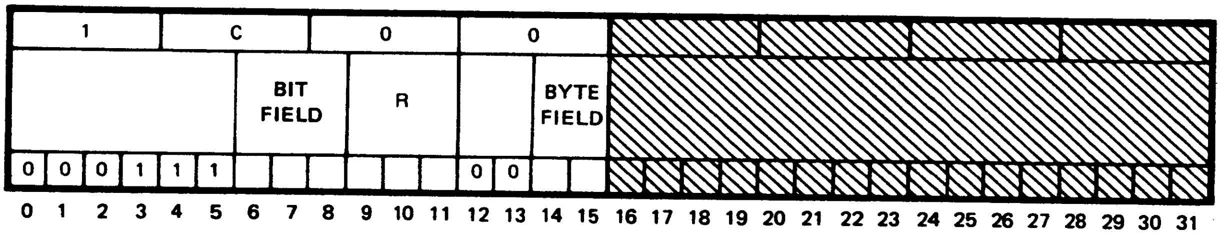
Before	PSD1	Memory Byte 20123
	1001F684	34
After	PSD1	Memory Byte 20123
	4801F688	30

ZERO BIT IN REGISTER

ZBR
d,b



BASE REGISTER FORMAT (1804)



NONBASE REGISTER FORMAT (1C00)

830241

DEFINITION

The specified bit (bit field) of the specified byte (byte field) in the general purpose register (GPR) specified by R is reset to zero. All other bits within the GPR specified by R remain unchanged. Condition code bit 3 (CC3) is transferred to CC4, CC2 is transferred to CC3, CC1 is transferred to CC2, and the previous status of the specified bit in register R is transferred to CC1.

NOTE

> Since the contents of the condition code field of the PSD are shifted to the next highest position before the specified bit is loaded into CC1, any four bits in memory or the GPRs can be loaded into the condition code register for a combined conditional branch test.

SUMMARY EXPRESSION

(CC3) → CC4
(CC2) → CC3
(CC1) → CC2
$(R_{SBL}) \rightarrow CC1$
Zero → EBL_{SBL}

CONDITION CODE RESULTS

CC1: Is set if R_{SBL} is equal to one
CC2: Is set if CC1 was one
CC3: Is set if CC2 was one
CC4: Is set if CC3 was one

BASE REGISTER MODE EXAMPLE

Bit 0 of byte 1 of GPR5 is reset to zero. CC1 is set and CC3 is transferred to CC4.

Memory Location:	00C56
Hexadecimal Instruction:	1855 (bit field=0, R=5, byte field=1)
Assembly Language Coding:	ZBR 5, 8

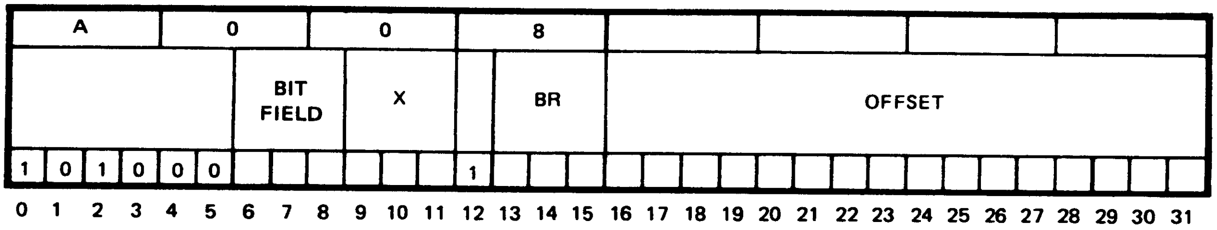
Before	PSD1 12000C56	GPR5 76A43B19
After	PSD1 4A000C59	GPR5 76243B19

NONBASE REGISTER MODE EXAMPLE

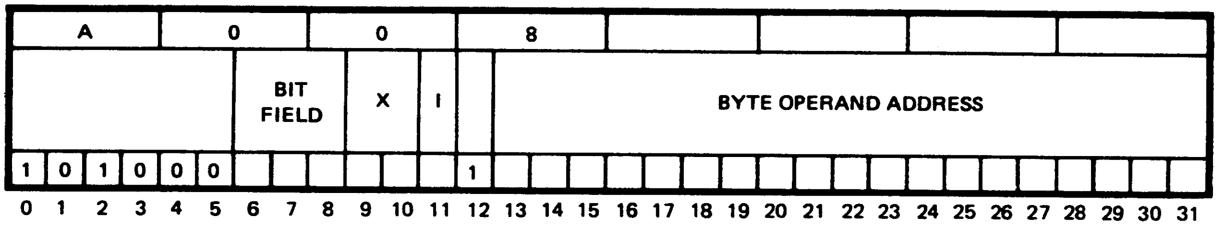
Bit 0 of byte 1 of GPR5 is reset to zero. CC1 is set and CC3 is transferred to CC4.

Memory Location:	00C56
Hexadecimal Instruction:	1C51 (bit field=0, R=5, byte field=1)
Assembly Language Coding:	ZBR 5,8

Before	PSD1 10000C56	GPR5 76A43B19
After	PSD1 48000C59	GPR5 76243B19



BASE REGISTER FORMAT



830242

NONBASE REGISTER FORMAT

DEFINITION

The effective word location (EWL) containing the byte specified by the effective byte address (EBA) is fetched from memory. A one is added to the bit position within the word specified by the byte field (C bits) and the bit field. The addition is performed on the entire word. Therefore, a carry may be propagated left to the sign bit. The resulting word is transferred to the EWL specified by the EWA.

NOTE

This instruction does not have read and lock capability.

SUMMARY EXPRESSION

$(EWL)+1_{SBL} \rightarrow EWL$

CONDITION CODE RESULTS

CC1: Is set if arithmetic exception occurs
CC2: Is set if (EWL) is greater than zero
CC3: Is set if (EWL) is less than zero
CC4: Is set if (EWL) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. A one is added to bit position 20 of memory word 003190 (byte 2, bit 4) which propagates a carry left to bit position 13. The result is returned to memory word 003190. CC2 is set.

Memory Location:	03000
Hexadecimal Instruction:	A20E3102 (bit field = 4, X=0, BR=6)
Assembly Language Coding:	ABM 4,X'3102'(6)

Before	PSD1 02003000	BR6 00000090	Memory Word 003190 0003F8F9
After	PSD1 22003004	BR6 00000090	Memory Word 003190 000400F9

NONBASE REGISTER MODE EXAMPLE

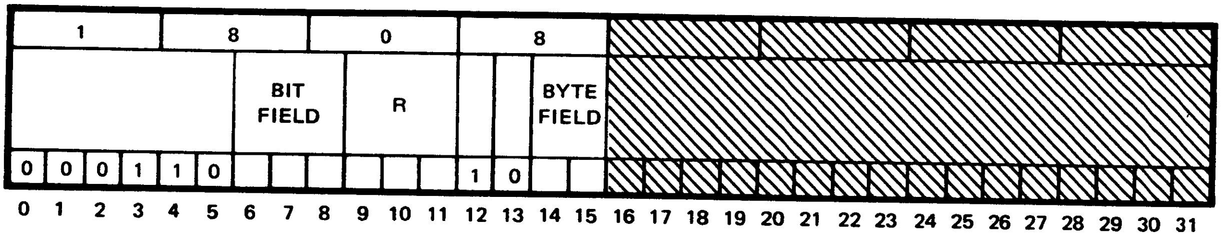
A one is added to bit position 20 of memory word 03190 (byte 2, bit 4) which propagates a carry left to bit position 13. The result is returned to memory word 03190. CC2 is set.

Memory Location:	03000
Hexadecimal Instruction:	A2 08 31 92 (bit field=4, X=0, I=0)
Assembly Language Coding:	ABM 4,X'3192'

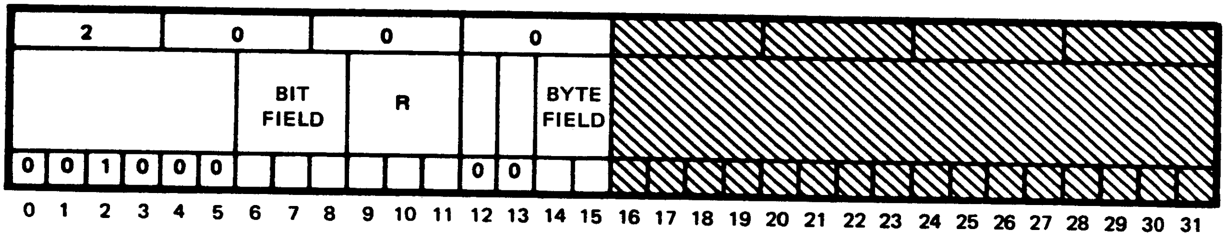
Before	PSD1 00003000	Memory Word 03190 0003F8F9
After	PSD1 20003004	Memory Word 03190 000400F9

ADD BIT IN REGISTER

ABR
d,b



BASE REGISTER FORMAT (1808)



NONBASE REGISTER FORMAT (2000)

DEFINITION

A one is added to the specified bit location (SBL) of the specified byte (byte field) in the general purpose register (GPR) specified by R. The addition is performed on the entire word of the GPR specified by R. Therefore, a carry may be propagated left to the sign bit. The result is then transferred to the GPR specified by R.

SUMMARY EXPRESSION

$(R)+1_{SBL} \rightarrow R$

CONDITION CODE RESULTS

CC1: Is set if arithmetic exception occurs
CC2: Is set if (R_{0-31}) is greater than zero
CC3: Is set if (R_{0-31}) is less than zero
CC4: Is set if (R_{0-31}) is equal to zero

BASE REGISTER MODE EXAMPLE

A one is added to bit position 2 of byte 1 of GPR6, and the result is replaced in GPR6. CC2 is set.

Memory Location:		0184E
Hexadecimal Instruction:		1969 (bit field =2, R=6, byte field =1)
Assembly Language Coding:		ABR 6,10
Before	PSD1 0A00184E	GPR6 3BE9AC48
After	PSD1 22001851	GPR6 3C09AC48

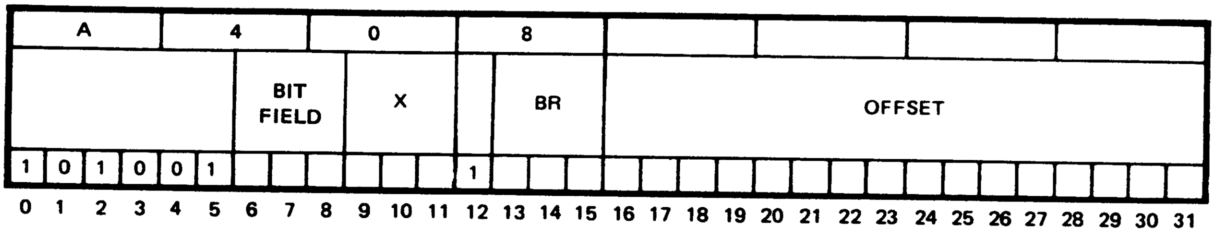
NONBASE REGISTER MODE EXAMPLE

A one is added to bit position 2 of byte 1 of GPR6, and the result is replaced in GPR6. CC2 is set.

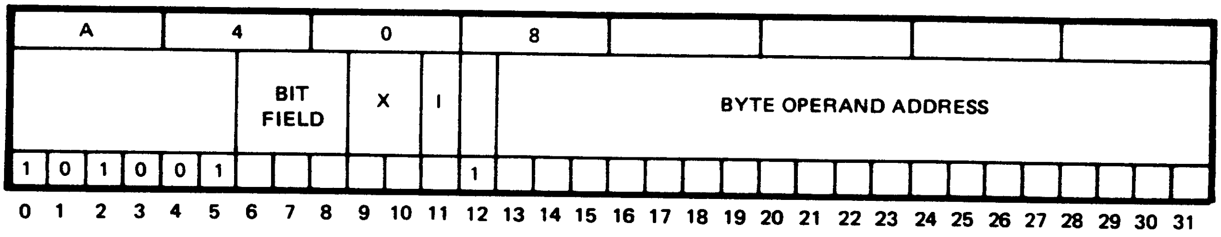
Memory Location:		0184E
Hexadecimal Instruction:		21 61 (bit field=2, R=6, byte field=1)
Assembly Language Coding:		ABR 6,10
Before	PSD1 0800184E	GPR6 3BE9AC48
After	PSD1 20001851	GPR6 3C09AC48

TEST BIT IN MEMORY A408

TBM
c,*m,x



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

DEFINITION

The specified bit location (SBL) of the effective byte location (EBL) is shifted into the condition code field of the PSD. Condition code bit 3 (CC3) is transferred to CC4, CC2 is transferred to CC3, CC1 is transferred to CC2, and the specified bit (bit field) of the byte specified by the effective byte address (EBA) is transferred to CC1.

NOTE

> Since the contents of the condition code field are shifted to the next highest position before the specified bit is loaded into CC1, any four bits in memory or the general purpose registers (GPR) can be loaded into the condition code register for a combined conditional branch test.

SUMMARY EXPRESSION

(CC3) → CC4
(CC2) → CC3
(CC1) → CC2
$(EBL_{SBL}) \rightarrow$ CC1

CONDITION CODE RESULTS

CC1: Is set if (R_{SBL}) is equal to one
CC2: Is set if (CC1) was equal to one
CC3: Is set if (CC2) was equal to one
CC4: Is set if (CC3) was equal to one

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. Bit 4 of memory byte 065B23 is transferred to CC1. CC3 is transferred to CC4.

Memory Location:	05A38
Hexadecimal Instruction:	A60E5B23 (bit field = 4, X=0, BR=6)
Assembly Language Coding:	TBM 4,X'5B23' (6)

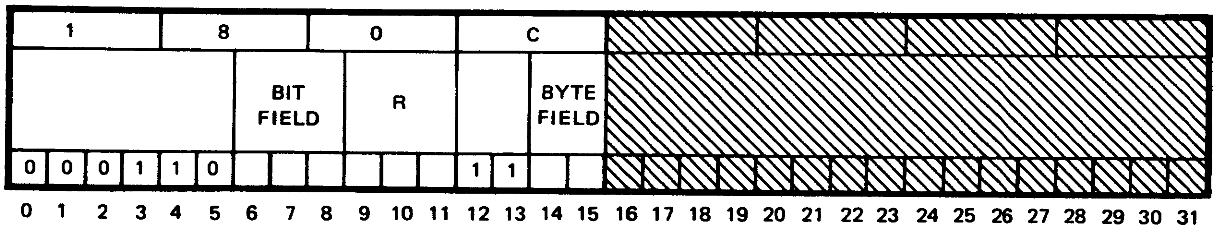
Before	PSD1 12005A38	BR6 00060000	Memory Byte 065B23 29
After	PSD1 4A005A3C	BR6 00060000	Memory Byte 065B23 29

NONBASE REGISTER MODE EXAMPLE

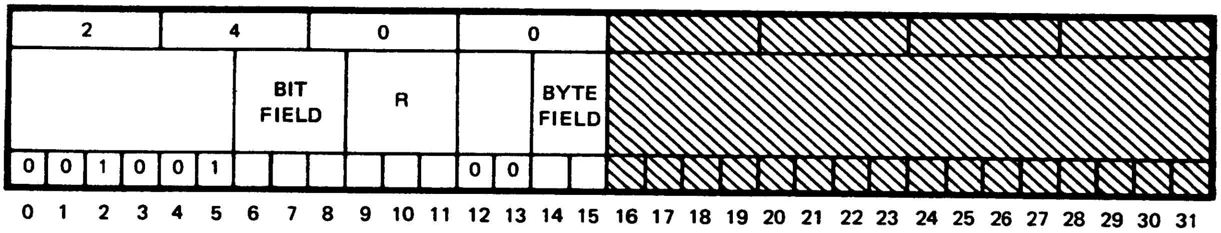
Bit 4 of memory byte 05B23 is transferred to CC1. CC3 is transferred to CC4.

Memory Location:	05A38
Hexadecimal Instruction:	A6 08 5B 23 (bit field=4, X=0, I=0)
Assembly Language Coding:	TBM 4,X'5B23'

Before	PSD1 10005A38	Memory Byte 05B23 29
After	PSD1 48005A3C	Memory Byte 05B23 29



BASE REGISTER FORMAT (180C)



NONBASE REGISTER FORMAT (2400)

830245

DEFINITION

The specified bit in the general purpose register (GPR) specified by R is shifted into the condition code field of the PSD. Condition code bit 3 (CC3) is transferred to CC4, CC2 is transferred to CC3, CC1 is transferred to CC2, and the specified bit (bit field) of the specified byte (byte field) in the GPR specified by R is transferred to CC1.

NOTE

> Since the contents of the condition code field are shifted to the next highest position before the specified bit is loaded into CC1, any four bits in memory or the GPRs can be loaded into the condition code register for a combined conditional branch test.

SUMMARY EXPRESSION

(CC3) → CC4
(CC2) → CC3
(CC1) → CC2
(R_{SBL}) → CC1

CONDITION CODE RESULTS

CC1: Is set if (R_{SBL}) was equal to one
CC2: Is set if (CC1) was equal to one
CC3: Is set if (CC2) was equal to one
CC4: Is set if (CC3) was equal to one

BASE REGISTER MODE EXAMPLE

CC2 through CC4 are right shifted one bit position. CC1 is reset to zero since bit 27 of GPR5 is zero.

Memory Location:	01982
Hexadecimal Instruction:	19DF (bit field=3, R=5, byte field=3)
Assembly Language Coding:	TBR 5, 27

Before	PSD1 1A001982	GPR5 81A2C64D
After	PSD1 0A001985	GPR5 81A2C64D

NONBASE REGISTER MODE EXAMPLE

CC2 through CC4 are right shifted one bit position. CC1 is reset to zero since bit 27 of GPR5 is zero.

Memory Location:	01982
Hexadecimal Instruction:	25 D3 (bit field=3, R=5, byte field=3)
Assembly Language Coding:	TBR 5,27

Before	PSD1 18001982	GPR5 81A2C64D
After	PSD1 08001985	GPR5 81A2C64D

6.2.9 Fixed-Point Arithmetic Instructions

The fixed-point arithmetic instructions perform addition, subtraction, multiplication, division, and sign control functions on bytes, halfwords, words, and doublewords in memory and general purpose registers. Provisions have been made to allow the result of a register-to-register addition or subtraction to be masked before final transfer into the destination register.

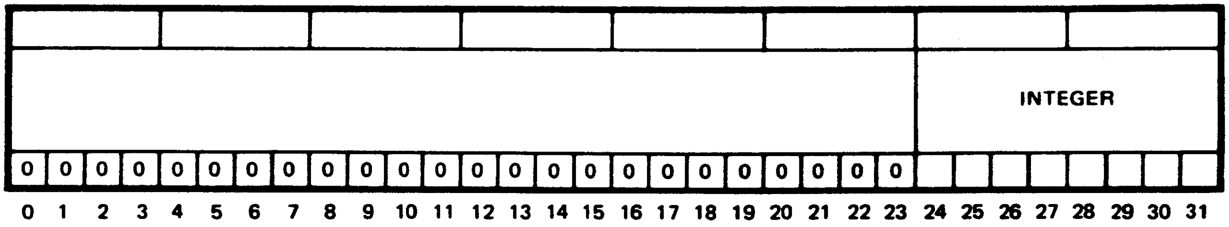
6.2.9.1 Instruction Format

The fixed-point arithmetic instructions use the memory reference, immediate, or interregister format, as applicable.

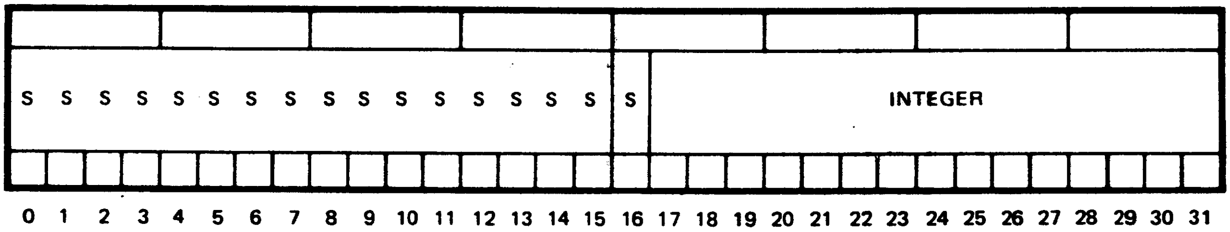
6.2.9.2 Data Formats

830359A

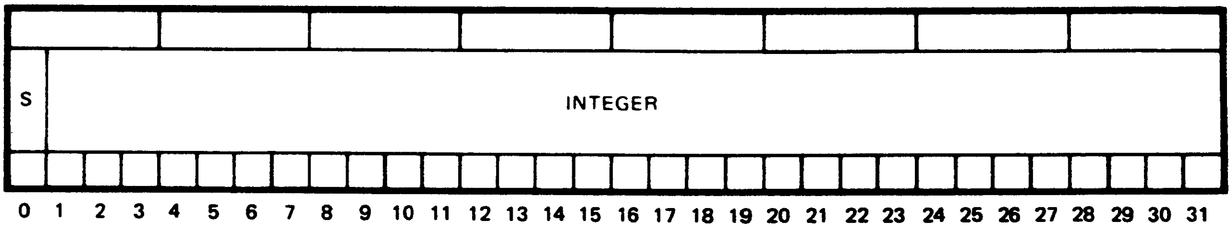
Byte



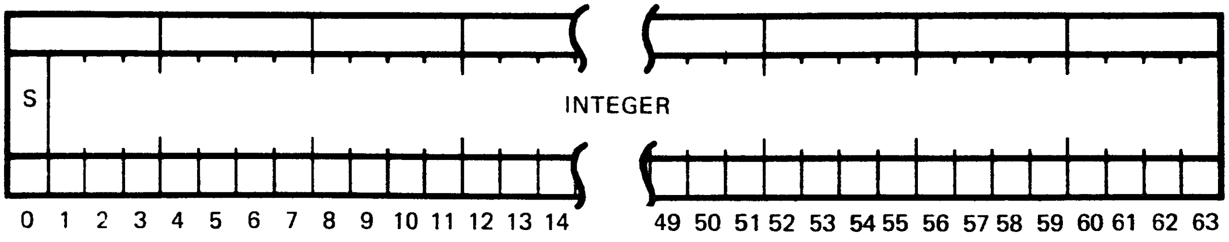
Halfword (Sign Extended)



Word



Doubleword



6.2.9.3 Handling Logical and Arithmetic Operations

In executing logical instructions, the CPU interprets operands as unsigned 32-bit words or unsigned 64-bit doublewords. However, for fixed-point arithmetic operations, the CPU recognizes arithmetic operands as signed numbers with a negative number represented by the twos complement of the absolute magnitude (except for the values of X'80000000' and X'8000000000000000' which do not have positive complements).

Numbers in the range of X'00000001' through X'7FFFFFFF' for single-word values, or X'0000000000000001' through X'7FFFFFFFFFFFFFFF' for doubleword values, are recognized as positive numbers (the sign bit is zero). Numbers in the range of X'FFFFFFFF' through X'80000000' for single-word values, or X'FFFFFFFFFFFFFFFF' through X'8000000000000000' for doubleword values, are recognized as negative numbers (the sign bit is one).

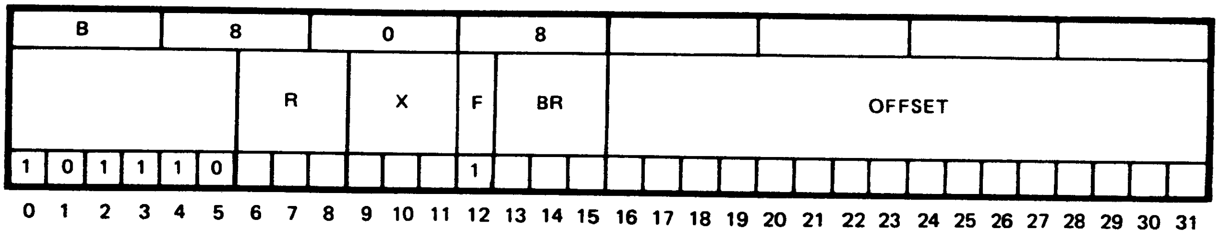
The addition of two numbers of like signs, or the subtraction of two numbers of different signs, where the carry propagates into the sign bit, will cause an arithmetic exception condition. The arithmetic exception condition results at the most-significant end of the arithmetic result, when the twos complement operation has overflowed into the sign bit. This exception condition is logically the exclusive OR of the carry-in or carry-out of the most-significant bit of the arithmetic logical unit.

6.2.9.4 Condition Code

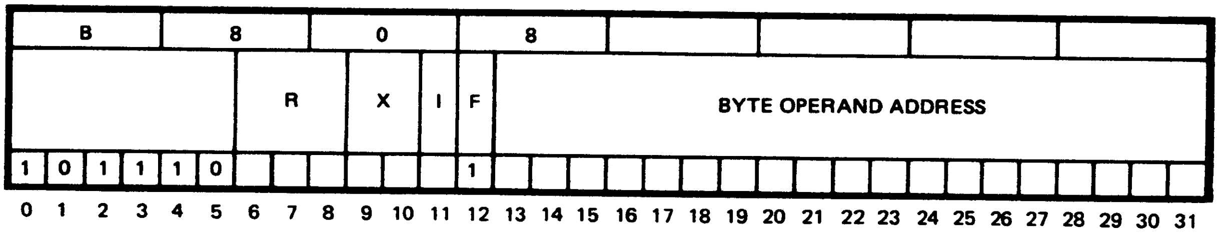
Execution of most fixed-point arithmetic instructions causes a condition code bit to be set to indicate whether the result of the operation was greater than, less than, or equal to zero. Arithmetic exceptions produced by an arithmetic operation cause condition code bit 1 (CC1) to be set.

ADD MEMORY BYTE
B808

ADMB
d,*m,x



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

830246

DEFINITION

The effective byte location (EBL) specified by the effective byte address (EBA) is accessed and 24 zeros are appended to the most-significant end to form a word. This word is algebraically added to the contents of the general purpose register (GPR) specified by R. The resulting word is then transferred to the GPR specified by R.

SUMMARY EXPRESSION

Zeros_{0-23}, (EBL)+(R) → R

CONDITION CODE RESULTS

CC1: Is set if arithmetic exception occurs
CC2: Is set if (R_{0-31}) is greater than zero
CC3: Is set if (R_{0-31}) is less than zero
CC4: Is set if (R_{0-31}) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of memory byte 001913, with zeros prefixed, are added to the contents of GPR4; the result is transferred to GPR4. CC2 is set.

Memory Location:	00800
Hexadecimal Instruction:	BA0E0913 (R=4, X=0, BR=6)
Assembly Language Coding:	ADMB 4,X'0913'(6)

Before	PSD1	GPR4	BR6	Memory Byte 001913
	12000800	00000099	00001000	8A
After	PSD1	GPR4	BR6	Memory Byte 001913
	22000804	00000123	00001000	8A

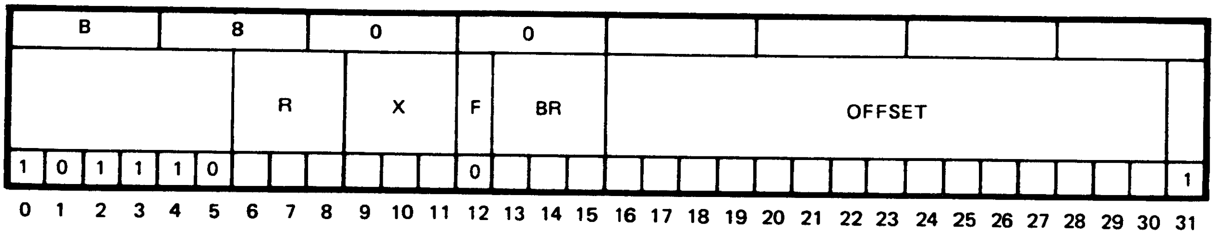
NONBASE REGISTER MODE EXAMPLE

The contents of memory byte 00915, with zeros prefixed, are added to the contents of GPR4; the result is transferred to GPR4. CC2 is set.

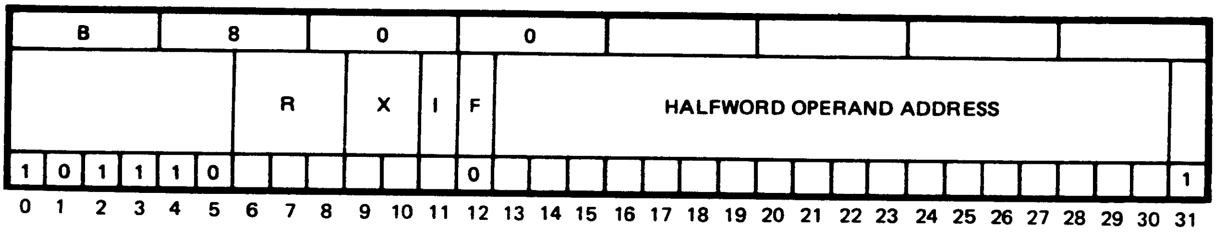
Memory Location:	00800
Hexadecimal Instruction:	BA 08 09 15 (R=4, X=0, I=0)
Assembly Language Coding:	ADMB 4,X'915'

Before	PSD1	GRP4	Memory Byte 00915
	10000800	00000099	8A
After	PSD1	GPR4	Memory Byte 00915
	20000804	00000123	8A

ADD MEMORY HALFWORD ADMH
B800 d,*m,x



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

830247

DEFINITION

The effective halfword location (EHL) specified by the effective halfword address (EHA) is accessed and the sign bit (bit 16) is extended 16 bits to the left to form a word. This word is algebraically added to the contents of the general purpose register (GPR) specified by R. The resulting word is then transferred to the GPR specified by R.

SUMMARY EXPRESSION

$(EHL)_{SE}+(R) \rightarrow R$

CONDITION CODE RESULTS

CC1: Is set if arithmetic exception occurs
CC2: Is set if (R_{0-31}) is greater than zero
CC3: Is set if (R_{0-31}) is less than zero
CC4: Is set if (R_{0-31}) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of memory halfword 001096 with sign extension are added to the contents of GPR7. The result replaces the contents of GPR7. CC3 is set.

Memory Location:	40D68
Hexadecimal Instruction:	BB861007 (R=7, X=0, BR=6)
Assembly Language Coding:	ADMH 7,X'1006'(6)

Before	PSD1	GPR7	BR6	Memory Halfword 001096
	22040D68	000006C4	00000090	8C42
After	PSD1	GPR7		BR6Memory Halfword 001096
	12040D6C	FFFF9306	00000090	8C42

NONBASE REGISTER MODE EXAMPLE

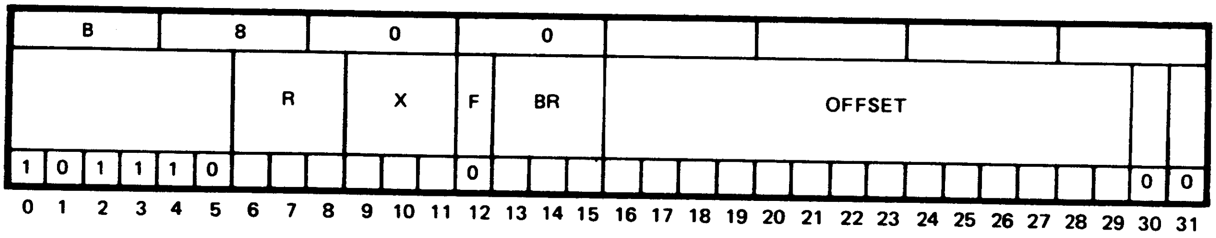
The contents of memory halfword 41096 with sign extension are added to the contents of GPR7, and the result replaces the contents of GPR7. CC3 is set.

Memory Location:	40D68
Hexadecimal Instruction:	BB 84 10 97 (R=7, X=0, I=0)
Assembly Language Coding:	ADMH 7,X'41096'

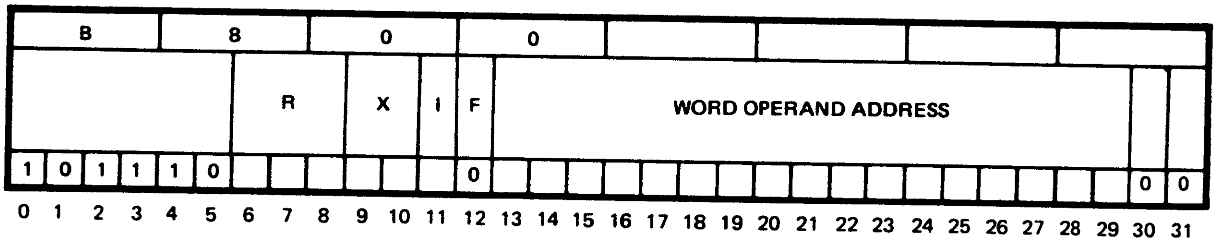
Before	PSD1	GPR7	Memory Halfword 41096
	20040D68	000006C4	8C42
After	PSD1	GPR7	Memory Halfword 41096
	10040D6C	FFFF9306	8C42

ADD MEMORY WORD
B800

ADMW
d,*m,x



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

830248

DEFINITION

The effective word location (EWL) specified by the effective word address (EWA) is accessed and algebraically added to the contents of the general purpose register (GPR) specified by R. The resulting word is then transferred to the GPR specified by R.

SUMMARY EXPRESSION

(EWL)+(R) → R

CONDITION CODE RESULTS

CC1: Is set if arithmetic exception occurs
CC2: Is set if (R_{0-31}) is greater than zero
CC3: Is set if (R_{0-31}) is less than zero
CC4: Is set if (R_{0-31}) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6, GPR4 and the instruction offset are added to obtain the logical address. The contents of memory word 0011AC are added to the contents of GPR6, the result is transferred to GPR6. CC2 is set.

Memory Location:	00D50
Hexadecimal Instruction:	BB461100 (R=6, BR=6, X=4)
Assembly Language Coding:	ADMW 6, X '1100' (6), 4

Before	PSD1	GPR6	BR6	GPR4	Memory Word 0011AC
	42000D50	0037C1F3	000000A0	0000000C	004FC276
After	PSD1	GPR6	BR6	GPR4	Memory Word 0011AC
	22000D54	00878469	000000A0	0000000C	004FC276

NONBASE REGISTER MODE EXAMPLE

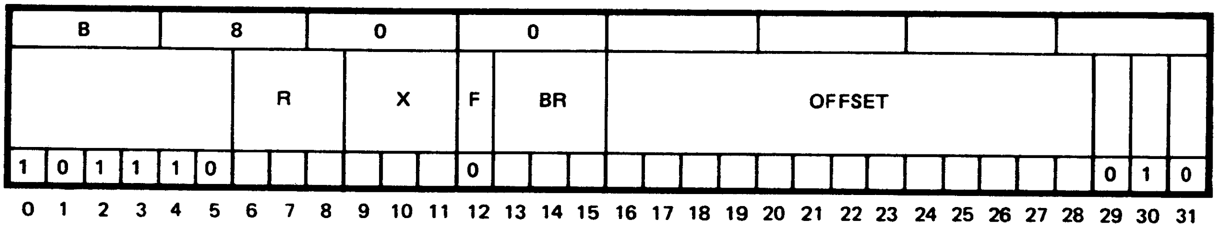
The contents of memory word 011AC are added to the contents of GPR6. The result is transferred to GPR6. CC2 is set.

Memory Location:	00D50
Hexadecimal Instruction:	BB 00 11 AC (R=6, X=0,I=0)
Assembly Language Coding:	ADMW 6,X'11AC'

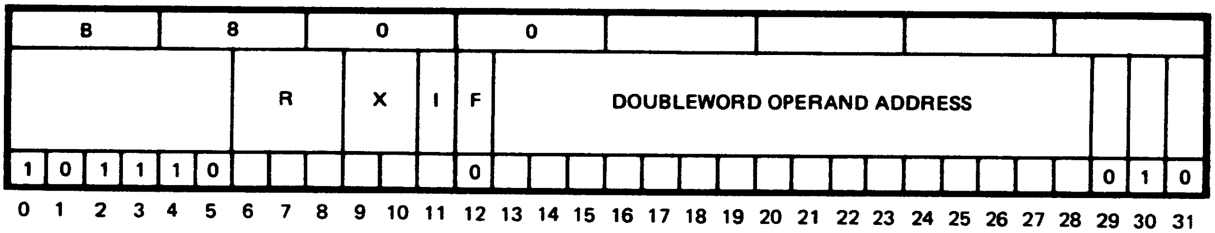
Before	PSD1	GPR6	Memory Word 011AC
	40000D50	0037C1F3	004FC276
After	PSD1	GPR6	Memory Word 011AC
	20000D54	00878469	004FC276

ADD MEMORY DOUBLEWORD B800

ADMD d,*m,x



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

830249

DEFINITION

The effective word locations (EWL, EWL+1) specified by the effective doubleword address (EDA) is accessed and algebraically added to the contents of the general purpose register (GPR) specified by R and R+1. R+1 is the GPR one greater than specified by R. The contents of the GPR specified by R+1 are added to the contents of the least-significant word of the doubleword first. The contents of the GPR specified by R are added to the contents of the most-significant word of the doubleword last. The resulting doubleword is transferred to the GPR specified by R and R+1.

NOTE

The GPR specified by R must be an even-numbered register.

SUMMARY EXPRESSION

(EWL + 1) + (R+1) → R+1 + Carry

(EWL) + (R) + Carry → R

CONDITION CODE RESULTS

CC1: Is set if arithmetic exception occurs
CC2: Is set if (R, R+1) is greater than zero
CC3: Is set if (R, R+1) is less than zero
CC4: Is set if (R, R+1) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The doubleword obtained from the contents of memory words 009250 and 009254 is added to the doubleword obtained from the contents of GPR4 and GPR5. The contents are transferred to GPR4 and GPR5. CC2 is set.

Memory Location:	08E3C
Hexadecimal Instruction:	BA069002 (R=4, X=0, BR=6)
Assembly Language Coding:	ADMD 4,X'9000'(6)

Before	PSD1	GPR4	GPR5	BR6
	0A008E3C	000298A1	815BC63E	00000250
	Memory Word 009250		Memory Word 009254	
	3B69A07E		7F3549A4	
After	PSD1	GPR4	GPR5	BR6
	22008E40	3B6C3920	00910FE2	00000250
	Memory Word 009250		Memory Word 009254	
	3B69A07E		7F3549A4	

NONBASE REGISTER MODE EXAMPLE

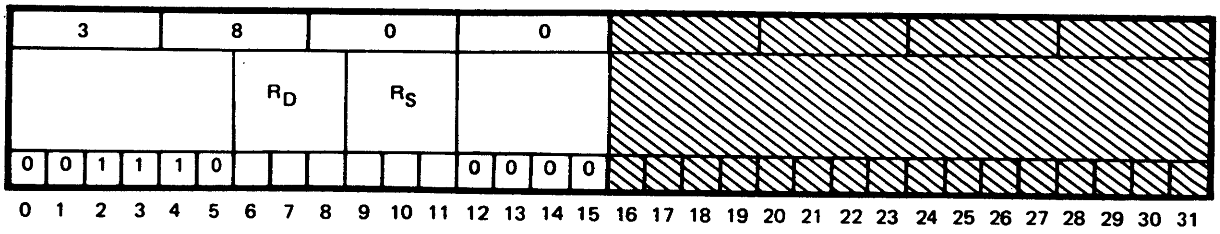
The doubleword obtained from the contents of memory words 09250 and 09254 is added to the doubleword obtained from the contents of GPR4 and GPR5. The result is transferred to GPR4 and GPR5. CC2 is set.

Memory Location:	08E3C
Hexadecimal Instruction:	BA 00 92 52 (R=4, X=0, I=0)
Assembly Language Coding:	ADMD 4,X'9250'

Before	PSD1	GPR4	GPR5
	08008E3C	000298A1	815BC63E
	Memory Word 09250		Memory Word 09254
	3B69A07E		7F3549A4
After	PSD1	GPR4	GPR5
	20008E40	3B6C3920	00910FE2
	Memory Word 09250		Memory Word 09254
	3B69A07E		7F3549A4

ADD REGISTER TO REGISTER
3800

ADR
s,d



830250

DEFINITION

The word in the general purpose register (GPR) specified by R_D is algebraically added to the word in the GPR specified by R_S. The resulting word is then transferred to the GPR specified by R_D.

SUMMARY EXPRESSION

$(R_S+R_D) \rightarrow R_D$

CONDITION CODE RESULTS

CC1: Is set if arithmetic exception occurs
CC2: Is set if (R_D) is greater than zero
CC3: Is set if (R_D) is less than zero
CC4: Is set if (R_D) is equal to zero

NONBASE AND BASE REGISTER MODE EXAMPLE

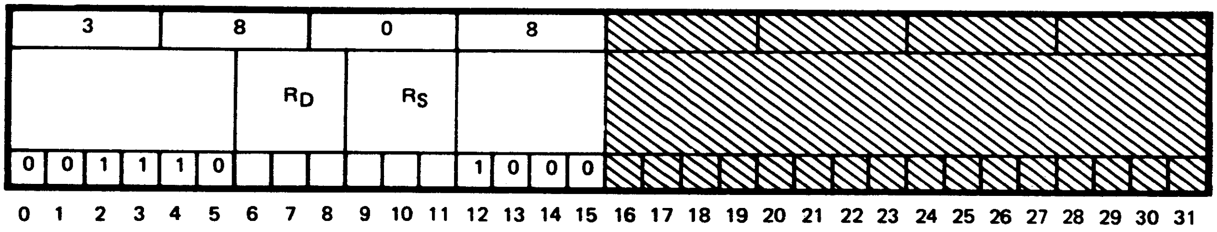
The contents of GPR6 and GPR7 are added and the result is transferred to GPR6. CC2 is set.

Memory Location:	03FA2
Hexadecimal Instruction:	3B70 (R_D=6, R_S=7)
Assembly Language Coding:	ADR 7,6

Before	PSD1 0A003FA2	GPR6 FF03C67D	GPR7 045C6E3F
After	PSD1 22003FA5	GPR6 036034BC	GPR7 045C6E3F

ADD REGISTER TO REGISTER MASKED 3808

ADRM s,d



830251

DEFINITION

The word in the general purpose register (GPR) specified by R_D is algebraically added to the word in the GPR specified by R_S. The result of this addition is masked (logical AND function) with the contents of the mask register (R4). The resulting word is then transferred to the GPR specified by R_D.

SUMMARY EXPRESSION

$(R_S)+(R_D)\ \&\ (R4) \rightarrow R_D$

CONDITION CODE RESULTS

CC1: Is set if arithmetic exception occurs
CC2: Is set if (R_D) is greater than zero
CC3: Is set if (R_D) is less than zero
CC4: Is set if (R_D) is equal to zero

NONBASE AND BASE REGISTER MODE EXAMPLE

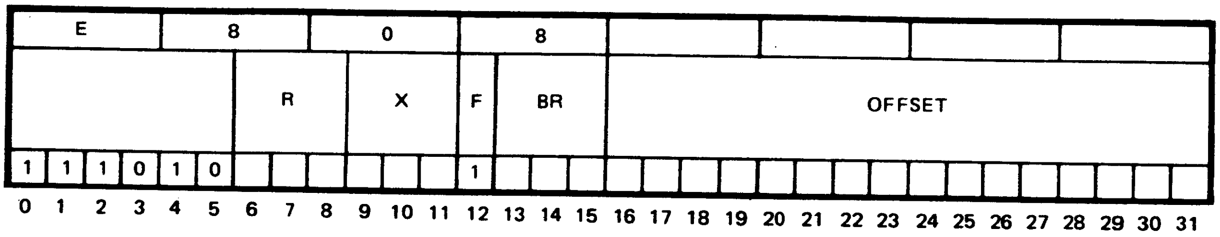
The contents of GPR6 and GPR7 are added; the result is ANDed with the contents of GPR4 and transferred to GPR6. CC2 is set.

Memory Location:	16A9A
Hexadecimal Instruction:	3B 78 (R_D=6, R_S=7)
Assembly Language Coding:	ADRM 7,6

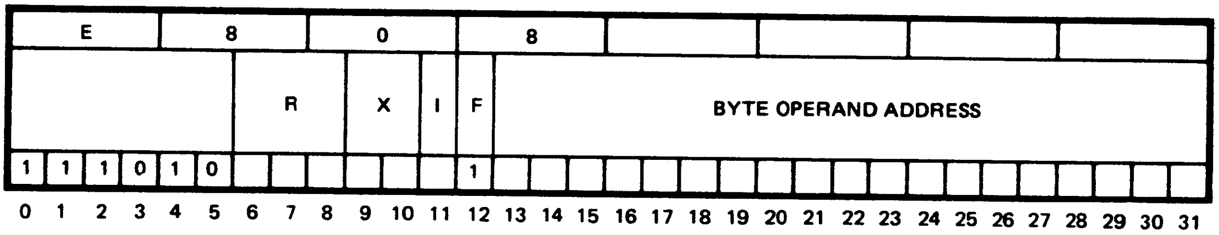
Before	PSD1	GPR4	GPR6	GPR7
	40016A9A (Nonbase)	007FFFFC	004FC276	0037C1F3
	42016A9A (Base)			
After	PSD1	GPR4	GPR6	GPR7
	20016A9D (Nonbase)	0007FFFC	00078468	0037C1F3
	22016A9D (Base)			

ADD REGISTER TO MEMORY BYTE
E808

ARMB
s,*m,x



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

830252

DEFINITION

The effective byte location (EBL) specified by the effective byte address (EBA) is accessed and algebraically added to the least-significant byte (bits 24-31) of the general purpose register (GPR) specified by R. The result is then transferred to the memory byte location specified by the EBA. The other three bytes in the word which contain the byte specified by the EBA remain unchanged.

SUMMARY EXPRESSION

$(R_{24-31})+(EBL) \rightarrow EBL$

(EBA_{0-23}) unchanged

CONDITION CODE RESULTS

CC1: Always zero
CC2: Always zero
CC3: Always zero
CC4: Is set if (EBL) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of bits 24-31 of GPR6 and memory byte 001A97 are added and the result is transferred to memory byte 001A97.

Memory Location:	01A64
Hexadecimal Instruction:	EB0E1A90 (R=6, X=0, BR=6)
Assembly Language Coding:	ARMB 6,X'1A90'(6)

Before	PSD1	GPR6	BR6	Memory Byte 001A97
	02001A64	0000004A	00000007	39
After	PSD1	GPR6	BR6	Memory Byte 001A97
	02001A68	0000004A	00000007	83

NONBASE REGISTER MODE EXAMPLE

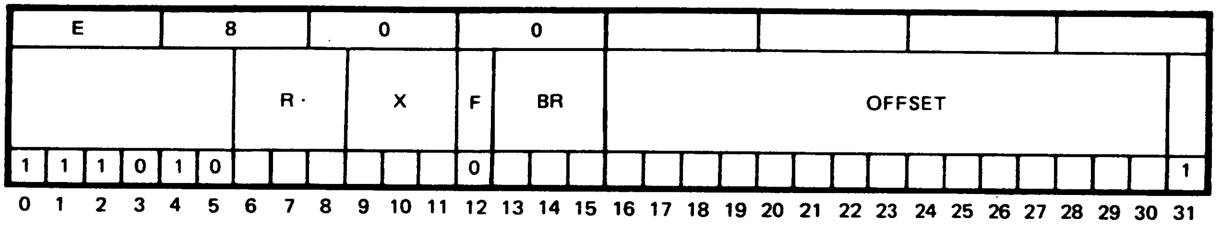
The contents of bits 24-31 of GPR6 and memory byte 01A97 are added and the result is transferred to memory byte 01A97.

Memory Location:	01A64
Hexadecimal Instruction:	EB 08 1A 97 (R=6, X=0, I=0)
Assembly Language Coding:	ARMB 6,X'1A97'

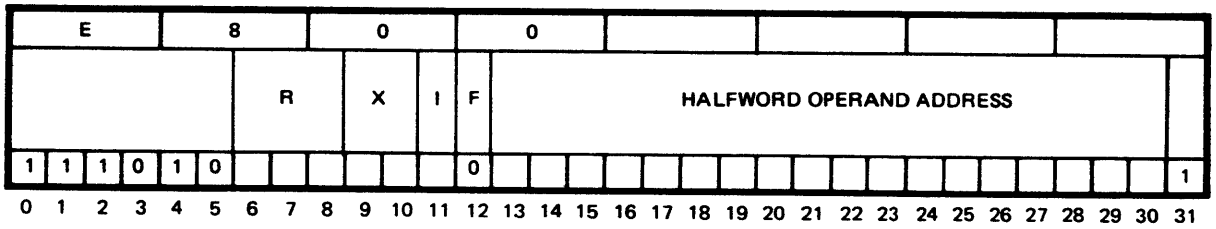
Before	PSD1	GPR6	Memory Byte 01A97
	00001A64	0000004A	39
After	PSD1	GPR6	Memory Byte 01A97
	00001A68	0000004A	83

ADD REGISTER TO MEMORY HALFWORD E800

ARMH
s,*m,x



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

830253

DEFINITION

The effective halfword location (EHL) specified by the effective halfword address (EHA) is accessed and algebraically added to the least-significant halfword (bits 16-31) of the general purpose register (GPR) specified by R. The result is then transferred to the memory halfword location specified by the EHA. The remaining halfword of the specified EHA remains unchanged.

SUMMARY EXPRESSION

$(R_{16-31})+(EHA) \rightarrow EHL$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Always zero
CC3: Always zero
CC4: Is set if EHL is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of bits 16-31 of GPR5 and memory halfword 001416 are added and the result is transferred to memory halfword 001416.

Memory Location:	200B4
Hexadecimal Instruction:	EA861017 (R=5, X=0, BR=6)
Assembly Language Coding:	ARMH 5,X'1016'(6)

Before	PSD1	GPR5	BR6	Memory Halfword 001416
	020200B4	FFFF8C42	00000400	06C4
After	PSD1	GPR5	BR6	Memory Halfword 001416
	020200B8	FFFF8C42	00000400	9306

NONBASE REGISTER MODE EXAMPLE

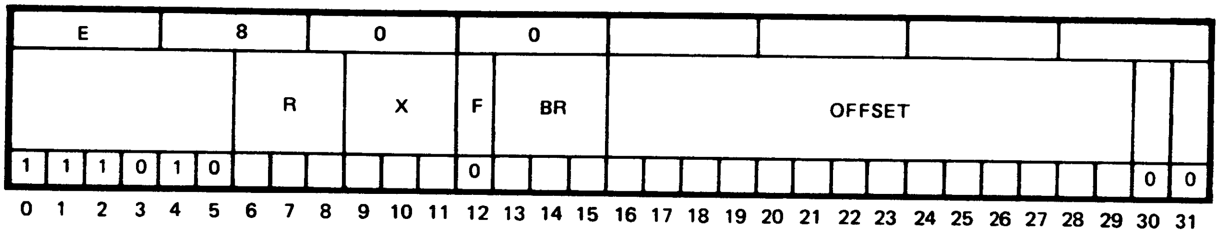
The contents of bits 16-31 of GPR5 and memory halfword 20918 are added and the result is transferred to memory halfword 20918.

Memory Location:	200B4
Hexadecimal Instruction:	EA 82 09 19 (R=5, X=0, I=0)
Assembly Language Coding:	ARMH 5,X'20918'

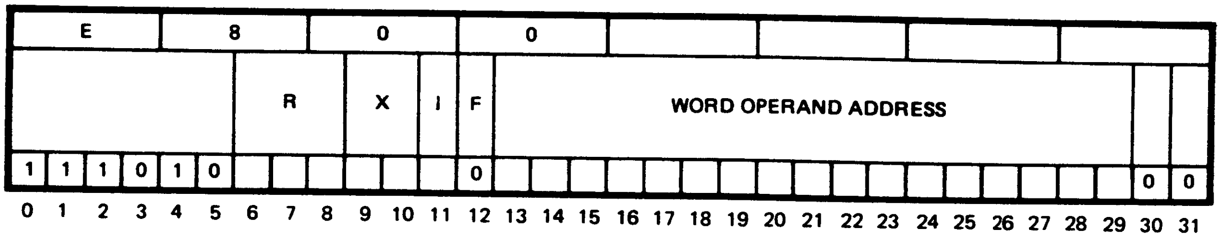
Before	PSD1	GPR5	Memory Halfword 20918
	000200B4	FFFF8C42	06C4
After	PSD1	GPR5	Memory Halfword 20918
	000200B8	FFFF8C42	9306

ADD REGISTER TO MEMORY WORD E800

ARMW s,*m,x



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

830254

DEFINITION

The contents of the effective word location (EWL) specified by the effective word address (EWA) is accessed and algebraically added to the word in the general purpose register (GPR) specified by R. The resulting word is then transferred to the memory word location specified by the EWA.

SUMMARY EXPRESSION

(R)+(EWL) → EWL

CONDITION CODE RESULTS

CC1: Is set if arithmetic exception occurs
CC2: Is set if (EWL) is greater than zero
CC3: Is set if (EWL) is less than zero
CC4: Is set if (EWL) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of GPR7 and the memory word 0031A0 are added and the result is transferred to memory word 0031A0. CC2 is set.

Memory Location:	03000
Hexadecimal Instruction:	EB863100 (R=7, X=0, BR=6)
Assembly Language Coding:	ARMW 7,X'3100'(6)

Before	PSD1	GPR7	BR6	Memory Word 0031A0
	0A003000	245C6E3F	000000A0	FF03C67D
After	PSD1	GPR7	BR6	Memory Word 0031A0
	22003004	245C6E3F	000000A0	236034BC

NONBASE REGISTER MODE EXAMPLE

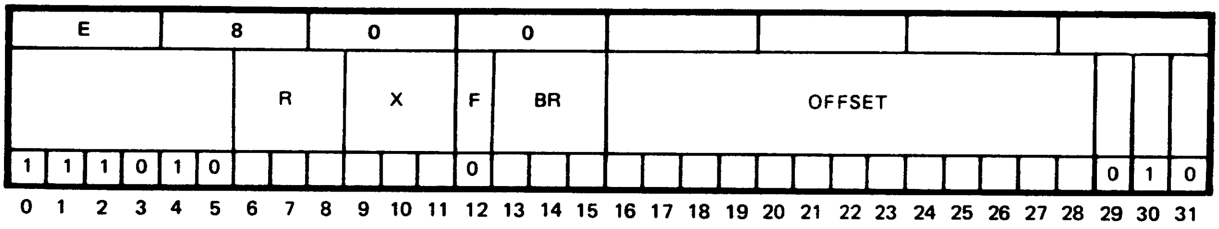
The contents of GPR7 and memory word 03100 are added and the result is transferred to memory word 03100. CC2 is set.

Memory Location:	03000
Hexadecimal Instruction:	EB 80 31 00 (R=7, X=0, I=0)
Assembly Language Coding:	ARMW 7,X'3100'

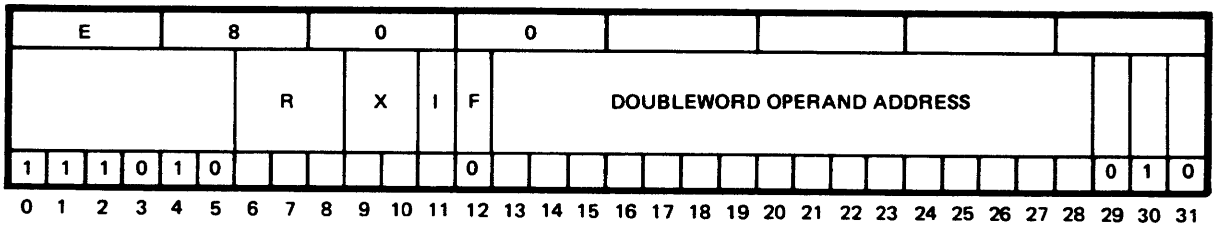
Before	PSD1	GPR7	Memory Word 03100
	08003000	245C6E3F	FF03C67D
After	PSD1	GPR7	Memory Word 03100
	20003004	245C6E3F	236034BC

ADD REGISTER TO MEMORY DOUBLEWORD E800

ARMD
s,*m,x



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

DEFINITION

The contents of the effective word locations (EWL, EWL+1) specified by the effective doubleword address (EDA) are accessed and algebraically added to the doubleword in the general purpose register (GPR) specified by R and R+1. R+1 is the GPR one greater than specified by R. The contents of the GPR specified by R+1 are added to the contents of the least-signifcant word of the doubleword first. The resulting doubleword is transferred to the memory doubleword location specified by the EDA.

NOTE

The GPR specified by R must be an even-numbered register.

SUMMARY EXPRESSION

(R+1)+(EWL+1) → EWL+1+Carry
(R)+(EWL)+Carry → EWL

CONDITION CODE RESULTS

CC1: Is set if arithmetic exception occurs
CC2: Is set if (EDL) is greater than zero
CC3: Is set if (EDL) is less than zero
CC4: Is set if (EDL) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The doubleword obtained from GPR6 and GPR7 is added to the doubleword from memory words 0083A8 and 0083AC. The result is transferred to memory words 0083A8 and 0083AC. CC2 is set.

Memory Location:	0819C
Hexadecimal Instruction:	EB06800A (R=6, X=0, BR=6)
Assembly Language Coding:	ARMD 6,X'8008'(6)

Before	PSD1 4200819C	GPR6 01A298A1	GPR7 F15BC63E	BR6 000003A0
	Memory Word 0083A8 3B69A07E		Memory Word 0083AC 7F3579A4	
After	PSD1 220081A0	GPR6 01A298A1	GPR7 F15BC63E	BR6 000003A0
	Memory Word 0083A8 3D0C3920		Memory Word 0083AC 70913FE2	

NONBASE REGISTER MODE EXAMPLE

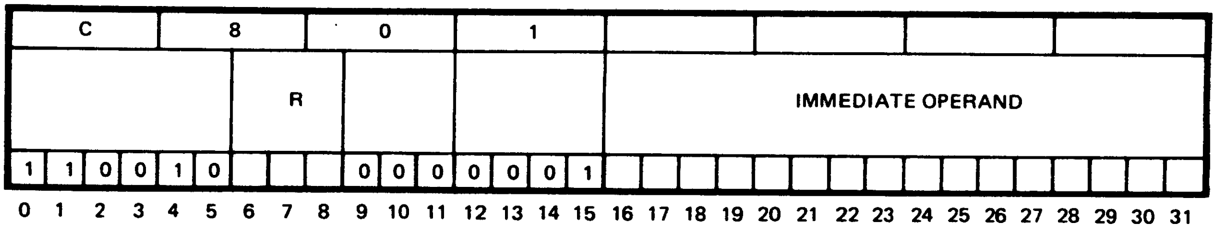
The doubleword obtained from GPR6 and GPR7 is added to the doubleword from memory words 083A8 and 083AC. The result is transferred to memory words 083A8 and 083AC. CC2 is set.

Memory Location:	0819C
Hexadecimal Instruction:	EB 00 83 AA (R=6, X=0, I=0)
Assembly Language Coding:	ARMD 6,X'83A8'

Before	PSD1 4000819C	GPR6 01A298A1	GPR7 F15BC63E
	Memory Word 083A8 3B69A07E		Memory Word 083AC 7F3579A4
After	PSD1 200081A0	GPR6 01A298A1	GPR7 F15BC63E
	Memory Word 083A8 3D0C3920		Memory Word 083AC 70913FE2

ADD IMMEDIATE C801

ADI
d, v



830256

DEFINITION

The sign of the least-significant halfword (bits 16-31) of the instruction word (IW) is extended 16 bits to the left to form a word. This word is algebraically added to the word in the general purpose register (GPR) specified by R. The resulting word is transferred to the GPR specified by R.

SUMMARY EXPRESSION

$(IW_{16-31})_{SE}+(R) \rightarrow R$

CONDITION CODE RESULTS

CC1: Is set if arithmetic exception occurs
CC2: Is set if R_{0-31} is greater than zero
CC3: Is set if R_{0-31} is less than zero
CC4: Is set if R_{0-31} is equal to zero

NONBASE AND BASE REGISTER MODE EXAMPLE

The immediate operand, sign extended, is added to the contents of the GPR0 and the result replaces the previous contents of GPR0. CC4 is set.

Memory Location:	00D88
Hexadecimal Instruction:	C8 01 86 B2 (R=0)
Assembly Language Coding:	ADI 0,X'86B2'

Before	PSD1	GPR0
	20000D88 (Nonbase)	0000794E
	22000D88 (Base)	
After	PSD1	GPR0
	08000D8C (Nonbase)	00000000
	0A000D8C (Base)	

SUBTRACT MEMORY BYTE
BC08

SUMB
d,*m,x

B				C				0				8																			
						R			X			F	BR			OFFSET															
1	0	1	1	1	1							1																			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

BASE REGISTER FORMAT

B				C				0				8																			
						R			X		I	F	BYTE OPERAND ADDRESS																		
1	0	1	1	1	1							1																			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

NONBASE REGISTER FORMAT

830257

DEFINITION

The contents of the effective byte location (EBL) specified by the effective byte address (EBA) is accessed and 24 zeros are appended to the most-significant end to form a word. This word is algebraically subtracted from the word in the general purpose register (GPR) specified by R. The resulting word is transferred to the GPR specified by R.

SUMMARY EXPRESSION

$(R)- \text{Zeros}_{0-23},(\text{EBL}) \rightarrow R$

CONDITION CODE RESULTS

CC1: Is set if arithmetic exception occurs
CC2: Is set if (R_{0-31}) is greater than zero
CC3: Is set if (R_{0-31}) is less than zero
CC4: Is set if (R_{0-31}) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of memory byte 001203, with 24 zeros prefixed, are subtracted from the contents of GPR1; the result is transferred to GPR1. CC2 is set.

Memory Location:	01000
Hexadecimal Instruction:	BC8E1200 (R=1, X=0, BR=6)
Assembly Language Coding:	SUMB 1,X'1200'(6)

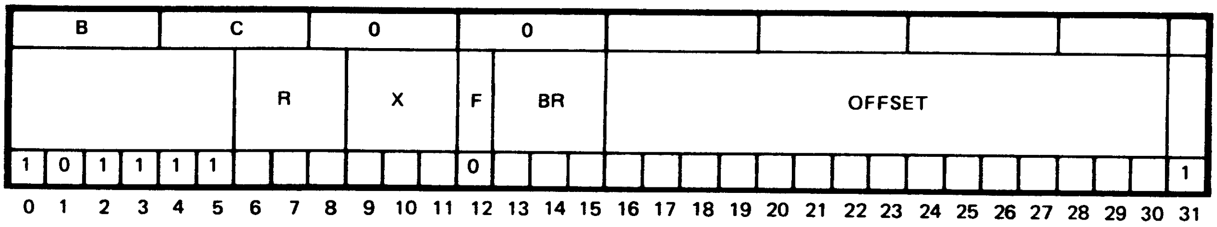
Before	PSD1	GPR1	BR6	Memory Byte 001203
	42001000	0194A7F2	00000003	9A
After	PSD1	GPR1	BR6	Memory Byte 001203
	22001004	0194A758	00000003	9A

NONBASE REGISTER MODE EXAMPLE

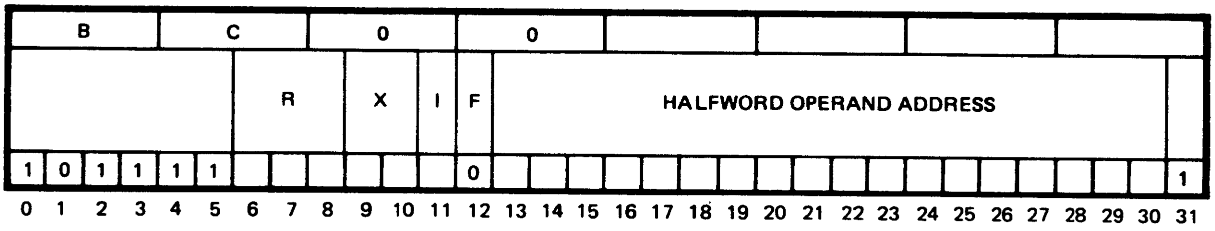
The contents of memory byte 01203, with 24 zeros prefixed, are subtracted from the contents of GPR1; the result is transferred to GPR1. CC2 is set.

Memory Location:	01000
Hexadecimal Instruction:	BC 88 12 03 (R=1, X=0, I=0)
Assembly Language Coding:	SUMB 1,X'1203'

Before	PSD1	GPR1	Memory Byte 01203
	40001000	0194A7F2	9A
After	PSD1	GPR1	Memory Byte 01203
	20001004	0194A758	9A



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

830258

DEFINITION

The contents of the effective halfword location (EHL) specified by the effective halfword address is accessed and the sign bit (bit 16) is extended 16 bits to the left to form a word. This word is algebraically subtracted from the word in the general purpose register (GPR) specified by R. The resulting word is then transferred to the GPR specified by R.

SUMMARY EXPRESSION

$(R)-(EHL)_{SE} \rightarrow R$

CONDITION CODE RESULTS

CC1: Is set if arithmetic exception occurs
CC2: Is set if (R_{0-31}) is greater than zero
CC3: Is set if (R_{0-31}) is less than zero
CC4: Is set if (R_{0-31}) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of memory halfword 001876, sign extended, are subtracted from the contents of GPR6. The result is transferred to GPR6. CC2 is set.

Memory Location:	01604
Hexadecimal Instruction:	BF061007 (R=6, X=0, BR=6)
Assembly Language Coding:	SUMH 6,X'1006'(6)

Before	PSD1	GPR6	BR6	Memory Halfword 001876
	12001604	00024CB3	00000870	34C6
After	PSD1	GPR6	BR6	Memory Halfword 001876
	22001608	000217ED	00000870	34C6

NONBASE REGISTER MODE EXAMPLE

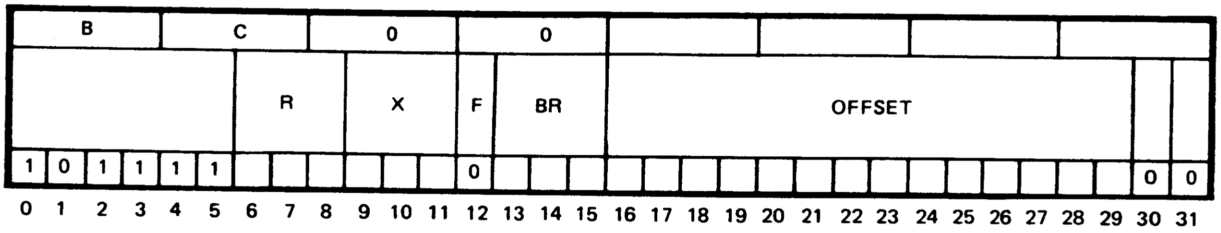
The contents of memory halfword 01876, sign extended, are subtracted from the contents of GPR6; the result is transferred to GPR6. CC2 is set.

Memory Location:	01604
Hexadecimal Instruction:	BF 00 18 77 (R=6, X=0, I=0)
Assembly Language Coding:	SUMH 6,X'1876'

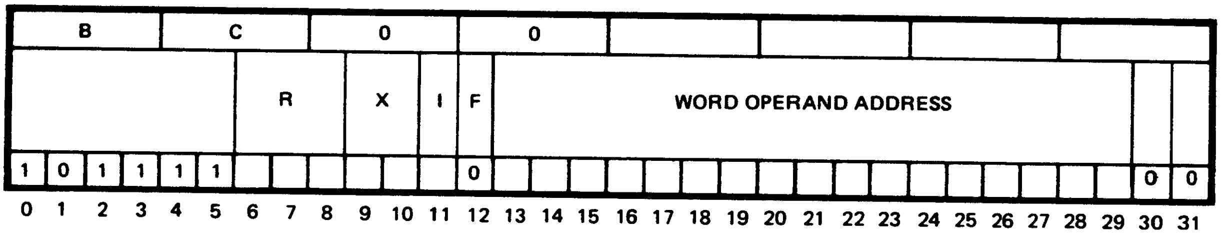
Before	PSD1	GPR6	Memory Halfword 01876
	10001604	00024CB3	34C6
After	PSD1	GPR6	Memory Halfword 01876
	20001608	000217ED	34C6

SUBTRACT MEMORY WORD BC00

SUMW d,*m,x



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

830259

DEFINITION

The contents of the effective word location (EWL) specified by the effective word address (EWA) is accessed and algebraically subtracted from the word in the general purpose register (GPR) specified by R. The resulting word is then transferred to the GPR specified by R.

SUMMARY EXPRESSION

(R)-(EWL) → R

CONDITION CODE RESULTS

CC1: Is set if arithmetic exception occurs
CC2: Is set if (R_{0-31}) is greater than zero
CC3: Is set if (R_{0-31}) is less than zero
CC4: Is set if (R_{0-31}) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6, GPR4 and the instruction offset are added to obtain the logical address. The contents of memory word 00F914 are subtracted from the contents of GPR1 and the result is transferred to GPR1. CC2 is set.

Memory Location:	6C208
Hexadecimal Instruction:	BCC6F900 (R=1, BR=6, X=4)
Assembly Language Coding:	SUMW 1, X 'F900' (6), 4

Before	PSD1	GPR1	BR6	GPR4	Memory Word 00F914
	0206C208	00A6264D	00000010	00000004	00074BC3
After	PSD1	GPR1	BR6	GPR4	Memory Word 00F914
	2206C20C	009EDA8A	00000010	00000004	00074BC3

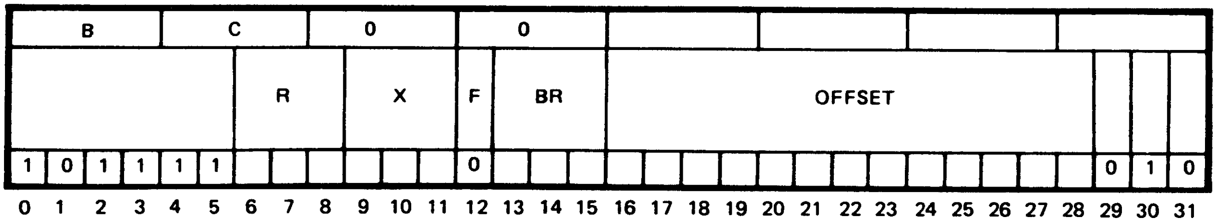
NONBASE REGISTER MODE EXAMPLE

The contents of memory word 6F914 are subtracted from the contents of GPR1 and the result is transferred to GPR1. CC2 is set.

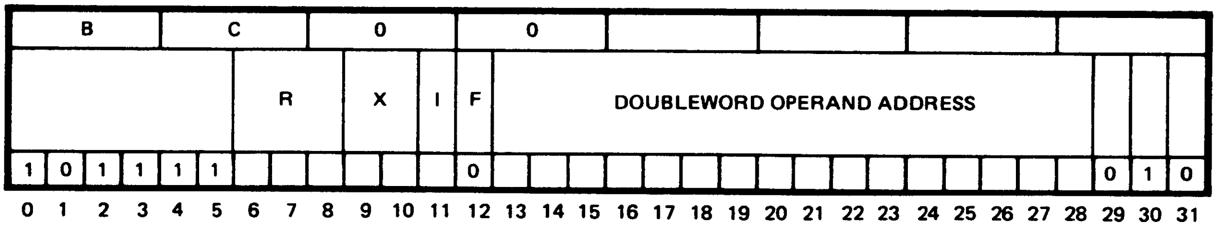
Memory Location:	6C208
Hexadecimal Instruction:	BC 86 F9 14 (R=1, X=0, I=0)
Assembly Language Coding:	SUMW 1,X'6F914'

Before	PSD1	GPR1	Memory Word 6F914
	0406C208	00A6264D	00074BC3
After	PSD1	GPR1	Memory Word 6F914
	2006C20C	009EDA8A	00074BC3

SUBTRACT MEMORY DOUBLEWORD SUMD
BC00 d,*m,x



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

DEFINITION

The contents of the effective word locations (EWL, EWL+1) specified by the effective doubleword address (EDA) is accessed and algebraically subtracted from the doubleword in the general purpose register (GPR) specified by R and R+1. R+1 is the GPR one greater than specified by R. The word located in the GPR specified by R+1 is subtracted from the least-significant word of the doubleword first. The resulting doubleword is transferred to the GPR specified by R and R+1.

NOTE

The GPR specified by R must be an even-numbered register.

SUMMARY EXPRESSION

(R+1)-(EWL+1) → R+1-Borrow

(R)-(EWL)-Borrow → R

CONDITION CODE RESULTS

CC1: Is set if arithmetic exception occurs
CC2: Is set if (R, R+1) is greater than zero
CC3: Is set if (R, R+1) is less than zero
CC4: Is set if (R, R+1) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The doubleword obtained from memory words 003100 and 003104 is subtracted from the doubleword in GPR6 and GPR7; the result is transferred to GPR6 and GPR7. CC2 is set.

Memory Location:	03000
Hexadecimal Instruction:	BF063002 (R=6, X=0, BR=6)
Assembly Language Coding:	SUMD 6,X'3000'(6)

Before	PSD1	GPR6	GPR7	BR6
	12003000	5AD983B7	C833D509	00000100
	Memory Word 003100		Memory Word 003104	
	153B0492		5BE87A16	
After	PSD1	GPR6	GPR7	BR6
	22003004	459E7F25	6C4B5AF3	00000100
	Memory Word 003100		Memory Word 003104	
	153B0492		5BE87A16	

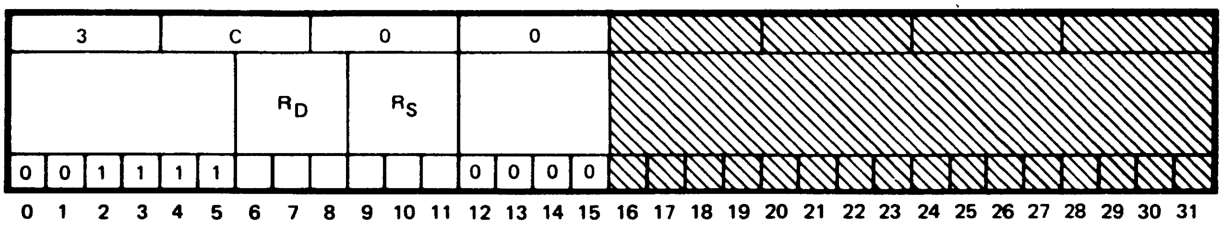
NONBASE REGISTER MODE EXAMPLE

The doubleword obtained from memory words 03100 and 03104 is subtracted from the doubleword in GPR6 and GPR7; the result is transferred to GPR6 and GPR7. CC2 is set.

Memory Location:	03000
Hexadecimal Instruction:	BF 00 31 02 (R=6, X=0, I=0)
Assembly Language Coding:	SUMD 6,X'3100'

Before	PSD1	GPR6	GPR7
	10003000	5AD983B7	C833D509
	Memory Word 03100		Memory Word 03104
	153B0492		5BE87A16
After	PSD1	GPR6	GPR7
	20003004	459E7F25	6C4B5AF3
	Memory Word 03100		Memory Word 03104
	153B0492		5BE87A16

SUBTRACT REGISTER FROM REGISTER 3C00 — SUR s,d



830261

DEFINITION

The word in the general purpose register (GPR) specified by R_S is algebraically subtracted from the word in the GPR specified by R_D. The resulting word is then transferred to the GPR specified by R_D.

SUMMARY EXPRESSION

$(R_D)-(R_S) \rightarrow R_D$

CONDITION CODE RESULTS

CC1: Is set if arithmetic exception occurs
CC2: Is set if (R_D) is greater than zero
CC3: Is set if (R_D) is less than zero
CC4: Is set if (R_D) is equal to zero

NONBASE AND BASE REGISTER MODE EXAMPLE

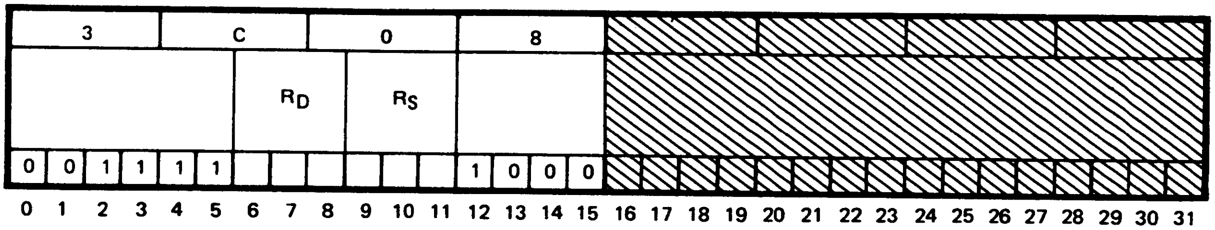
The contents of GPR2 are subtracted from the contents of GPR1. The result is transferred to GPR1. CC4 is set.

Memory Location:	106AE
Hexadecimal Instruction:	3C A0 (R_D=1, R_S=2)
Assembly Language Coding:	SUR 2,1

Before	PSD1	GPR1	GPR2
	100106AE (Nonbase)	12345678	12345678
	120106AE (Base)		
After	PSD1	GPR1	GPR2
	080106B1 (Nonbase)	00000000	12345678
	0A0106B1 (Base)		

SUBTRACT REGISTER FROM REGISTER MASKED
3C08

SURM
s,d



DEFINITION

The word in the general purpose register (GPR) specified by R_S is algebraically subtracted from the word in the GPR specified by R_D. The result of this subtraction is then masked (logical AND function) with the contents of the mask register (R4). The resulting word is transferred to the GPR specified by R_D.

SUMMARY EXPRESSION

$(R_D)-(R_S)\ \&(R4) \rightarrow R_D$

CONDITION CODE RESULTS

CC1: Is set if arithmetic exception occurs
CC2: Is set if (R_D) is greater than zero
CC3: Is set if (R_D) is less than zero
CC4: Is set if (R_D) is equal to zero

NONBASE AND BASE REGISTER MODE EXAMPLE

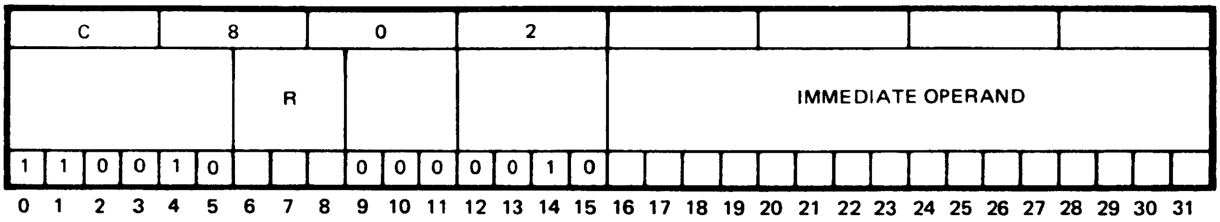
The contents of GPR5 are subtracted from the contents of GPR6. The result is ANDed with the contents of GPR4 and transferred to GPR6. CC2 is set.

Memory Location:	00496
Hexadecimal Instruction:	3F 58 (R_D=6, R_S=5)
Assembly Language Coding:	SURM 5,6

Before	PSD1	GPR4	GPR5	GPR6
	10000496 (Nonbase)	00FFFF00	00074BC3	00A6264D
	12000496 (Base)			
After	PSD1	GPR4	GPR5	GPR6
	20000499 (Nonbase)	00FFFF00	00074BC3	009EDA00
	22000499 (Base)			

SUBTRACT IMMEDIATE C802

SUI d,v



830263

DEFINITION

The sign of the least-significant halfword (bits 16-31) of the instruction word is extended 16 bits to the left to form a word. This word is algebraically subtracted from the word in the general purpose register (GPR) specified by R. The resulting word is transferred to the GPR specified by R.

SUMMARY EXPRESSION

$(R)-(IW_{16-31})_{SE} \rightarrow R$

CONDITION CODE RESULTS

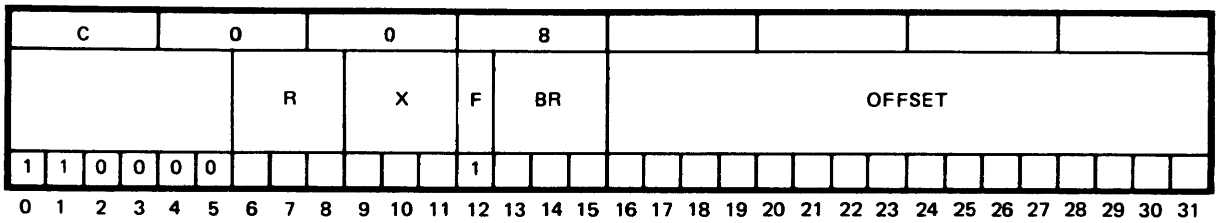
CC1: Is set if arithmetic exception occurs
CC2: Is set if R_{0-31} is greater than zero
CC3: Is set if R_{0-31} is less than zero
CC4: Is set if R_{0-31} is equal to zero

NONBASE AND BASE REGISTER MODE EXAMPLE

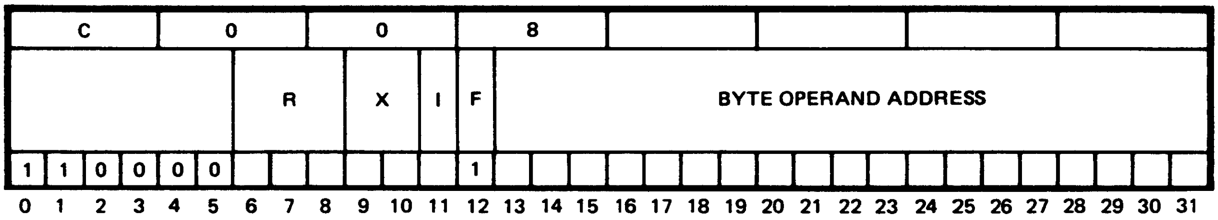
The immediate operand with sign extension is subtracted from the contents of GPR7. The result is transferred to GPR7. CC4 is set.

Memory Location:	019B8
Hexadecimal Instruction:	CB 82 83 9A (R=7)
Assembly Language Coding:	SUI 7,'X'839A'

Before	PSD1	GPR7
	100019B8 (Nonbase)	FFFF839A
	120019B8 (Base)	
After	PSD1	GPR7
	080019BD (Nonbase)	00000000
	0A0019BD (Base)	



BASE REGISTER FORMAT



830264

NONBASE REGISTER FORMAT

DEFINITION

The contents of the effective byte location (EBL) specified by the effective byte address (EBA) is accessed and 24 zeros are appended to the most-significant end to form a word. This word is algebraically multiplied by the word in the general purpose register (GPR) specified by R+1. R+1 is the GPR one greater than specified by R. The doubleword result is transferred to the GPR specified by R and R+1.

NOTES

1. An arithmetic exception will never occur since the result of a multiplication can never exceed the length of the doubleword register.

2. GPR specified by R must have an even address.

3. The previous content of R is not used and is destroyed.

4. Operation will be performed in the FPA if present and enabled.

SUMMARY EXPRESSION

Zeros_{0-23}, (EBL)x(R+1) → R,R+1

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R, R+1) is greater than zero
CC3: Is set if (R, R+1) is less than zero
CC4: Is set if (R, R+1) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of memory byte 00C3D3, with zeros prefixed, are multiplied by the contents of GPR1; the result is transferred to GPR0 and GPR1. CC2 is set.

Memory Location:	2BA28
Hexadecimal Instruction:	C00EC300 (R=0, X=0, BR=6)
Assembly Language Coding:	MPMB 0,X'C300'(6)

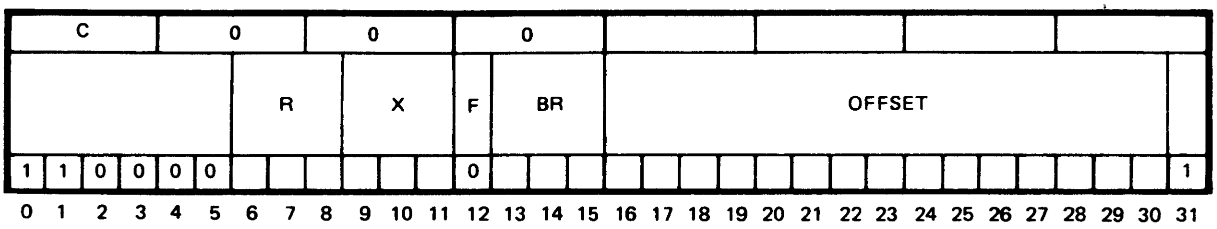
Before	PSD1	GPR0	GPR1	BR6	Memory Byte 00C3D3
	0202BA28	12345678	6F90C859	000000D3	40
After	PSD1	GPR0	GPR1	BR6	Memory Byte 00C3D3
	2202BA2C	0000001B	E4321640	000000D3	40

NONBASE REGISTER MODE EXAMPLE

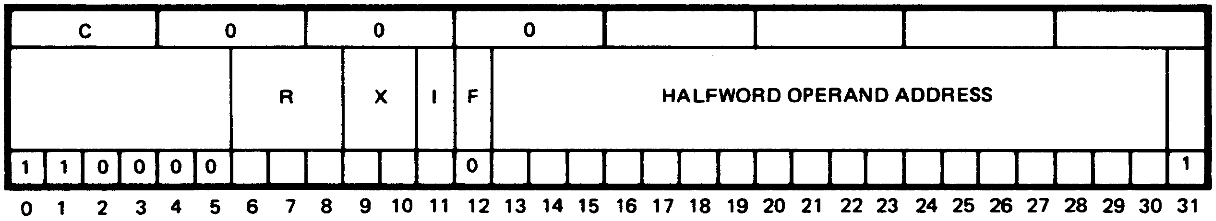
The contents of memory byte 2C3D3, with zeros prefixed, are multiplied by the contents of GPR1; the result is transferred to GPR0 and GPR1. CC2 is set.

Memory Location:	2BA28
Hexadecimal Instruction:	C0 0A C3 D3 (R=0, X=0, I=0)
Assembly Language Coding:	MPMB 0,X'2C3D3'

Before	PSD1	GPR0	GPR1	Memory Byte 2C3D3
	0002BA28	12345678	6F90C859	40
After	PSD1	GPR0	GPR1	Memory Byte 2C3D3
	2002BA2C	0000001B	E4321640	40



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

830265

DEFINITION

The contents of the effective halfword location (EHL) specified by the effective halfword address (EHA) is accessed and the sign bit (bit 16) is extended 16 bits to the left to form a word. This word is algebraically multiplied by the word in the general purpose register (GPR) specified by R+1. R+1 is the GPR one greater than specified by R. The doubleword result is transferred to the GPR specified by R and R+1.

NOTES

1. An arithmetic exception will never occur since the result of a multiplication can never exceed the length of the doubleword register.

2. GPR specified by R must have an even address.

3. The previous content of R is not used and is destroyed.

4. Operation will be performed in the FPA if present and enabled.

SUMMARY EXPRESSION

$(EHL)_{SE} x (R+1) \rightarrow R,R+1$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R, R+1) is greater than zero
CC3: Is set if (R, R+1) is less than zero
CC4: Is set if (R, R+1) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The contents of GPR3 are multiplied by the contents of memory halfword 009B56. The doubleword result is transferred to GPR2 and GPR3. CC3 is set.

Memory Location:	096A4
Hexadecimal Instruction:	C1069B07 (R=2, X=0, BR=6)
Assembly Language Coding:	MPMH 2,X'9B06'(6)

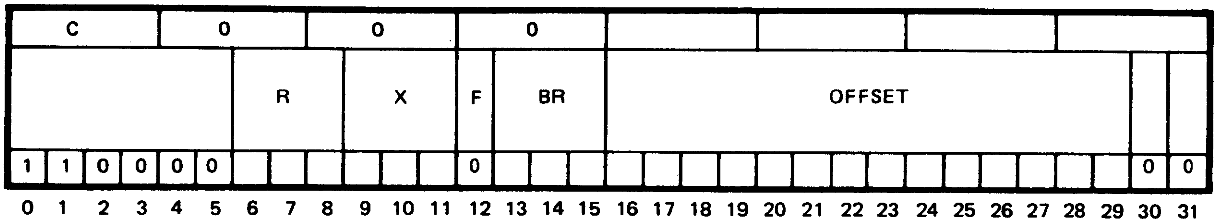
Before	PSD1	GPR2	GPR3	BR6
	0A0096A4	12345678	00000003	00000050
	Memory Halfword 009B56			
	FFFD			
After	PSD1	GPR2	GPR3	BR6
	120096A8	FFFFFFFF	FFFFFFF7	00000050
	Memory Halfword 009B56			
	FFFD			

NONBASE REGISTER MODE EXAMPLE

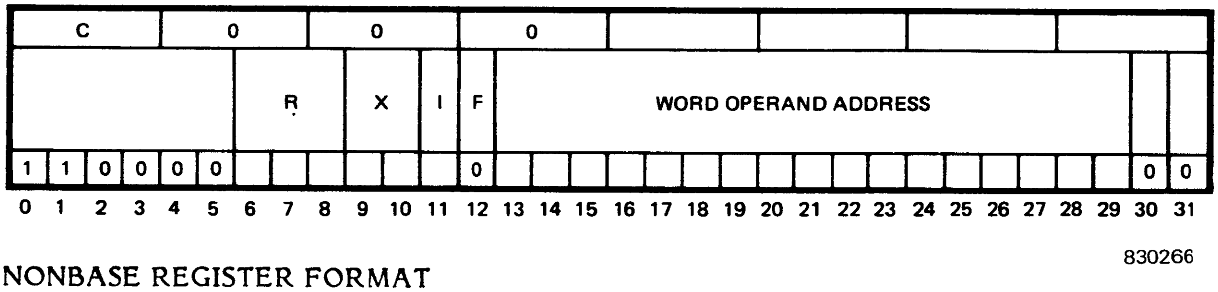
The contents of GPR3 are multiplied by the contents of memory halfword 09B56; the doubleword result is transferred to GPR2 and GPR3. CC3 is set.

Memory Location:	096A4
Hexadecimal Instruction:	C1 00 9B 57 (R=2, X=0, I=0)
Assembly Language Coding:	MPMH 2,X'9B56'

Before	PSD1	GPR2	GPR3	Memory Halfword 09B56
	080096A4	12345678	00000003	FFFD
After	PSD1	GPR2	GPR3	Memory Halfword 09B56
	100096A8	FFFFFFFF	FFFFFFF7	FFFD



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

DEFINITION

The contents of the effective word location (EWL) specified by the effective word address (EWA) is accessed and algebraically multiplied by the word in the general purpose register (GPR) specified by R+1. R+1 is the GPR one greater than specified by R. The doubleword result is transferred to the GPR specified by R and R+1.

NOTES

1. An arithmetic exception will never occur since the result of a multiplication can never exceed the length of the doubleword register.

2. GPR specified by R must have an even address.

3. The previous content of R is not used and is destroyed.

4. Operation will be performed in the FPA if present and enabled.

SUMMARY EXPRESSION

(EWL)x(R+1) → (R,R+1)

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R, R+1) is greater than zero
CC3: Is set if (R, R+1) is less than zero
CC4: Is set if (R, R+1) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of GPR4, BR6 and the instruction offset are added to obtain the logical address. The contents of GPR7 and memory word 004B1C are multiplied; the result is transferred to GPR6 and GPR7. CC2 is set.

Memory Location:	04AC8
Hexadecimal Instruction:	C3464B00 (R=6, BR=6, X=4)
Assembly Language Coding:	MPMW 6, X '4B00' (B6), R4

Before	PSD1	GPR6	GPR7	BR6	GPR4
	12004AC8	00000000	80000000	00000010	0000000C
	Memory Word 004B1C				
	80000000				
After	PSD1	GPR6	GPR7	BR6	GPR4
	22004ACC	40000000	00000000	00000010	0000000C
	Memory Word 004B1C				
	80000000				

NONBASE REGISTER MODE EXAMPLE

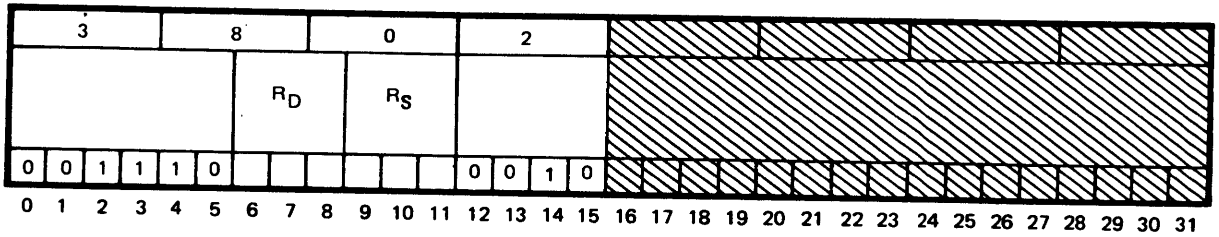
The contents of GPR7 and memory word 04B1C are multiplied; the result is transferred to GPR6 and GPR7. CC2 is set.

Memory Location:	04AC8
Hexadecimal Instruction:	C3 00 4B 1C (R=6, X=0, I=0)
Assembly Language Coding:	MPMW 6,X'4B1C'

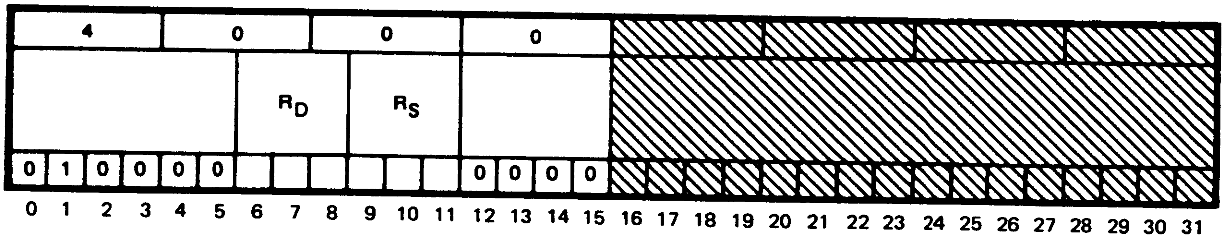
Before	PSD1	GPR6	GPR7	Memory Word 04B1C
	10004AC8	00000000	80000000	80000000
After	PSD1	GPR6	GPR7	Memory Word 04B1C
	20004ACC	40000000	00000000	80000000

MULTIPLY REGISTER BY REGISTER

MPR
s,d



BASE REGISTER FORMAT (3802)



NONBASE REGISTER FORMAT (4000)

830267

DEFINITION

The word in the general purpose register (GPR) specified by R_S is algebraically multiplied by the word in the GPR specified by R_D+1. R_D+1 is the GPR one greater than specified by R_D. The doubleword result is transferred to the GPR specified by R_D and R_D+1.

NOTES

1. The multiplicand register R_S can be any register, including either register R_D or register R_D+1; however, R_D must be an even-numbered register.

2. An arithmetic exception will never occur since the result of a multiplication can never exceed the length of the doubleword register.

3. The previous content of R_D is destroyed.

4. Operation will be performed in the FPA if present and enabled.

SUMMARY EXPRESSION

$(R_S)x(R_D+1) \rightarrow R_D,R_D+1$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R_D,R_D+1) is greater than zero
CC3: Is set if (R_D,R_D+1) is less than zero
CC4: Is set if (R_D,R_D+1) is equal to zero

BASE REGISTER MODE EXAMPLE

The word of GPR1 is multiplied by itself; the doubleword product is transferred to GPR0 and GPR1. CC2 is set.

Memory Location:	0098E
Hexadecimal Instruction:	3812 (R_D=0, R_S=1)
Assembly Language Coding:	MPR 1,0

	PSD1	GPR0	GPR1
Before	1200098E	00000000	0000000F
After	22000991	00000000	000000E1

NONBASE REGISTER MODE EXAMPLE

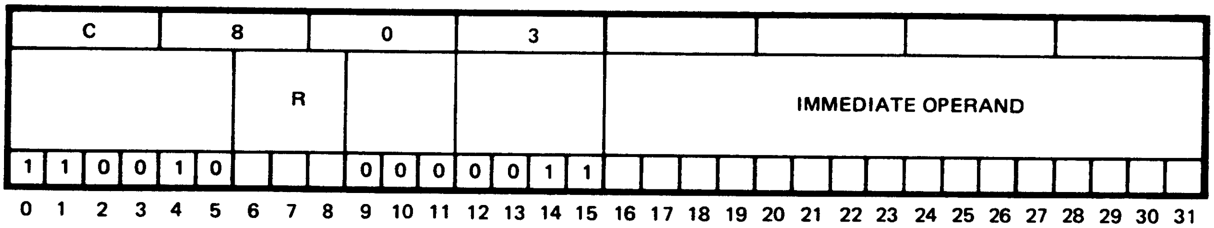
The word of GPR1 is multiplied by itself; the doubleword product is transferred to GPR0 and GPR1. CC2 is set.

Memory Location:	0098E
Hexadecimal Instruction:	40 10 (R_D=0, R_S=1)
Assembly Language Coding:	MPR 1,0

	PSD1	GPR0	GPR1
Before	1000098E	00000000	0000000F
After	20000991	00000000	000000E1

MULTIPLY IMMEDIATE C803

MPI
d,v



830268

DEFINITION

The sign of the least-significant halfword (bits 16-31) of the instruction word (IW) is extended 16 bits to the left to form a word. This word is algebraically multiplied by the word in the general purpose register (GPR) specified by R+1. R+1 is the GPR one greater than specified by R. The result is transferred to the GPR specified by R and R+1.

NOTES

1. An arithmetic exception will never occur since the result of a multiplication can never exceed the length of the doubleword register.
2. The GPR specified by R must have an even address.
3. The previous content of R is not used and is destroyed.
4. Operation will be performed in the FPA if present and enabled.

SUMMARY EXPRESSION

$(IW_{16-31})_{SE} x (R+1) \rightarrow R,R+1$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R, R+1) is greater than zero
CC3: Is set if (R, R+1) is less than zero
CC4: Is set if (R, R+1) is equal to zero

NONBASE AND BASE REGISTER MODE EXAMPLE

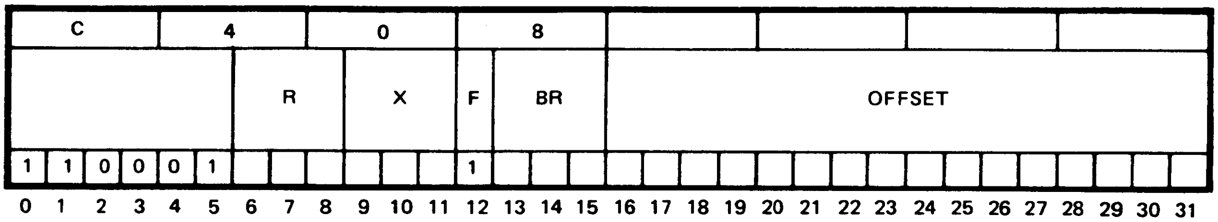
The immediate operand, sign extended, is multiplied by the contents of GPR7; the result is transferred to GPR6 and GPR7. CC3 is set.

Memory Location:	00634
Hexadecimal Instruction:	CB 03 01 00 (R=6)
Assembly Language Coding:	MPI 6,X'0100'

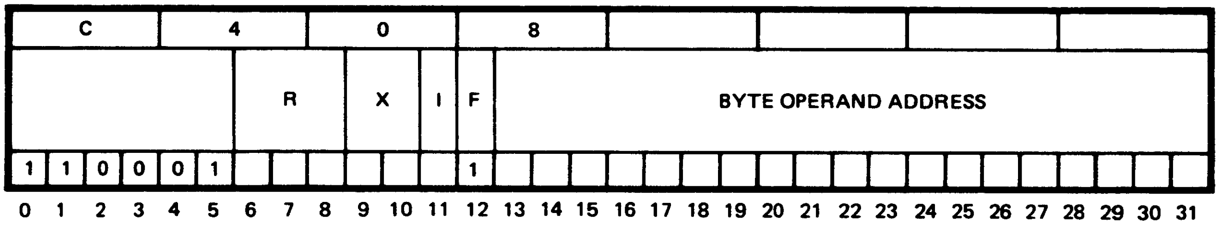
Before	PSD1	GPR6	GPR7
	20000634 (Nonbase)	12345678	F37A9B15
	22000634 (Base)		
After	PSD1	GPR6	GPR7
	10000638 (Nonbase)	FFFFFFF3	7A9B1500
	12000638 (Base)		

DIVIDE BY MEMORY BYTE C408

DVMB
d,*m,x



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

DEFINITION

The contents of the effective byte location (EBL) specified by the effective byte address (EBA) is accessed and 24 zeros are prefixed to form a word. This word is algebraically divided into the doubleword in the general purpose register (GPR) specified by R and R+1. R+1 is the GPR one greater than specified by R. The resulting quotient is then transferred to the GPR specified by R+1, and the remainder is transferred to the GPR specified by R. The sign of the GPR specified by R (remainder) is the same as to the sign of the dividend. The sign of the GPR specified by R+1 (quotient) will be the algebraic product of the original of the dividend and the divisor except when the absolute value of the dividend is less than the absolute value of the divisor. In the latter case, the resulting quotient (GPR specified by R+1) will be set to zero.

NOTES

1. An arithmetic exception occurs if the divisor is equal to zero, or the value of the quotient exceeds 31 bits. If an arithmetic exception occurs, the unchanged dividend will be retained in the GPR specified by R and R+1.

2. The GPR specified by R must have an even address.

SUMMARY EXPRESSION

$(R, R+1)/\ Zeros_{0-23},(EBL) \rightarrow R+1$

Remainder → R

CONDITION CODE RESULTS

If an arithmetic exception (overflow or underflow) does not occur, the result is valid, and the condition codes are set as follows:

CC1: 0 (valid results)
CC2: Is set if $(R+1_{0-31})$ is greater than zero
CC3: Is set if $(R+1_{0-31})$ is less than zero
CC4: Is set if $(R+1_{0-31})$ is equal to zero

If an arithmetic exception does occur, the condition codes are set as follows:

CC1: 1 (arithmetic exception)
CC2: Is set if (R and R+1) is greater than zero
CC3: Is set if (R and R+1) is less than zero
CC4: Is set if (R and R+1) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The doubleword contents of GPR0 and GPR1 are divided by the contents of memory byte 0030BF with 24_{10} zeros prefixed. The quotient is transferred to GPR1 and the remainder is transferred to GPR0. CC2 is set.

Memory Location:	03000
Hexadecimal Instruction:	C40E30B0 (R=0, X=0, BR=6)
Assembly Language Coding:	DVMB 0,X'30B0'(6)

Before	PSD1	GPR0	GPR1	BR6	Memory Byte 0030BF
	12003000	00000000	00000139	0000000F	04
After	PSD1	GPR0	GPR1	BR6	Memory Byte 0030BF
	22003004	00000001	0000004E	0000000F	04

NONBASE REGISTER MODE EXAMPLE

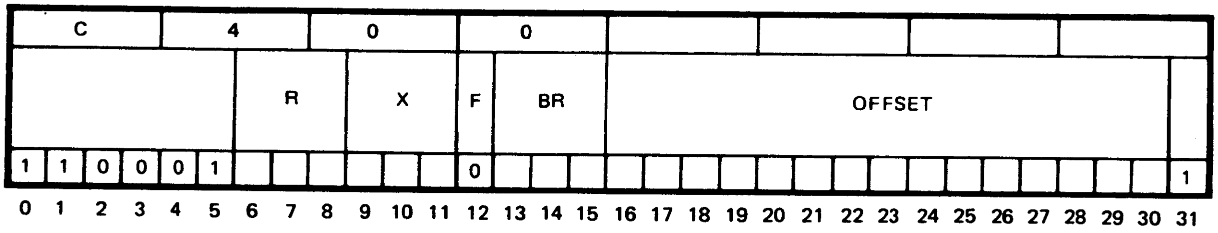
The doubleword contents of GPR0 and GPR1 are divided by the contents of memory byte 030BF with 24_{10} zeros prefixed. The quotient is transferred to GPR1 and the remainder is transferred to GPR0. CC2 is set.

Memory Location:	03000
Hexadecimal Instruction:	C4 08 30 BF (R=0, X=0, I=0)
Assembly Language Coding:	DVMB 0,X'30BF'

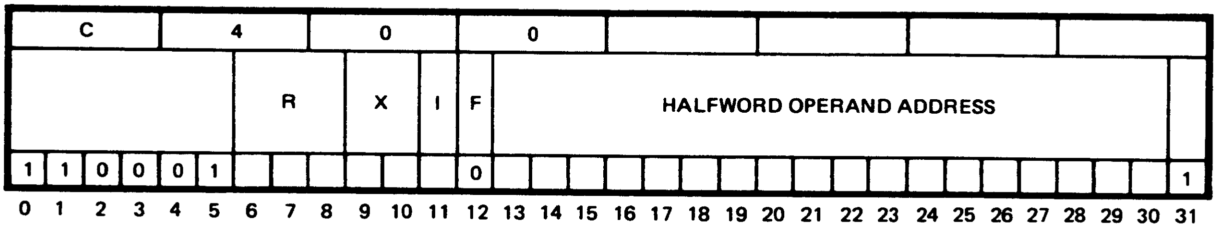
Before	PSD1	GPR0	GPR1	Memory Byte 030BF
	10003000	00000000	00000139	04
After	PSD1	GPR0	GPR1	Memory Byte 030BF
	20003004	00000001	0000004E	04

DIVIDE BY MEMORY HALFWORD C400

DVMH
d,*m,x



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

830270

DEFINITION

The contents of the effective halfword location (EHL) specified by the effective halfword address (EHA) is accessed, and the sign is extended 16 bits to the left to form a word. This word is algebraically divided into the doubleword in the general purpose register (GPR) specified by R and R+1. R+1 is the GPR one greater than specified by R. The resulting quotient is then transferred to the GPR specified by R+1 and the remainder is transferred to the GPR specified by R. The sign of the GPR specified by R (remainder) is the same as the sign of the dividend. The sign of the GPR specified by R+1 (quotient) will be the algebraic product of the signs of the dividend and the divisor, except when the absolute value of the dividend is less than the absolute value of the divisor. In the latter case, the resulting quotient (GPR specified by R+1) will be set to zero.

NOTES

1. An arithmetic exception occurs if the divisor is equal to zero, or the value of the quotient exceeds 31 bits. If an arithmetic exception occurs, the unchanged dividend will be retained in the GPR specified by R and R+1.

2. The GPR specified by R must have an even address.

SUMMARY EXPRESSION

$(R, R+1)/(EHL)_{SE} \rightarrow R+1$

Remainder → R

CONDITION CODE RESULTS

If an arithmetic exception (overflow or underflow) does not occur, the result is valid, the condition codes should be set as follows:

CC1: 0 (valid results)
CC2: Is set if ($R+1_{0-31}$) is greater than zero
CC3: Is set if ($R+1_{0-31}$) is less than zero
CC4: Is set if ($R+1_{0-31}$) is equal to zero

If an arithmetic exception does occur, the condition codes should be set as follows:

CC1: 1 (arithmetic exception)
CC2: Is set if (R and R+1) is greater than zero
CC3: Is set if (R and R+1) is less than zero
CC4: Is set if (R and R+1) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of BR6 and the instruction offset are added to obtain the logical address. The doubleword contents of GPR6 and GPR7 are divided by the contents of memory halfword 005D6A with sign extension. The quotient is transferred to GPR7 and the remainder is transferred to GPR6. CC3 is set.

Memory Location:	05A94
Hexadecimal Instruction:	C706500B (R=6, X=0, BR=6)
Assembly Language Coding:	DVMH 6,X'500A'(6)

Before	PSD1	GPR6	GPR7	BR6
	0A005A94	00000000	0000003B	00000D60
	Memory Halfword 005D6A			
	FFF8			
After	PSD1	GPR6	GPR7	BR6
	12005A98	00000005	FFFFFFF9	00000D60
	Memory Halfword 005D6A			
	FFF8			

NONBASE REGISTER MODE EXAMPLE

The doubleword contents of GPR6 and GPR7 are divided by the contents of memory halfword 05D6A with sign extension. The quotient is transferred to GPR7 and the remainder is transferred to GPR6. CC3 is set.

Memory Location:	05A94
Hexadecimal Instruction:	C7 00 5D 6B (R=6, X=0, I=0)
Assembly Language Coding:	DVMH 6,X'5D6A'

Before	PSD1	GPR6	GPR7	Memory Halfword 05D6A
	08005A94	00000000	0000003B	FFF8
After	PSD1	GPR6	GPR7	Memory Halfword 05D6A
	10005A98	00000005	FFFFFFF9	FFF8

DIVIDE BY MEMORY WORD DVMW
C400 d,*m,x

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
C				4				0				0																			
						R			X			F	BR			OFFSET															
1	1	0	0	0	1							0																		0	0

BASE REGISTER FORMAT

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
C				4				0				0																			
						R			X		I	F	WORD OPERAND ADDRESS																		
1	1	0	0	0	1							0																		0	0

NONBASE REGISTER FORMAT

830271

DEFINITION

The contents of the effective word location (EWL) specified by the effective word address (EWA) is accessed and algebraically divided into the doubleword in the general purpose register (GPR) specified by R and R+1. R+1 is the GPR one greater than specified by R. The resulting quotient is then transferred to the GPR specified by R+1, and the remainder is transferred to the GPR specified by R. The sign of the GPR specified by R (remainder) is the same as the sign of the dividend. The sign of the GPR specified by R+1 (quotient) will be the algebraic product of the signs of the dividend and the divisor, except when the absolute value of the dividend is less than the absolute value of the divisor. In the latter case, the resulting quotient (GPR specified by R+1) will be set to zero.

NOTES

1. An arithmetic exception occurs if the divisor is equal to zero, or the value of the quotient exceeds 31 bits. If an arithmetic exception occurs, the unchanged dividend will be retained in the GPR specified by R_D and R_D+1.

2. The GPR specified by R must have an even address.

SUMMARY EXPRESSION

(R,R+1)/(EWL) → R+1

Remainder → R

CONDITION CODE RESULTS

If an arithmetic exception (overflow or underflow) does not occur, the result is valid, the conditon codes will be as follows:

CC1: 0 (valid results)
CC2: Is set if $(R+1_{0-31})$ is greater than zero
CC3: Is set if $(R+1_{0-31})$ is less than zero
CC4: Is set if $(R+1_{0-31})$ is equal to zero

If an arithmetic exception occurs, the condition codes will be as follows:

CC1: 1 (arithmetic exception)
CC2: Is set if (R and R+1) is greater than zero
CC3: Is set if (R and R+1) is less than zero
CC4: Is set if (R and R+1) is equal to zero

BASE REGISTER MODE EXAMPLE

The contents of GPR3, BR6 and the instruction offset are added to obtain the logical address. The doubleword obtained from GPR4 and GPR5 is divided by the contents of memory word 007B5C. The quotient is transferred to GPR5, and the remainder is transferred to GPR4. CC4 is set.

Memory Location:	078C0
Hexadecimal Instruction:	C6367B00 (R=4, BR=6, X=3)
Assembly Language Coding:	DVMW 4, X '7B00' (6), 3

Before	PSD1	GPR4	GPR5	BR6	GPR3
	420078C0	00000000	039A20CF	00000050	0000000C
	Memory Word 007B5C				
	FC000000				
After	PSD1	GPR4	GPR5	BR6	GPR3
	A0078C4	039A20CF	00000000	00000050	0000000C
	Memory Word 007B5C				
	FC000000				

NONBASE REGISTER MODE EXAMPLE

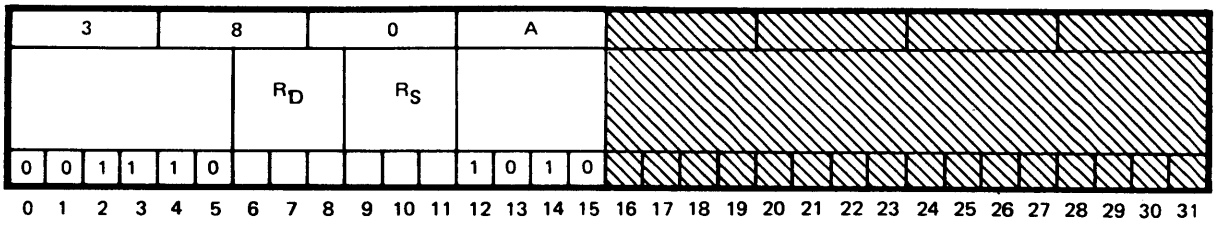
The doubleword obtained from GPR4 and GPR5 is divided by the contents of memory word 07B5C. The quotient is transferred to GPR5, and the remainder is transferred to GPR4. CC4 is set.

Memory Location:	078C0
Hexadecimal Instruction:	C6 00 7B 5C (R=4, X=0, I=0)
Assembly Language Coding:	DVMW 4,X'7B5C'

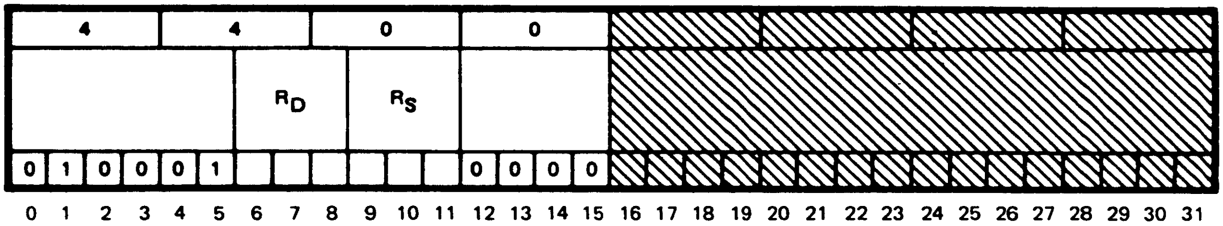
Before	PSD1	GPR4	GPR5	Memory Word 07B5C
	400078C0	00000000	039A20CF	FC000000
After	PSD1	GPR4	GPR5	Memory Word 07B5C
	080078C4	039A20CF	00000000	FC000000

DIVIDE REGISTER BY REGISTER

DVR
s,d



BASE REGISTER FORMAT (380A)



NONBASE REGISTER FORMAT (4400)

830272

DEFINITION

The word in the general purpose register (GPR) specified by R_S is algebraically divided into the doubleword in the GPR specified by R_D and R_D+1. R_D+1 is the GPR one greater than specified by R_D. The resulting quotient is then transferred to the GPR specified by R_D+1, and the remainder is transferred to the GPR specified by R_D. The sign of the GPR specified by R_D (remainder) is the same as the sign of the dividend. The sign of the GPR specified by R_D+1 (quotient) will be the algebraic product of the signs of the dividend and the divisor, except when the absolute value of the dividend is less than the absolute value of the divisor. In the latter case, the resulting quotient (GPR specified by R_D+1) will be set to zero.

NOTES

1. An arithmetic exception occurs if the divisor is equal to zero, or the value of the quotient exceeds 31 bits. If an arithmetic exception occurs, the unchanged dividend will be retained in the GPR specified by R_D and R_D+1.

2. The GPR specified by R_D must have an even address.

SUMMARY EXPRESSION

$(R_D,R_D+1)/R_S \rightarrow R_D+1$
Remainder → R_D

CONDITION CODE RESULTS

If an arithmetic exception (overflow or underflow) does not occur, the result is valid, the condition codes will be as follows:

CC1: 0 (valid results)
CC2: Is set if (R_D+1_{0-31}) is greater than zero
CC3: Is set if (R_D+1_{0-31}) is less than zero
CC4: Is set if (R_D+1_{0-31}) is equal to zero

If an arithmetic exception does occur, the condition codes will be as follows:

CC1: 1 (arithmetic exception)
CC2: Is set if (R_D and R_D+1) is greater than zero
CC3: Is set if (R_D and R_D+1) is less than zero
CC4: Is set if (R_D and R_D+1) is equal to zero

BASE REGISTER MODE EXAMPLE

The doubleword obtained from GPR6 and GPR7 is divided by the contents of GPR2. The quotient is transferred to GPR7, and the remainder is transferred to GPR6. CC2 is set.

Memory Location:	04136
Hexadecimal Instruction:	3B2A (R_D = 6, R_S = 2)
Assembly Language Coding:	DVR 2,6

Before	PSD1 12004136	GPR2 0000000A	GPR6 00000000	GPR7 000000FF
After	PSD1 22004139	GPR2 0000000A	GPR6 00000005	GPR7 00000019

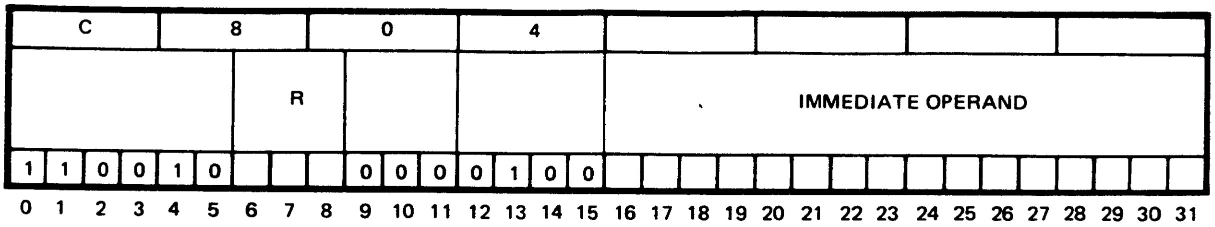
NONBASE REGISTER MODE EXAMPLE

The doubleword obtained from GPR6 and GPR7 is divided by the contents of GPR2. The quotient is transferred to GPR7, and the remainder is transferred to GPR6. CC2 is set.

Memory Location:	04136
Hexadecimal Instruction:	47 20 (R_D=6, R_S=2)
Assembly Language Coding:	DVR 2,6

Before	PSD1 10004136	GPR2 0000000A	GPR6 00000000	GPR7 000000FF
After	PSD1 20004139	GPR2 0000000A	GPR6 00000005	GPR7 00000019

DIVIDE IMMEDIATE C804 — DVI d,v



830273

DEFINITION

The sign of the least-significant halfword (bits 16-31) of the instruction word (IW) is extended 16 bits to the left to form a word. This word is algebraically divided into the doubleword in the general purpose register (GPR) specified by R and R+1. R+1 is the GPR one greater than specified by R. The resulting quotient is then transferred to the GPR specified by R+1, and the remainder is transferred to the GPR specified by R. The sign of the GPR specified by R (remainder) is the same as the sign of the dividend. The sign of the GPR specified by R+1 (quotient) will be the algebraic product of the signs of the dividend and the divisor, except when the absolute value of the dividend is less than the absolute value of the divisor. In the latter case, the resulting quotient (GPR specified by R+1) will be set to zero.

NOTES

1. An arithmetic exception occurs if the divisor is equal to zero, or the value of the quotient exceeds 31 bits. If an arithmetic exception occurs, the unchanged dividend will be retained in the GPR specified by R and R+1.

2. The GPR specified by R must have an even address.

SUMMARY EXPRESSION

$(R,R+1)/(IW_{16-31})_{SE} \rightarrow R+1$

Remainder → R

CONDITION CODE RESULTS

If an arithmetic exception (overflow or underflow) does not occur,the result is valid, the condition codes will be as follows:

CC1: 0 (valid results)
CC2: Is set if $(R+1_{0-31})$ is greater than zero
CC3: Is set if $(R+1_{0-31})$ is less than zero
CC4: Is set if $(R+1_{0-31})$ is equal to zero

If an arithmetic exception does occur, the condition codes will be as follows:

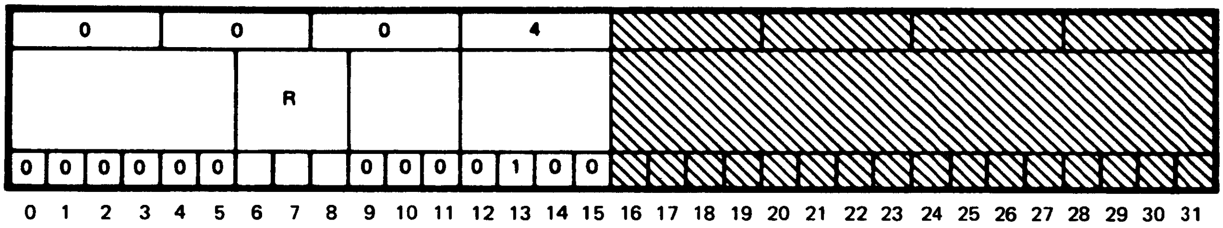
CC1: 1 (arithmetic exception)
CC2: Is set if (R and R+1) is greater than zero
CC3: Is set if (R and R+1) is less than zero
CC4: Is set if (R and R+1) is equal to zero

NONBASE AND BASE REGISTER MODE EXAMPLE

The doubleword obtained from GPR2 and GPR3 is divided by the immediate operand, sign extended. The quotient is transferred to GPR3, and the remainder is transferred to GPR2. CC3 is set.

Memory Location:		08000	
Hexadecimal Instruction:		C9 04 FF FD (R=2)	
Assembly Language Coding:		DVI 2,-3	
Before	PSD1	GPR2	GPR3
	04008000 (Nonbase)	00000000	000001B7
	02008000 (Base)		
After	PSD1	GPR2	GPR3
	10008004 (Nonbase)	00000001	FFFFFF6F
	12008004 (Base)		

EXTEND SIGN 0004 — ES d



DEFINITION

The sign (bit 0) of the contents of the general purpose register (GPR) specified by R+1 is extended through all 32 bit positions of the GPR specified by R.

SUMMARY EXPRESSION

$(R+1_0) \rightarrow R_{0-31}$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Always zero
CC3: Is set if (R_{0-31}) is less than zero
CC4: Is set if (R_{0-31}) is equal to zero

NOTE

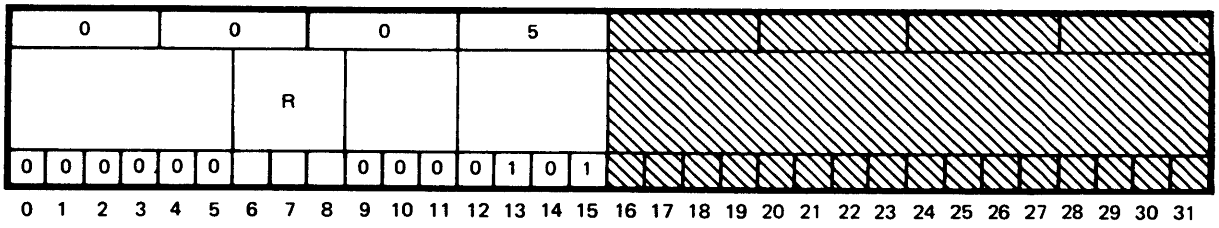
1. The register specified by R must be an even register.
2. This instruction is commonly used in preparation for the division of single word dividend occupying an odd register.

NONBASE AND BASE REGISTER MODE EXAMPLE

Bits 0-31 of general purpose register 2 are set to ones. CC3 is set.

Memory Location:	0083A
Hexadecimal Instruction:	0104 (R=2)
Assembly Language Coding:	ES 2

Before	PSD1	GPR2	GPR3
	0800083A (Nonbase)	0000B074	8000C361
	0A00083A (Base)		
After	PSD1	GPR2	GPR3
	1000083D (Nonbase)	FFFFFFFF	8000C361
	1200083D (Base)		



DEFINITION

The contents of the general purpose register (GPR) specified by R are incremented by one if bit position 0 of the GPR specified by R+1 is equal to one. R+1 is the GPR one greater than specified by R.

SUMMARY EXPRESSION

$(R)+1,\text{if}(R+1_0)=1$

CONDITION CODE RESULTS

CC1: Is set if arithmetic exception occurs
CC2: Is set if (R_{0-31}) is greater than zero
CC3: Is set if (R_{0-31}) is less than zero
CC4: Is set if (R_{0-31}) is equal to zero

NONBASE AND BASE REGISTER MODE EXAMPLE

The contents of GPR6 are incremented by one, and the result is returned to GPR6. CC2 is set.

Memory Location:	00FFE
Hexadecimal Instruction:	03 75 (R=6)
Assembly Language Coding:	RND 6

Before	PSD1	GPR6	GPR7
	40000FFE (Nonbase)	783A05B2	92CD061F
	42000FFE (Base)		
After	PSD1	GPR6	GPR7
	20001001 (Nonbase)	783A05B3	92CD061F
	22001001 (Base)		

6.2.10 Floating-Point Arithmetic Instructions

The floating-point arithmetic instructions provide the capability to add, subtract, multiply, or divide operands of large magnitude with precise results. A floating-point number is made up of three parts: a sign, a fraction, and an exponent. The sign applies to the fraction and denotes a positive or negative value. The fraction is a binary number with an assumed radix point between the sign bit and the most-significant bit. The exponent is a 7-bit binary power to which the base 16 is raised. The quantity that the floating-point number represents is obtained by multiplying the fraction by the number 16 raised to the power represented by the exponent.

6.2.10.1 Instruction Format

The floating-point arithmetic instructions use the standard memory reference and interregister formats.

6.2.10.2 Condition Code

Execution of all floating-point arithmetic instructions changes the appropriate condition code bits to indicate the result of the operation. Arithmetic exceptions are produced by overflow or underflow of the exponent value and are reflected in the state of CC1 (condition code 1).

When CC1 is a zero, an arithmetic exception has not occurred, and the result of the arithmetic operation is valid. The condition codes, therefore, indicate whether the result of the operation was greater than, less than, or equal to zero, as shown in the upper portion of the table below.

Conversely, when CC1 is a one, it indicates that an arithmetic exception has occurred, and the condition codes should be interpreted as indicated in the lower portion of the table. CC4 is also set when the overflow condition occurs. However, for overflow and underflow, either the CC2 or CC3 is used to indicate the state of what would have been the sign of the resultant fraction had the arithmetic exception not occurred.

Condition Code				Definition
CC1	CC2	CC3	CC4	
0	1	0	0	Positive fraction
0	0	1	0	Negative
0	0	0	1	Zero fraction
1	1	0	0	Exponent underflow, positive fraction
1	0	1	0	Exponent underflow, negative fraction
1	1	0	1	Exponent overflow, positive fraction
1	0	1	1	Exponent overflow, negative fraction

NOTE

> When the AE trap is enabled and an arithmetic exception condition occurs, GPR R (and R+1 in doubleword operations) retains its original operand value and the CPU is trapped into the AE handler.

When the AE trap is disabled and the execution of any floating point results in an overflow or underflow condition the CPU modifies the destination register to the following values:

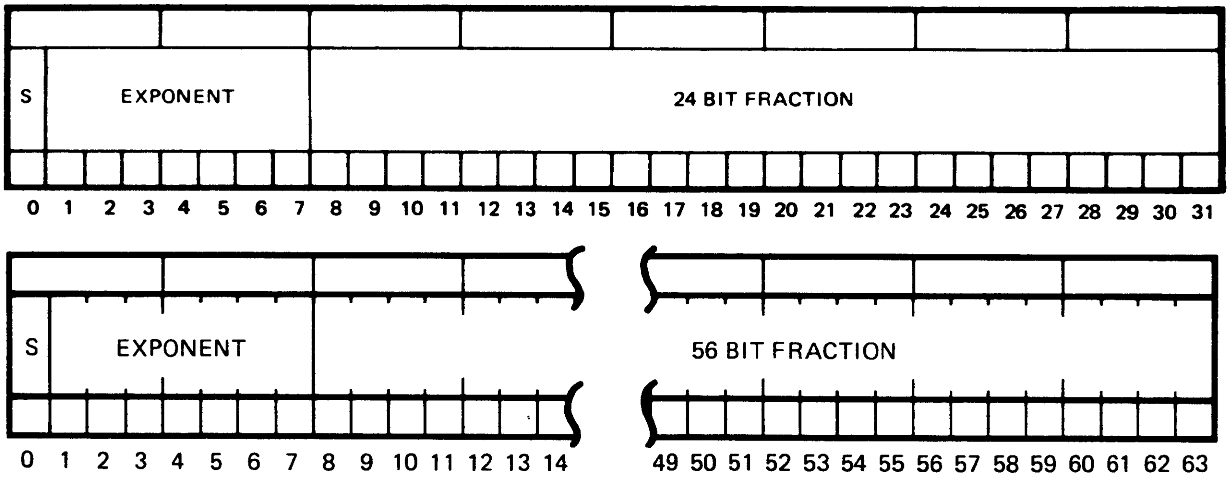
1. For positive overflow the destination register is set to 7FFFFFFF (single precision) or 7FFFFFFF FFFFFFFF (double precision).

2. For negative overflow the destination register is set to 80000001 (single precision) or 80000000 00000001 (double precision).

3. For positive or negative underflow the destination register is set to 00000000 (single precision) or 00000000 00000000 (double precision).

6.2.10.3 Floating-Point Arithmetic Operands

A floating-point number can be represented in two different formats: word and doubleword. These two formats are similar, except that the doubleword contains eight additional hexadecimal digits of significance in the fraction. These two formats are shown below.



830360A

The floating-point number, in either format, is made up of three parts: a sign, a fraction, and an exponent. The sign bit (bit 0) applies to the fraction and denotes a positive or negative value. The fraction is a hexadecimal normalized number with a radix point to the left of the highest order fraction bit (bit 8). The exponent (bits 1 through 7) is a 7-bit binary number to which the base 16 is raised.

Negative exponents are carried in the twos complement format. To remove the sign, and therefore enable exponents to be compared directly, both positive and negative exponents are biased up by 40 (hexadecimal, excess 64 (decimal) notation).

The quantity that a floating-point number represents is obtained by multiplying the fraction by the number 16 (decimal) raised to the power of the exponent minus 40 (hexadecimal).

A positive floating-point number is converted to a negative floating-point number by taking the twos complement of the positive fraction and the ones complement of the biased exponent. If the minus zero case is ruled illegitimate, all floating-point numbers can be converted from positive to negative and from negative to positive by taking the twos complement of the number in floating-point format. Signed numbers in the floating-point format can thus be compared directly, one with another, by using the compare arithmetic instructions.

All floating-point operands must be normalized before being operated on by a floating-point instruction. A positive floating-point number is normalized when the value of the fraction (F) is less than one and greater than or equal to one-sixteenth $(1 < F \geq 1/16)$. A negative floating-point number is normalized when its two complement, as a positive floating point number is normalized. All floating-point results are normalized by the CPU. If a floating-point operation results in an intermediate value of the form

1 XXX XXXX 0000...0000

the CPU will convert that result to a legitimate normalized floating-point number of the form

1 YYY YYYY 1111 0000...0000

where YYY YYYY is one less than XXX XXXX.

In single precision operations, a hexadecimal guard digit is appended to the least-significant hexadecimal digit of the floating-point word operands by the CPU. The most-significant bit of the guard digit is used as the basis for rounding by the CPU at the end of every floating-point word computation. If Guard + 1 is equal to one then rounding occurs. If Guard + 1 is equal to zero the truncating occurs.

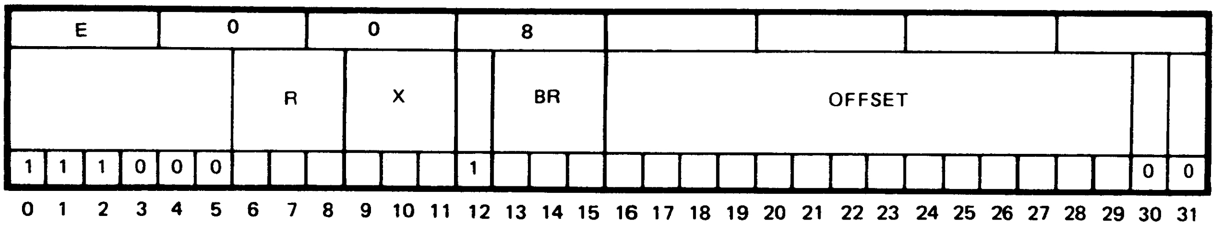
NOTES

1. In conversion of a doubleword floating-point value to a word floating-point value by truncation of the least significant word, an unnormalized negative floating-point number can result. Example: BF00000000000001 is normalized as a doubleword floating-point value, but BF000000 is not a normalized word floating-point value since its two complement is 41000000 (fraction part is zero) . In such a case, a floating-point add of a word of all zeros will result in a properly normalized value. (In the example BF000000 will be transformed to BEF00000 by use of ADFW R,=0). This is a permissable exception to the general rule that all floating-point instruction operands must be normalized.

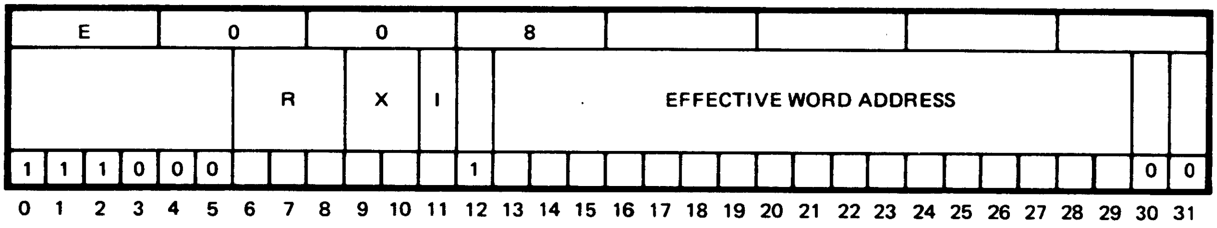
2. For divide operations only, floating point arithmetic performed by the hardware floating point accelerator (FPA) may have slightly different results in the LSB of single precision results or the four LSBs of double precision results or the four LSBs of double precision results than the arithmetic performed by the firmware.

3. FPA hardware is used if present and enabled by the software via the SETCPU instruction (CPU status word bit 22) otherwise floating point is performed by the CPU's firmware.

This page intentionally left blank



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

830276

DEFINITION

The addend is the floating-point word operand found in the effective word location (EWL) specified by the effective word address (EWA). The augend of the operation is contained in the general purpose register (GPR) specified by R in the instruction word. Both of these operands are accessed, and if either one or both are negative, the ones complement of the base 16 exponent (bits 1-7) of the negative number is taken. Both exponents of the word operands are stripped of their 40_{16} bias and algebraically compared.

If the two exponents are equal, the 24-bit signed fractions of the addend and augend are added algebraically. If the exponents differ, and the absolute value of the difference is greater than zero, but equal to or less than six ($0 >$ exponent difference≤ 6), the operand containing the smaller exponent needs adjustment. The fraction of this operand is shifted right and the exponent is incremented once for each shift until the two exponents are equal. After the exponents are equalized, the fractions are added algebraically. The normalized and rounded sum of the two fractions is assembled with the incremented exponent that has been biased up by 40_{16}.

If the resultant fraction is negative, the ones complement of the exponent is used. When the exponent difference is greater than six, the operand that contains the larger exponent is normalized and considered to be the answer.

NOTES

1. If the result fraction equals zero, the exponent and fraction are set to zero in the GPR specified by R.

2. Operands are expected to be normalized.

SUMMARY EXPRESSION

(R)+(EWL) → R

CONDITION CODE RESULTS

CC1: Is set if arithmetic exception occurs
CC2: Is set if $(R_{0,8-31})$ is greater than zero
CC3: Is set if $(R_{0,8-31})$ is less than zero
CC4: Is set if $(R_{0,8-31})$ is equal to zero

BASE REGISTER MODE EXAMPLE

Memory Location:	008DC
Hexadecimal Instruction:	E34E0500 (R=6, BR=6, X=4)
Assembly Language Coding:	ADFW 6, X '0500' (6), 4

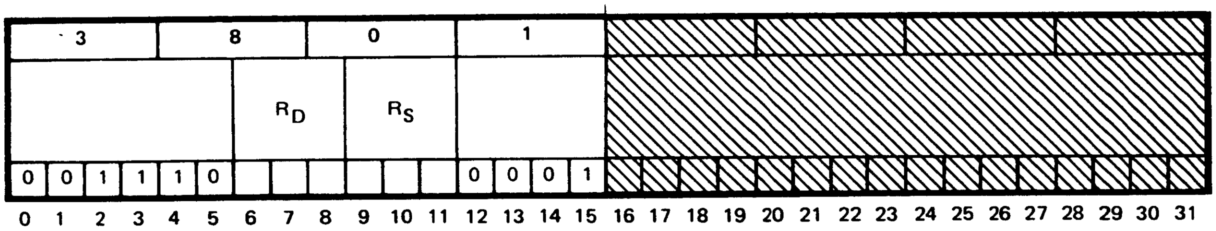
Before	PSD1	GPR6	BR6	GPR4	Memory Word 0570
	020008DC	41100000	00000070	00000000	41200000
After	PSD1	GPR6	BR6	GPR4	Memory Word 0570
	220008E0	41300000	00000070	00000000	41200000

NONBASE REGISTER MODE EXAMPLE

Memory Location:	008DC
Hexadecimal Instruction:	E3 08 05 70 (R=6, X=0, I=0)
Assembly Language Coding:	ADFW 6,X'00570'

Before	PSD1	GPR6	Memory Word 0570
	000008DC	41100000	41200000
After	PSD1	GPR6	Memory Word 0570
	200008E0	41300000	41200000

ADD FLOATING-POINT WORD REGISTER-TO-REGISTER **ADRFW**
3801 **s,d**



830277

DEFINITION

The floating-point numbers contained in the general purpose registers specified by R_S (addend) and R_D (augend) are accessed. If either of the floating-point numbers is negative, the one's complement of the base 16 exponent (bits 1-7) is taken for the negative number. Both exponents are then stripped of their 40_{16} bias and algebraically compared. If the two exponents are equal, the signed fractions of the two numbers are algebraically added. If the exponents differ, and the absolute value of the difference is greater than zero, or less than or equal to six (0 exponent difference 6), the fraction of the operand containing the smaller exponent is shifted right by a number of hexadecimal digits corresponding to the value of the exponent absolute difference. After exponent equalization, the fractions are added algebraically. The normalized and rounded sum of the two fractions is placed in bit positions 0 and 8-31 of GPR R_D. The result exponent is biased up by 40_{16} and, if the result fraction is negative, the one's complement of the exponent is placed in bit positions 1-7 of GPR R_D.

NOTES

1. If the result fraction equals zero, the exponent and fraction are set to zero in GPR specified by R_D.

2. Operands are expected to be normalized.

SUMMARY EXPRESSION

$(R_D) + (R_S) \rightarrow R_D$

CONDITION CODE RESULTS

CC1: Is set if arithmetic exception occurs
CC2: Is set if $(R_{0,\ 8-31})$ is greater than zero
CC3: Is set if $(R_{0,\ 8-31})$ is less than zero
CC4: Is set if $(R_{0,\ 8-31})$ is equal to zero

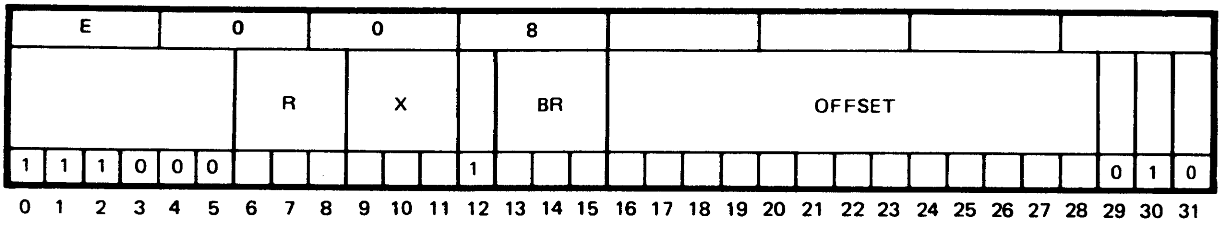
NONBASE AND BASE REGISTER MODE EXAMPLE

The contents of R6 (R_D) and R2 (R_S) are added and the result is transferred to R6 (R_D).

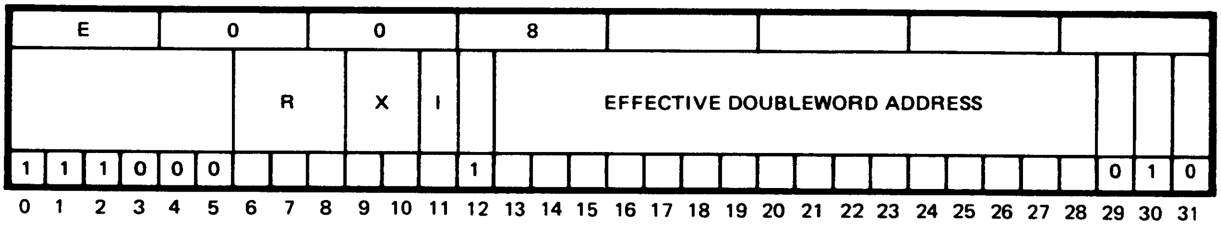
Memory Location:	1000
Hexadecimal Instruction:	3B21 (R_D = 6, R_S = 2)
Assembly Language Coding:	ADRFW 2,6

Before	PSD1 00001000 (Nonbase) 02001000 (Base	GPR6 41100000	GPR2 41200000
After	PSD1 20001002 (Nonbase) 22001002 (Base)	GPR6 41300000	GPR2 41200000

ADD FLOATING-POINT DOUBLEWORD **ADFD**
E008 **d,*m,x**



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

830278

DEFINITION

The addend is the floating-point doubleword operand found in the memory location specified by the effective doubleword address (EDA). The augend of the operation is contained in two general purpose registers (GPR) referred to as R and R+1. Both of these operands are accessed, and if either one or both are negative, the ones complement of the base 16 exponent (bits 1-7) of the negative doubleword is taken. Both exponents of the word operands are then stripped of their 40_{16} bias and algebraically compared.

If the two exponents are equal, the 56-bit signed fractions of the addend and augend are added algebraically. If the exponents differ, and the absolute value of the difference is greater than zero, but equal to or less than 13 (0 exponent difference 13), the operand containing the smaller exponent needs to be adjusted. The fraction of this operand is shifted right and the exponent is incremented once for each shift until the two exponents are equal. After the exponents are equalized, the fractions are added algebraically. The normalized sum of the two fractions is assembled with the incremented exponent that has been biased up by 40_{16}.

If the resultant fraction is negative, the ones complement of the exponent is used. When the exponent difference is greater than 13, the operand that contains the larger exponent is normalized and considered to be the answer. The assembled sign, exponent, and fraction are transferred to the GPR locations R and R+1.

NOTES

1. If the result fraction equals zero, the exponent and fraction are set to zero in GPRs R and R+1.

2. Operands are expected to be normalized.

3. The GPR specified by R must be an even-numbered register.

SUMMARY EXPRESSION

(R),(R+1)+(EWL),(EWL+1) → R,R+1

CONDITION CODE RESULTS

CC1: Is set if arithmetic exception occurs
CC2: Is set if $(R_{0,8-31}, R+1_{0-31})$ is greater than zero
CC3: Is set if $(R_{0,8-31}, R+1_{0-31})$ is less than zero
CC4: Is set if $(R_{0,8-31}, R+1_{0-31})$ is equal to zero

BASE REGISTER MODE EXAMPLE

Memory Location:	00FE8
Hexadecimal Instruction:	E30E0502 (R=6, X=0, BR=6)
Assembly Language Coding:	ADFD 6,X'500'(6)

Before	PSD1	GPR6,7	BR6	Memory Doubleword 0570, 0574
	02000FE8	42200000	00000070	41100000
		20000000		10000000
After	PSD1	GPR6,7	BR6	Memory Doubleword 0570, 0574
	22000FEC	42210000	00000070	41100000
		21000000		10000000

NONBASE REGISTER MODE EXAMPLE

Memory Location:	00FE8
Hexadecimal Instruction:	E3 08 05 72 (R=6, X=0, I=0)
Assembly Language Coding:	ADFD 6,X'572'

Before	PSD1	GPR6,7	Memory Doubleword 0570, 0574
	00000FE8	42200000	41100000
		20000000	10000000
After	PSD1	GPR6,7	Memory Doubleword 0570, 0574
	20000FEC	42210000	41100000
		21000000	10000000

3				8				0				9				▨				▨				▨				▨			
						R_D			R_S							▨															
0	0	1	1	1	0							1	0	0	1	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

DEFINITION

The floating-point doublewords contained in the general purpose registers specified by R_S and R_S+1 (addend) and R_D and R_D+1 (augend) are accessed. If either of the floating-point numbers is negative, the one's complement of the base 16 exponent (bit 1-7) is taken of the negative number. Both exponents are then stripped of their 40_{16} bias and algebraically compared. If the two exponents are equal, the signed fractions of the two numbers are algebraically added. If the exponents differ, and the absolute value of the difference is greater than zero, or less than or equal to 13 (0 exponent difference 13) the operand containing the smaller exponent needs to be adjusted. The fraction of this exponent is shifted right and the exponent is incremented once for each shift until the two exponents are equal. If the exponent difference is greater than 13, the operand that contains the larger exponent is normalized and considered to be the answer. After exponent equalization, the fractions are added algebraically. The normalized sum of the two fractions is placed in bit positions 0 and 8-31 of GPR R_D and bit positions 0-31 of the GPR R_D+1. The result exponent is biased up to 40_{16} and, if the result fraction is negative, the one's complement of the exponent is placed in bit positions 1-7 in GPR R_D.

NOTES

1. If the result fraction equals zero, the exponent and fraction are set to zero in GPRs R_D, R_D+1.
2. Operands are expected to be normalized.
3. The GPRs specified by R_D and R_S must be even-numbered registers.

SUMMARY EXPRESSION

$(R_D,R_D+1) + (R_S,R_S+1) \rightarrow R_D,R_D+1$

CONDITION CODE RESULTS

CC1: Is set if arithmetic exception
CC2: Is set if $(R_{0,8-31})$ is greater than zero
CC3: Is set if $(R_{0,8-31})$ is less than zero
CC4: Is set if $(R_{0,8-31})$ is equal to zero

NONBASE AND BASE REGISTER MODE EXAMPLE

The contents of GPR6 and GPR7 are added with the contents of GPR2 and GPR3; the result is transferred to GPR6 and GPR7.

Memory Location:	01000
Hexadecimal Instruction:	3B29 (R_D = 6, R_S = 2)
Assembly Language Coding:	ADRFD 2,6

Before	PSD1	GPR6,7	GPR2,3
	00001000 (Nonbase)	42200000	41100000
	02001000 (Base)	20000000	10000000
After	PSD1	GPR6,7	GPR2,3
	20001002 (Nonbase)	42210000	41100000
	22001002 (Base)	21000000	10000000

SUBTRACT FLOATING-POINT WORD
E000

SUFW
d,*m,x

E				0				0				0																			
						R			X				BR			OFFSET															
1	1	1	0	0	0							0																		0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

BASE REGISTER FORMAT

E				0				0				0																			
						R			X		I		EFFECTIVE WORD ADDRESS																		
1	1	1	0	0	0							0																		0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

NONBASE REGISTER FORMAT

DEFINITION

The subtrahend is the floating-point word operand found in the effective word location (EWL) specified by the effective word address (EWA). The minuend of the operation is contained in the general purpose register (GPR) specified by R in the instruction word. Both of these operands are accessed, and if either one or both of the floating-point words are negative, the ones complement of the base 16 exponent (bits 1-7) of the negative number is taken. Both exponents of the operands are then stripped of their 40_{16} bias and algebraically compared.

If the two exponents are equal, the 24-bit signed fractions of the subtrahend and minuend are subtracted algebraically. If the exponents differ, and the absolute value of the difference is greater than zero, but equal to or less than six (0 exponent difference 6), the operand containing the smaller exponent needs to be adjusted. The fraction of this operand is shifted right and the exponent is incremented once for each shift until the two exponents are equal. After the exponents are equalized, the fractions are subtracted algebraically. The normalized, and rounded difference between the two fractions is assembled with the incremented exponent that has been biased up by 40_{16}.

If the resultant fraction is negative, the ones complement of the exponent is used. When the exponent difference is greater than six, the operand that contains the larger exponent is normalized and considered to be the answer. The assembled sign, exponent, and fraction portions are transferred to the GPR specified by R.

NOTE

1. If the result fraction is equal to zero, the exponent and fraction are set to zero in the GPR specified by R.

2. Operands are expected to be normalized.

SUMMARY EXPRESSION

(R)-(EWL) → R

SUBTRACT FLOATING-POINT WORD (Cont.)

SUFW
d,*m,x

CONDITION CODE RESULTS

CC1: Is set if arithmetic exception occurs
CC2: Is set if $(R_{0,8-31})$ is greater than zero
CC3: Is set if $(R_{0,8-31})$ is less than zero
CC4: Is set if $(R_{0,8-31})$ is equal to zero

BASE REGISTER MODE EXAMPLE

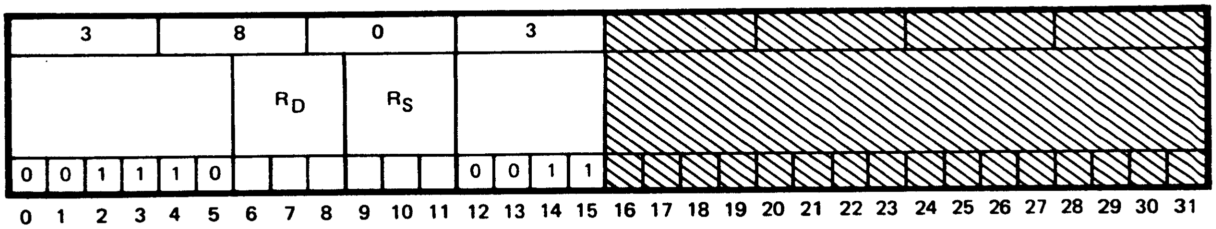
Memory Location:	00A9C
Hexadecimal Instruction:	E3460500 (R=6, BR=6, X=4)
Assembly Language Coding:	SUFW 6, X '0500' (6), 4

Before	PSD1	GPR6	BR6	GPR4	Memory Word 000570
	02000A9C	41100000	00000070	00000000	41200000
After	PSD1	GPR6	BR6	GPR4	Memory Word 000570
	12000AA0	BEF00000	00000070	00000000	41200000

NONBASE REGISTER MODE EXAMPLE

Memory Location:	00A9C
Hexadecimal Instruction:	E3 00 05 70 (R=6, X=0, I=0)
Assembly Language Coding:	SUFW 6,X'570'

Before	PSD1	GPR6	Memory Word 570
	00000A9C	41100000	41200000
After	PSD1	GPR6	Memory Word 570
	10000AA0	BEF00000	41200000



830281

DEFINITION

The floating-point numbers contained in the general purpose registers specified by R_S (subtrahend) and R_D (minuend) are accessed. If either the floating-point number in the GPR or memory is negative, the ones complement of the base 16 exponent (bits 1-7) is taken. Both exponents are then stripped of their 40_{16} bias and algebraically compared.

If the two exponents are equal, the 24-bit signed fractions are algebraically subtracted. If the exponents differ, and the absolute value of the difference is greater than zero, or equal to or less than six (0 exponent difference 6), the fraction of the operand containing the smaller exponent must be equalized. The exponent of this operand is effectively incremented by one each time the fraction is shifted right one hexadecimal digit until the exponents of both operands are equal. After exponent equalization, the fractions are subtracted algebraically. The normalized and rounded difference between the two fractions is placed in bit positions 0 and 8-31 of GPR R_D. The result exponent is biased up by 40_{16}, and, if the result fraction is negative, the ones complement of the exponent is placed in bit positions 1-7 of GPR R_D.

NOTES

1. If the result fraction is equal to zero, the exponent and fraction are set to zero in the GPR R_D.
2. Operands are expected to be normalized.

SUMMARY EXPRESSION

$(R_D) - (R_S) \rightarrow R_D$

CONDITION CODES

CC1: Is set if arithmetic exception occurs
CC2: Is set if $(R_{0,\ 8-31})$ is greater than zero
CC3: Is set if $(R_{0,\ 8-31})$ is less than zero
CC4: Is set if $(R_{0,\ 8-31})$ is equal to zero

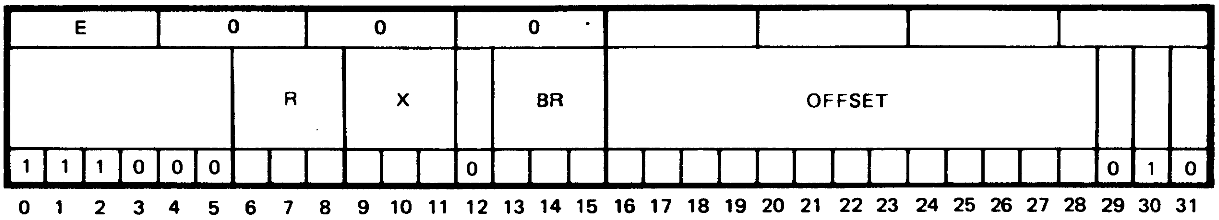
NONBASE AND BASE REGISTER MODE EXAMPLE

The contents of R6 (R_D) are subtracted from the contents of R2 (R_S) and, the result is transferred to R6 (R_D).

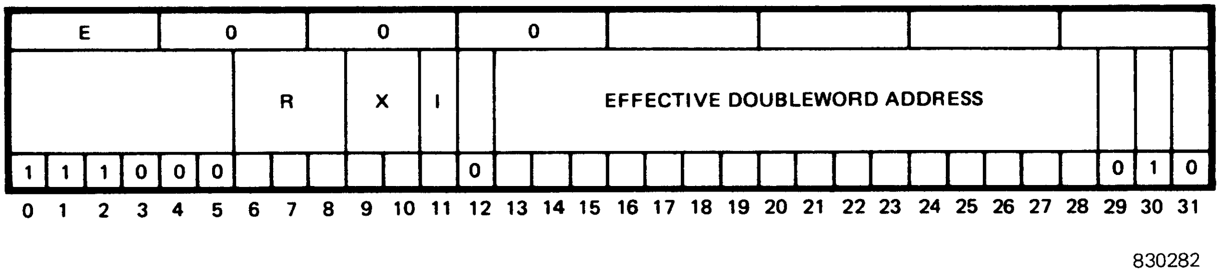
Memory Location:		01000	
Hexadecimal Instruction:		3B23 (R_D = 6, R_S = 2)	
Assemb' [illegible]		[illegible]	
Before	PSD1	GPR6	GPR2
	00001000 (Nonbase)	41100000	41200000
	02001000 (Base)		
After	PSD1	GPR6	GPR2
	10001002 (Nonbase)	BEF00000	41200000
	12001002 (Base)		

SUBTRACT FLOATING-POINT DOUBLEWORD E000

SUFD d,*m,x



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

DEFINITION

The subtrahend is the floating-point doubleword operand found in the effective word locations (EWL, EWL+1) specified by the effective doubleword address (EDA). The minuend of the operation is contained in the two general purpose registers (GPR) referred to as R and R+1. Both of these operands are accessed and, if either one or both are negative, the ones complement of the base 16 exponent (bits 1-7) of the negative doubleword is taken. Both exponents of the operands are then stripped of their 40_{16} bias and algebraically compared.

If the two exponents are equal, the 56-bit signed fractions of the subtrahend and minuend are subtracted algebraically. If the exponents differ, and the absolute value of the difference is greater than zero, but equal to or less than 13 (0 exponent difference 13), the operand containing the smaller exponent needs to be adjusted. The fraction of this operand is shifted right and the exponent is incremented once for each shift until the two exponents are equal. After the exponents are equalized, the fractions are subtracted algebraically. The normalized difference between the two fractions is assembled with the incremented exponent that has been biased up by 40_{16}.

If the resultant fraction is negative, the ones complement of the exponent is used. When the exponent difference is greater than 13, the operand that contains the larger exponent is normalized and considered to be the answer. The assembled sign, exponent, and fraction are transferred to the GPR locations R and R+1.

NOTES

1. If the result fraction is equal to zero, the exponent and fraction are set to zero in the GPRs specified by R and R+1.

2. Operands are expected to be normalized.

3. The GPR specified by R must be an even-numbered register.

SUMMARY EXPRESSION

(R),(R+1)-(EWL),(EWL+1) → R,R+1

CONDITION CODE RESULTS

CC1: Is set if arithmetic exception occurs
CC2: Is set if $(R_{0,8-31}, R+1_{0-31})$ is greater than zero
CC3: Is set if $(R_{0,8-31}, R+1_{0-31})$ is less than zero
CC4: Is set if $(R_{0,8-31}, R+1_{0-31})$ is equal to zero

BASE REGISTER MODE EXAMPLE

Memory Location:	0125C
Hexadecimal Instruction:	E3060502 (R=6, X=0, BR=6)
Assembly Language Coding:	SUFD 6,X'500'(6)

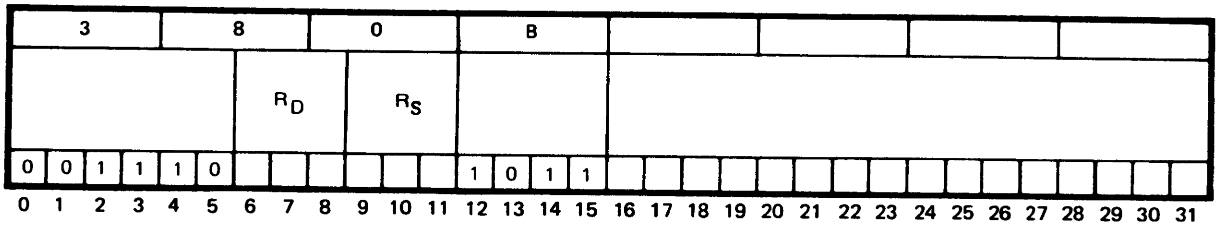
Before	PSD1	GPR6,7	BR6	Memory Doubleword 000570, 000574
	0200125C	41333333	00000070	41222222
		33333333		22222222
After	PSD1	GPR6,7	BR6	Memory Doubleword 000570, 000574
	22001260	41111111	00000070	41222222
		11111111		22222222

Memory Location:	0125C
Hexadecimal Instruction:	E3 00 05 72 (R=6, X=0, I=0)
Assembly Language Coding:	SUFD 6,X'572'

Before	PSD1	GPR6,7	Memory Doubleword 570,574
	0000125C	41333333	41222222
		33333333	22222222
After	PSD1	GPR6,7	Memory Doubleword 570,574
	20001260	41111111	41222222
		11111111	22222222

SUBTRACT FLOATING-POINT DOUBLEWORD REGISTER TO REGISTER 380B

SURFD s,d



830283

DEFINITION

The floating-point numbers contained in the general purpose registers specified by R_S and R_S+1 (subtrahend) and R_D and R_D+1 (minuend) are accessed. If either or both of the floating-point numbers are negative, the ones complement of the base 16 exponent (bits 1-7) of the negative doubleword(s) is taken. Both exponents are then stripped of their 40_{16} bias and algebraically subtracted. If the exponents differ, and the absolute value of the difference is greater than zero or less than or equal to thirteen (0 exponent difference 13), the fraction of the operand containing the smaller exponent is shifted right one hexadecimal digit at a time, while its exponent is incremented by one, until the exponents are equalized. After exponent equalization, the fractions are subtracted algebraically. The normalized and rounded difference between the two fractions is placed in bit positions 0 and 8-31 of GPR R_D and 0-31 of GPR R_D+1. The result exponent is biased up by 40_{16}, and, if the result fraction is negative, the ones complement of the exponent is placed in bit positions 1-7 of GPR R_D.

NOTES

1. If the result fraction is equal to zero, the exponent and fraction are set to zero in GPRs R_D, R_D+1.
2. Operands are expected to be normalized.
3. The GPR's specified by R_D and R_S must be even-numbered registers.

SUMMARY EXPRESSION

$(R_D,R_D+1) - (R_S,R_S+1) \rightarrow R_D,R_D+1$

CONDITION CODES

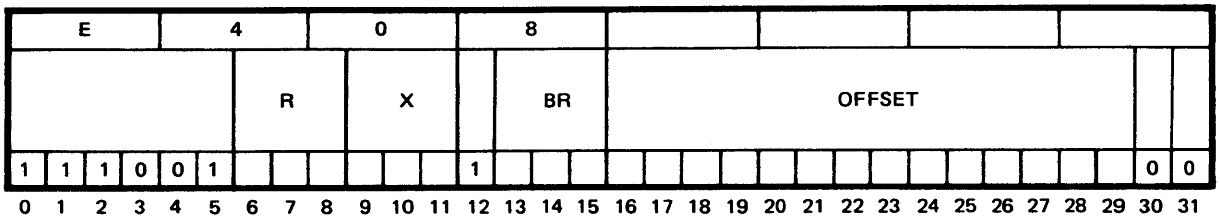
CC1: Is set if arithmetic exception occurs
CC2: Is set if $(R_{0,\ 8-31})$ is greater than zero
CC3: Is set if $(R_{0,\ 8-31})$ is less than zero
CC4: Is set if $(R_{0,\ 8-31})$ is equal to zero

NONBASE AND BASE REGISTER MODE EXAMPLE

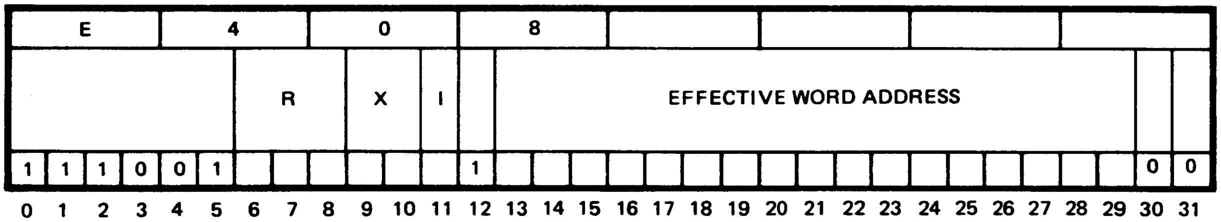
The contents of GPR6 and GPR7 are subtracted from the contents of GPR2 and GPR3; the result is transferred to GPR6 and GPR7.

Memory Location:	01000
Hexadecimal Instruction:	3B2B (R_D = 6, R_S = 2)
Assembly Language Coding:	SURFD 2,6

Before	PSD1	GPR6,7	GPR2,3
	00001000 (Nonbase)	42200000	41100000
	02001000 (Base)	20000000	10000000
After	PSD1	GPR6,7	GPR2,3
	20001002 (Nonbase)	421F0000	41100000
	22001002 (Base)	1F000000	10000000



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

830284

DEFINITION

The multiplicand is the floating-point word operand found in the effective word location (EWL) specified by the effective word address (EWA). The multiplier of the operation is contained in the general purpose register (GPR) specified by R in the instruction word. Both of these operands are accessed, and the 24-bit signed fraction (bit 0 and bits 8-31) of the multiplicand is multiplied by the signed fraction of the multiplier. If either one or both of the word operands are negative, the exponent (bits 1-7) of the negative number is changed to its ones complement. Both exponents are then stripped of their 40_{16} bias and algebraically added to obtain the initial value of the result exponent; this value may be decremented by one during the ensuing normalization.

The normalized, rounded, result of the multiplication operation is assembled with the (possibly adjusted) exponent sum that has been biased up by 40_{16}.

However, if the final fraction is negative, the ones complement of the exponent is used. The assembled sign, exponent, and fraction are transferred to the GPR specified by R.

NOTES

1. If result fraction is zero, both exponent and fraction are set to zero in the GPR specified by R.
2. Operands are expected to be normalized.

SUMMARY EXPRESSION

$(EWL_{0,8-31})x(R_{0,8-31}) \rightarrow R_{0,8-31}$

$(EWL_{1-7})+(R_{1-7}) \rightarrow R_{1-7}$

CONDITION CODE RESULTS

CC1: Is set if arithmetic exception occurs
CC2: Is set if $(R_{0,8-31})$ is greater than zero
CC3: Is set if $(R_{0,8-31})$ is less than zero
CC4: Is set if $(R_{0,8-31})$ is equal to zero

BASE REGISTER MODE EXAMPLE

Memory Location:	00C68
Hexadecimal Instruction:	E74E0500 (R=6, BR=6, X=4)
Assembly Language Coding:	MPFW 6, X '0500' (6), 4

Before	PSD1	GPR6	BR6	GPR4	Memory Word 000570
	02000C68	41200000	00000070	00000000	41300000
After	PSD1	GPR6	BR6	GPR4	Memory Word 000570
	22000C6C	41600000	00000070	00000000	41300000

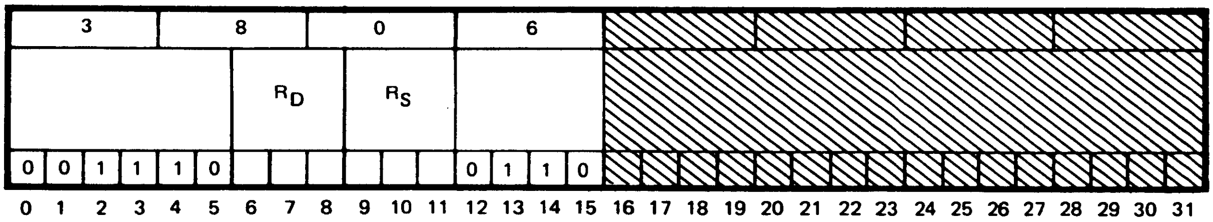
NONBASE REGISTER MODE EXAMPLE

Memory Location:	00C68
Hexadecimal Instruction:	E7 08 05 70 (R=6, X=0, I=0)
Assembly Language Coding:	MPFW 6,X'570'

Before	PSD1	GPR6	Memory Word 570
	00000C68	41200000	41300000
After	PSD1	GPR6	Memory Word 570
	20000C6C	41600000	41300000

MULTIPLY FLOATING-POINT WORD REGISTER TO REGISTER 3806

MPRFW s,d



830285

DEFINITION

The 24-bit signed fraction (bits 0, 8-31) contained in the general purpose register specified by R_S (multiplicand) is multiplied by the fraction contained in the GPR specified by R_D (multiplier). If either one or both of the floating-point numbers are negative, the exponent of the negative number is changed to its ones complement. Both exponents are then stripped of their 40_{16} bias and algebraically added to obtain the initial value of the result exponent; this value may be decremented by one during the ensuing normalization. The normalized and rounded result of the multiplication is placed in bits 0 and 8-31 of GPR R_D. The result exponent is biased up by 40_{16} and, if the result fraction is negative, the one's complement of the result exponent is placed in bits 1-7 of GPR R_D.

NOTES

1. If result fraction is zero, both exponent and fraction are set to zero in the GPR specified by R.
2. Operands are expected to be normalized.

SUMMARY EXPRESSION

$(R_S\ 0,8\text{-}31) \times (R_D\ 0,8\text{-}31) \rightarrow R_D\ 0,8\text{-}31$

$(R_S\ 1\text{-}7) + (R_D\ 1\text{-}7) \rightarrow R_D\ 1\text{-}7$

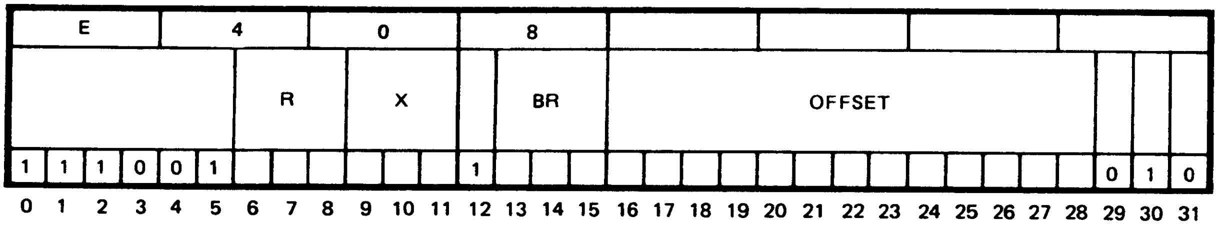
CONDITION CODE RESULTS

CC1: Is set if arithmetic exception occurs
CC2: Is set if ($R_{0,\ 8-31}$) is greater than zero
CC3: Is set if ($R_{0,\ 8-31}$) is less than zero
CC4: Is set if ($R_{0,\ 8-31}$) is equal to zero

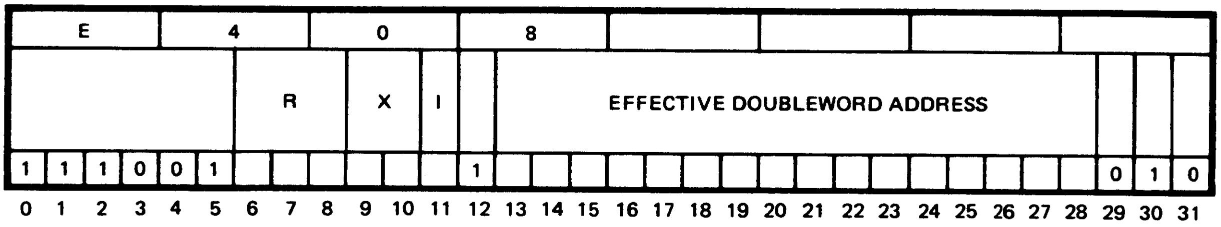
NONBASE AND BASE REGISTER MODE EXAMPLE

Memory Location:	01000
Hexadecimal Instruction:	3B26 (R_D = 6, R_S = 2)
Assembly Language Coding:	MPRFW 2,6

Before	PSD1	GPR6	GPR2
	00010000 (Nonbase)	41200000	41300000
	02010000 (Base)		
After	PSD1	GPR6	GPR2
	20010002 (Nonbase)	41600000	41300000
	22010002 (Base)		



BASE REGISTER FORMAT



830286

NONBASE REGISTER FORMAT

DEFINITION

The multiplicand is the floating-point doubleword operand found in the effective word locations (EWL, EWL+1) specified by the effective doubleword address (EDA). The multiplier of the operation is contained in the two general purpose registers (GPR) referred to as R and R+1. Both of these operands are accessed, and the 56-bit signed fraction (bits 0 and 8-31 of the first memory word and bits 0-31 of the second memory word) is multiplied by the signed fraction of the multiplier. The 56-bit signed fraction of the multiplier is made up of bits 0 and 8-31 of the GPR specified by R, and bits 0-31 of the GPR specified by R+1. If either one or both of the doubleword operands is negative, the ones complement of the base 16 exponent (bits 1-7) of the negative doubleword is taken. Both exponents are then stripped of their 40_{16} bias and algebraically added to obtain the initial value of the result exponent; this value may be decremented by one during the ensuing normalization.

The normalized result of the multiplication operation is assembled with the exponent sum that has been biased up by 40_{16}. However, if the final fraction is negative, the ones complement of the exponent is used. The assembled sign, exponent, and fraction are transferred to the GPR locations R and R+1.

NOTES

1. If result fraction is zero, both exponent and fraction are set to zero in the GPR specified by R.
2. Operands are expected to be normalized.
3. The GPR specified by R must be an even-numbered register.

SUMMARY EXPRESSION

$$(EWL_{0,8\text{-}31}), (EWL{+}1_{0\text{-}31}) \times (R_{0,8\text{-}31}, R{+}1_{0\text{-}31}) \rightarrow R_{0,8\text{-}31}, R{+}1_{0\text{-}31}$$
$$(EWL_{1\text{-}7}){+}(R_{1\text{-}7}) \rightarrow R_{1\text{-}7}$$

CONDITION CODE RESULTS

CC1: Is set if arithmetic exception occurs
CC2: Is set if ($R_{0,8-31}$, $R+1_{0-31}$) is greater than zero
CC3: Is set if ($R_{0,8-31}$, $R+1_{0-31}$) is less than zero
CC4: Is set if ($R_{0,8-31}$, $R+1_{0-31}$) is equal to zero

BASE REGISTER MODE EXAMPLE

Memory Location:	014D0
Hexadecimal Instruction:	E70E0502 (R=6, X=0, BR=6)
Assembly Language Coding:	MPFD 6,X'500'(6)

Before	PSD1	GPR6,7	BR6	Memory Doubleword 000570, 000574
	020014D0	BEF00000	00000070	41200000
			00000000	00000000
After	PSD1	GPR6,7	BR6	Memory Doubleword 000570, 000574
	120014D4	BEE00000	00000070	41200000
			00000000	00000000

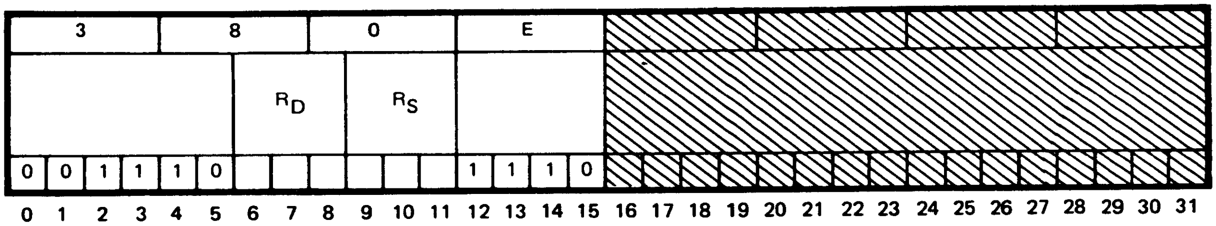
NONBASE REGISTER MODE EXAMPLE

Memory Location:	014D0
[illegible]	[illegible]

Before	PSD1	GPR6,7	Memory Doubleword 570,574
	000014D0	BEF00000	41200000
		00000000	00000000
After	PSD1	GPR6,7	Memory Doubleword 570,574
	100014D4	BEE00000	41200000
		00000000	00000000

MULTIPLY FLOATING-POINT DOUBLEWORD REGISTER TO REGISTER 380E

MPRFD s,d



830287

DEFINITION

The 56-bit signed fraction (multiplicand) contained in the general purpose registers specified by R_S (bits 0, 8-31) and R_S+1 (bits 0-31) is multiplied by the fraction (multiplier) contained in the GPRs specified by R_D and R_D+1. If either one or both of the floating-point numbers are negative, the exponent of the negative number is changed to its ones complement. Both exponents are then stripped of their 40_{16} bias and algebraically added to obtain the initial value of the result exponent; this value may be decremented by one during the ensuing normalization. The normalized result of the multiplication is transferred to bits 0 and 8-31 of GPR R_D and 0-31 of GPR R_D+1. The result exponent is biased up by 40_{16}, and if the result fraction is negative, the ones complement of the result exponent is placed in bits 1-7 of the GPR R_D.

NOTES

1. If result fraction is zero, both exponent and fraction are set to zero in the GPR specified by R.

2. Operands are expected to be normalized.

3. The GPRs specified by R_D and R_S must be even-numbered registers.

SUMMARY EXPRESSION

$(R_{S0,8-31}), (R_{S+1\ 0-31}) \times (R_{D\ 0,8-31}),(R_{D+1\ 0-31}) \rightarrow R_{D0,8-31}, R_{D+1\ 0-31}$

$(R_{S1-7})+(R_{D1-7}) \rightarrow R_{D1-7}$

CONDITION CODE RESULTS

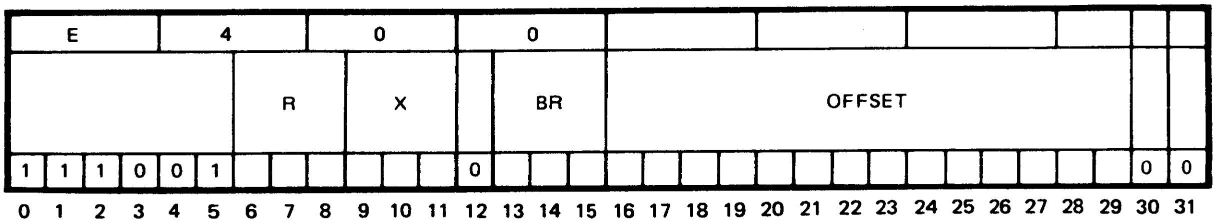
CC1: Is set if the arithmetic exception occurs
CC2: Is set if $(R_{0,8-31})$ is greater than zero
CC3: Is set if $(R_{0,8-31})$ is less than zero
CC4: Is set if $(R_{0,8-31})$ is equal to zero

NONBASE AND BASE REGISTER MODE EXAMPLE

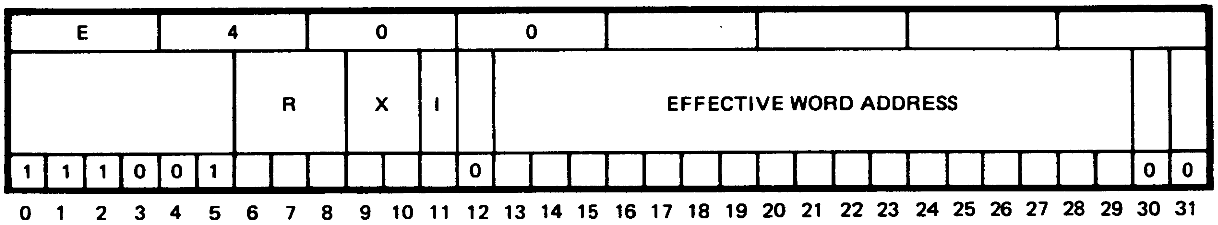
Memory Location:	01000
Hexadecimal Instruction:	3B2E (R_D = 6, R_S = 2)
Assembly Language Coding:	MPRFD 2,6

Before	PSD1	GPR6,7	GPR2,3
	00001000 (Nonbase)	BEF00000	41200000
	02001000 (Base)	00000000	00000000
After	PSD1	GPR6,7	GPR2,3
	10001002 (Nonbase)	BEE00000	41200000
	12001002 (Base)	00000000	00000000

DIVIDE FLOATING-POINT WORD E400 — DVFW d,*m,x



BASE REGISTER FORMAT



830288

NONBASE REGISTER FORMAT

DEFINITION

The divisor is the floating-point word operand found in the effective word location (EWL) specified by the effective word address (EWA). The dividend of the operation is contained in the GPR specified by R in the instruction word. Both of the operands are accessed and the 24-bit signed fraction (bit 0 and bits 8-31) of the divisor is divided into the signed fraction of the dividend. If either one or both of the word operands is negative, the exponent (bits 1-7) of the negative number is changed to its ones complement. Both exponents are then stripped of their 40_{16} bias and the exponent of the divisor is subtracted algebraically from the exponent of the dividend to obtain the initial value of the result exponent; this value may be incremented by one in the ensuing normalization.

The normalized, rounded, quotient is assembled with the exponent (possibly adjusted) difference that has been biased up to 40_{16}. However, if the final fraction is negative, the ones complement of the exponent is used. The assembled sign, exponent, and fraction are transferred to the GPR specified by R.

NOTES

1. If result fraction is zero, both exponent and fraction are set to zero in the GPR specified by R.
2. Operands are expected to be normalized.
3. An arithmetic exception occurs if the divisor is equal to zero. For an arithmetic exception occurrence, GPR R (and R+1 in doubleword operations) retain their original operand value.

SUMMARY EXPRESSION

$(R_{0,8-31})/(EWL_{0,8-31}) \rightarrow R_{0,8-31}$
$(R_{1-7})-(EWL_{1-7}) \rightarrow R_{1-7}$

CONDITION CODE RESULTS

CC1: Is set if arithmetic exception occurs
CC2: Is set if $(R_{0,8-31})$ is greater than zero
CC3: Is set if $(R_{0,8-31})$ is less than zero
CC4: Is set if $(R_{0,8-31})$ is equal to zero

BASE REGISTER MODE EXAMPLE

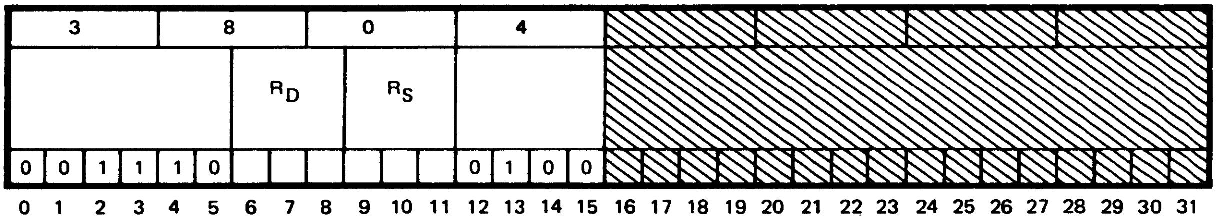
Memory Location:	00E34
Hexadecimal Instruction:	E7460500 (R=6, BR=6, X=4)
Assembly Language Coding:	DVFW 6, X '0500' (6), 4

	PSD1	GPR6	BR6	GPR4	Memory Word 000570
Before	02000E34	41600000	00000070	00000000	41200000
After	22000E38	41300000	00000070	00000000	41200000

NONBASE REGISTER MODE EXAMPLE

Memory Location:	00E34
Hexadecimal Instruction:	E7 00 05 70 (R=6, X=0, I=0)
Assembly Language Coding:	DVFW 6,X'570'

	PSD1	GPR6	Memory Word 570
Before	00000E34	41600000	41200000
After	20000E38	41300000	41200000



830289

DEFINITION

The 24-bit signed fraction (divisor) contained in the general purpose register specified by R_S (bits 0, 8-31) is divided into the fraction (dividend) contained in the GPR specified by R_D. If either one or both of the floating-point numbers are negative, the ones complement of the exponent is taken. Both exponents are then stripped of their 40_{16} bias, and the exponent of the divisor is subtracted algebraically from the exponent of the dividend. The normalized and rounded quotient is placed in bit 0 and bit positions 8-31 of GPR R_D. The result exponent is biased up by 40_{16} and, if the result fraction is negative, the ones complement of the result exponent is placed in bits 1-7 of GPR R_D.

NOTES

1. If result fraction is zero, both exponent and fraction are set to zero in the GPR specified by R.

2 Operands are expected to be normalized.

SUMMARY EXPRESSION

R_D (0,8-31) / R_S (0,8-31) → R_D 0,8-31

R_D(1-7) - R_S (1-7) → R_D 1-7

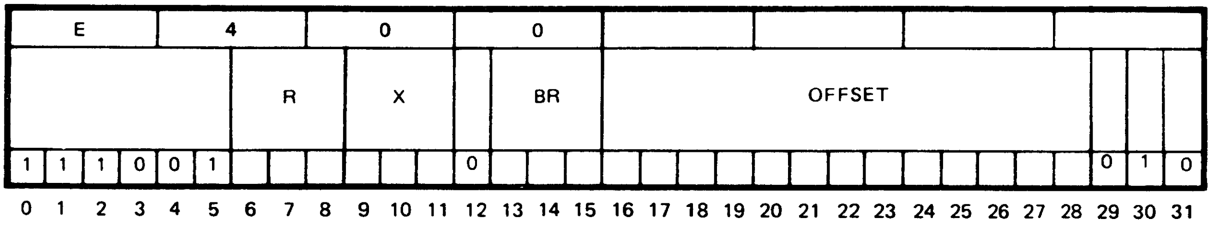
CONDITION CODE RESULTS

CC1: Is set if arithmetic exception occurs
CC2: Is set if ($R_{0, 8-31}$) is greater than zero
CC3: Is set if ($R_{0, 8-31}$) is less than zero
CC4: Is set if ($R_{0, 8-31}$) is equal to zero

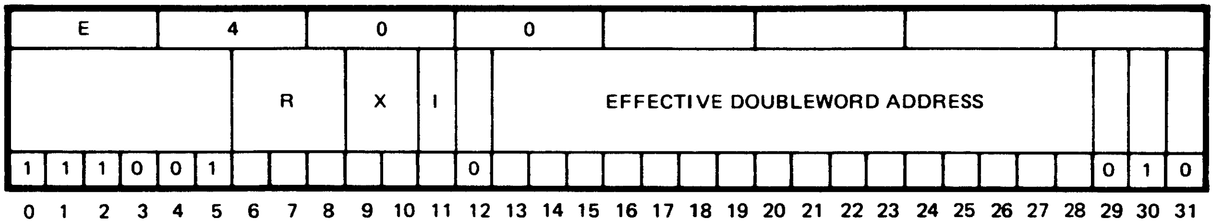
NONBASE AND BASE REGISTER MODE EXAMPLE

Memory Location:	01000
Hexadecimal Instruction:	3B24 (R_D = 6, R_S = 2)
Assembly Language Coding:	DVRFW 2,6

Before	PSD1	GPR6	GPR2
	00001000 (Nonbase) 02001000 (Base)	41600000	41200000
After	PSD1	GPR6	GPR2
	20001002 (Nonbase) 22001002 (Base)	41300000	41200000



BASE REGISTER FORMAT



830290

NONBASE REGISTER FORMAT

DEFINITION

The divisor is the floating-point doubleword operand found in the effective word locations (EWL, EWL+1) specified by the effective doubleword address (EDA). The dividend of the operation is contained in two GPRs referred to as R and R+1. Both of these operands are accessed, and the 56-bit signed fraction (bits 0 and 8-31 of the first memory word and bits 0-31 of the second memory word) is divided into the signed fraction of the dividend. The 56-bit signed fraction of the dividend is made up of bits 0 and 8-31 of the GPR specified by R and bits 0-31 of the GPR specified by R+1. If either one or both of the doubleword operands are negative, the ones complement of the base 16 exponent (bits 1-7) of the negative doubleword is taken. Both exponents are then stripped of their 40_{16} bias and the exponent of the divisor is subtracted algebraically from the exponent of the dividend to obtain the initial value of the result exponent; this value may be incremented by one in the ensuing normalization.

The normalized quotient is assembled with (possibly adjusted) the exponent difference that has been biased up by 40_{16}.

However, if the final fraction is negative, the ones complement of the exponent is used. The assembled sign, exponent, and fraction are transferred to the GPR locations R and R+1.

NOTES

1. If result fraction is zero, both exponent and fraction are set to zero in the GPR specified by R.
2. Operands are expected to be normalized.
3. The GPR specified by R must be an even-numbered register.
4. An arithmetic exception occurs if the divisor is equal to zero. For an arithmetic exception occurrence, GPR R (and R+1 in doubleword operations) retains its original operand value.

SUMMARY EXPRESSION

$$(R_{0,8-31},R+1_{0-31})/(EWL_{0,8-31},EWL+1_{0-31}) \rightarrow R_{0,8-31},R+1_{0-31}$$
$$(R_{1-7})-(EWL_{1-7}) \rightarrow R_{1-7}$$

CONDITION CODE RESULTS

CC1: Is set if arithmetic exception occurs
CC2: Is set if $(R_{0,8-31}, R+1_{0-31})$ is greater than zero
CC3: Is set if $(R_{0,8-31}, R+1_{0-31})$ is less than zero
CC4: Is set if $(R_{0,8-31}, R+1_{0-31})$ is equal to zero

BASE REGISTER MODE EXAMPLE

Memory Location:	0175C
Hexadecimal Instruction:	E7060502 (R=6, X=0, BR=6)
Assembly Language Coding:	DVFD 6,X'500' (6)

Before	PSD1	GPR6,7	BR6	Memory Doubleword 000570, 000574
	0200175C	40606060	00000070	40303030
		60606060		30303030
After	PSD1	GPR6,7	BR6	Memory Doubleword 000570, 000574
	22001760	41200000	00000070	40303030
		00000000		30303030

NONBASE REGISTER MODE EXAMPLE

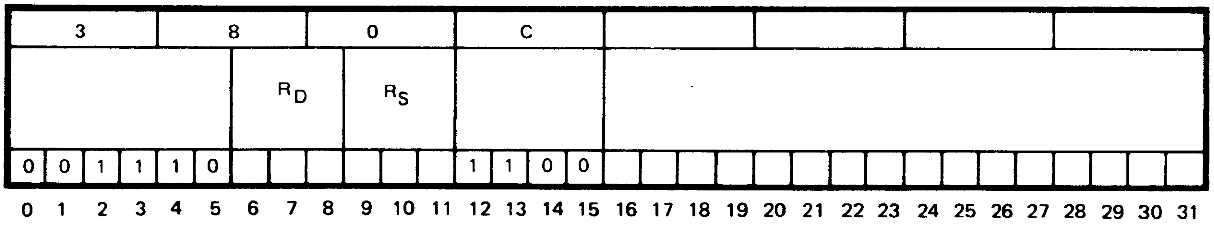
Memory Location:	0175C
Hexadecimal Instruction:	E7 00 05 72 (R=6, X=0, I=0)
Assembly Language Coding:	DVFD 6,X'572'

Before	PSD1	GPR6,7	Memory Doubleword 570,574
	0000175C	40606060	40303030
		60606060	30303030
After	PSD1	GPR6,7	Memory Doubleword 570,574
	20001760	41200000	40303030
		00000000	30303030

DIVIDE FLOATING-POINT DOUBLEWORD REGISTER TO REGISTER

380C

DVRFD
s,d



DEFINITION

830291

The 56-bit signed fraction (divisor) contained in the general purpose registers specified by R_S (bits 0, 8-31) and R_S+1 (bits 0-31) is divided into the 56 bit signed fraction (dividend) contained in the GPRs specified by R_D and R_D+1. If either one or both of the floating-point numbers are negative, the ones complement of the exponent is taken. Both exponents are then stripped of their 40_{16} bias, and the exponent of the divisor is subtracted algebraically from the exponent of the dividend to obtain the initial value of the result exponent; this value may be incremented by one in the ensuing normalization. The normalized quotient is placed in bit 0 and bit positions 8-31 of GPR R_D and 0-31 of GPR R_D+1. The result exponent is biased up by 40_{16} and, if the result fraction is negative, the ones complement of the result exponent is placed in bits 1-7 of GPR R_D.

NOTES

1. If result fraction is zero, both exponent and fraction are set to zero in the GPR specified by R.
2. Operands are expected to be normalized.
3. The GPRs specified by R_D and R_S must be even-numbered registers.

SUMMARY EXPRESSION

$$(R_{D\ 0,8-31}),(R_{D+1\ 0-31})\ /\ (R_{S0,8-31})(R_{S+1\ 0-31}) \rightarrow R_{D\ 0,8-31},R_{D+1\ 0-31}$$

$$(R_{D1-7}) - (R_{S\ 1-7}) \rightarrow R_{D\ 1-7}$$

CONDITION CODE RESULTS

CC1: Is set if arithmetic exception occurs
CC2: Is set if $(R_{0,\ 8-31})$ is greater than zero
CC3: Is set if $(R_{0,\ 8-31})$ is less than zero
CC4: Is set if $(R_{0,\ 8-31})$ is equal to zero

NONBASE AND BASE REGISTER MODE EXAMPLE

Memory Location:	01000
Hexadecimal Instruction:	3B2C (R_D = 6, R_S = 2)
Assembly Language Coding:	DVRFD 2,6

Before	PSD1	GPR6,7	GPR2,3
	00001000 (Nonbase)	40606060	40303030
	02001000 (Base)	60606060	30303030
After	PSD1	GPR6,7	GPR2,3
	20001002 (Nonbase)	41200000	40303030
	22001002 (Base)	00000000	30303030

6.2.11 Floating-Point Conversion Instructions

The floating-point conversion instructions provide the capability to convert floating-point form to fixed-point form and vice-versa.

6.2.11.1 Instructions Format

The floating-point conversion instructions use the interregister formats.

6.2.11.2 Condition Code

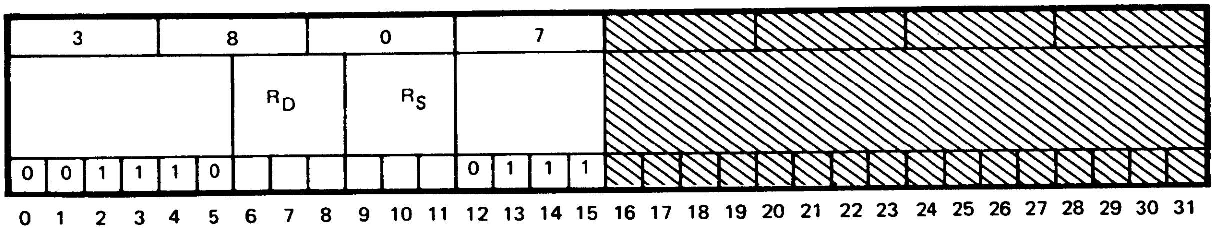
When a floating-point conversion instruction causes an arithmetic exception condition (FIXW and FIXD only), CC1 is set to indicate that an arithmetic exception has occurred.

The condition codes CC2, CC3, and CC4 are set to indicate whether the result of the operation was greater than, less than or equal to zero.

Refer to page 6-262 for a description of fixed-point (integer) operand formats, and 6-324 for a description of proper floating-point operand formats.

FLOAT INTEGER WORD TO FLOATING-POINT WORD 3807

FLTW s,d



830292

DEFINITION

The signed integer word contained in the general purpose register (GPR) specified by R_S is converted to floating-point format by creating a signed fraction normalized for an assumed radix point between bit positions 7 and 8. A 7-bit base 16 exponent is calculated by assigning it an initial value of 6_{16} and subtracting from it a value equal to the number of hexadecimal left shifts required to correctly normalize the operand. If the operand requires right shifting to create a correctly normalized fraction (non-sign bits to the left of bit 8), a value equal to the number of the hexadecimal right shifts will be added to the initial exponent value of 6_{16}. The normalized fraction is then truncated to 24 bits of significance and the signed fraction is placed in bit positions 0 and 8 through 31 of the GPR specified by R_D. The result exponent is biased up by 40_{16} and, if the result fraction is negative, is replaced by its ones complement then placed in bit positions 1 through 7 of the GPR specified by R_D.

NOTE

> If the result fraction equals zero, the exponent and fraction are set to zero in the GPR specified by R_D.

SUMMARY EXPRESSION

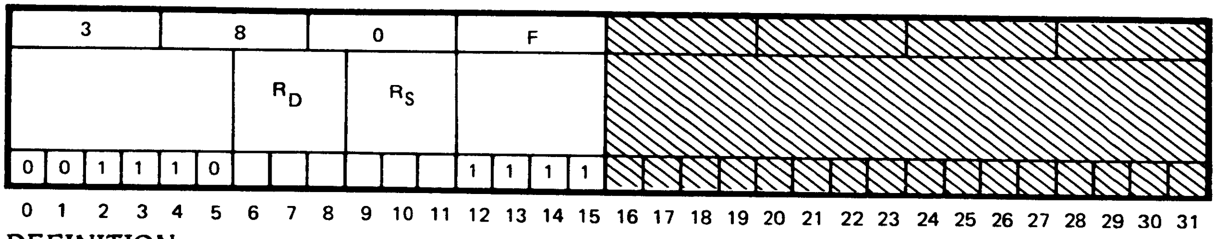
FLT $(R_S) \rightarrow R_D$

CONDITION CODE RESULTS

CC1: Always zero
CC2: $(R_D$ 0, 8-31) is greater than zero
CC3: $(R_D$ 0, 8-31) is less than zero
CC4: $(R_D$ 0, 8-31) is equal to zero

NONBASE AND BASE REGISTER MODE EXAMPLE

Memory Location:		1000	
Hexadecimal Instruction:		3BB7 (R_D=7,R_S=3)	
Assembly Language Coding:		FLTW 3,7	
Before	PSD1 00001000 (Nonbase) 02001000 (Base)	GPR3 04000000	GPR7 06300000
After	PSD1 20001002 (Nonbase) 22001002 (Base)	GPR3 04000000	GPR7 47400000



830293

DEFINITION

The signed integer doubleword contained in the general purpose register (GPRs) specified by R_S and R_S +1 is converted to floating-point format by creating a signed fraction normalized for an assumed radix point between bit positions 7 and 8. A 7-bit base 16 exponent is calculated by assigning it an initial value of $0E_{16}$ and subtracting from it a value equal to the number of hexadecimal left shifts required to correctly normalize the operand. If the integer requires right-shifting to create a correctly normalized fraction (non-sign bits to the left of bit position 8), a value equal to the number of hexadecimal right shifts will be added to the initial exponent value of $0E_{16}$. The normalized fraction is then truncated to 56 bits of significance and the signed fraction is placed in bit positions 0 and 8 through 31 of the GPR specified by R_D and bit positions 0 through 31 of the GPR specified by R_D +1. The result exponent is biased up by 40_{16} and, if the resultant fraction is a negative, is replaced by its ones complement then placed in bit positions 1 through 7 of the GPR specified by R_D.

NOTES

1. If the result fraction equals zero, the exponent and fraction are set to zero in the GPR specified by R_D and R_D +1.
2. The GPRs specified by R_S and R_D must both be even-numbered , registers.

SUMMARY EXPRESSION

FLT R_S, R_S +1 → R_D, R_D + 1

CONDITION CODE RESULTS

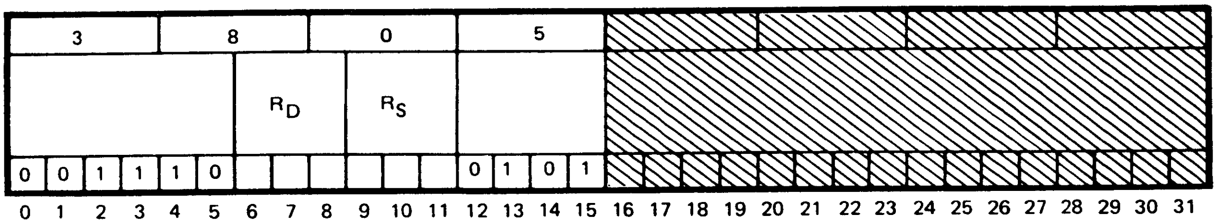
CC1: Always zero
CC2: $(R_{D\ 0,8-31}, R_{D\ +\ 1\ 0-31})$ is greater than zero
CC3: $(R_{D\ 0,8-31}, R_{D\ +\ 1\ 0-31})$ is less than zero
CC4: $(R_{D\ 0,8-31}, R_{D\ +\ 1\ 0-31})$ is equal to zero

NONBASE AND BASE REGISTER MODE EXAMPLE

Memory Location:	1000
Hexadecimal Instruction:	3B2F (R_D = 6, R_S = 2)
Assembly Language Coding:	FLTD 2,6

Before	PSD1	GPR2,3	GPR6,7
	00001000 (Nonbase)	04000000	06300000
	02001002 (Base)	20000000	10000000
After	PSD1	GPR2,3	GPR6,7
	20001002 (Nonbase)	04000000	4F400000
	22001002 (Base)	20000000	02000000

FIX FLOATING-POINT WORD TO INTEGER WORD **FIXW**
3805 **s,d**



830294

DEFINITION

The exponent (bits 1-7) of the floating-point word in the GPR specified by R_S is stripped of it's 40_{16} bias and replaced by its ones complement if the fraction sign (bit 0) is negative. If the resultant exponent is greater than 8, or else is equal to 8 and the most significant fraction bit (bit 8) is not a sign bit, an arithmetic exception condition is generated. If the resultant exponent is less than or equal to zero, or the contents of R_S equals zero, a result of zero is placed in bit positions 0 to 31 of the GPR specified by R_D. Otherwise, the floating-point fraction is sign extended in bit positions 0-7, and converted to integer format by right shifting the assumed radix point a number of hexadecimal positions equal to the value of the exponent. This integer number is then arithmetically right or left shifted, as necessary, to place the new radix position to the right of bit 31 (truncating any bits to the right of the radix point). The resultant 32 bit signed integer is placed in bit positions 0 through 31 of the GPR specified by R_D.

NOTE

1. The operand is expected to be normalized.
2. If an arithmetic exception occurs, the contents of the GPR specified by R_D are unchanged.

SUMMARY OF EXPRESSIONS

FIX (R_S) → R_D

CONDITION CODE RESULTS

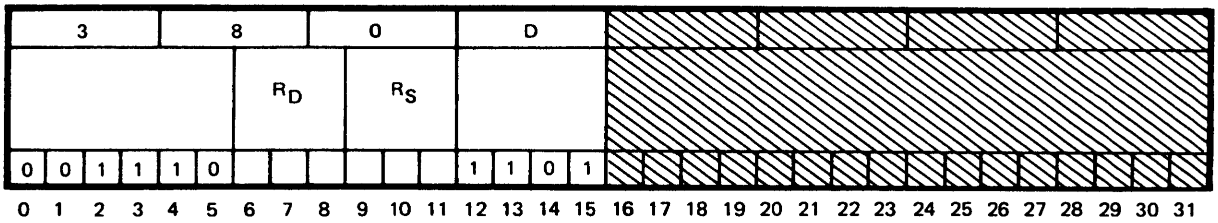
CC1: Is set if arithmetic exception occurs
CC2: Is set if (R_D) is greater than zero
CC3: Is set if (R_D) is less than zero
CC4: Is set if (R_D) is equal to zero

NONBASE AND BASE REGISTER MODE EXAMPLE

The floating point value in GPR3 is converted to an integer by shifting it one hexadecimal position to the left. The result is placed in GPR7. CC2 is set.

Memory Location:	1000
Hexadecimal Instruction:	3BB5 (R_D = 7, R_S = 3)
Assembly Language Coding:	FIXW 3,7

Before	PSD1	GPR3	GPR7
	00001000 (Nonbase)	47400000	F3803000
	02001000 (Base)		
After	PSD1	GPR3	GPR7
	2001002 (Nonbase)	47400000	04000000
	2201002 (Base)		



830295

DEFINITION

The exponent (bits 1-7) of the floating-point doubleword in the GPR-pair specified by R_S and R_S+1 is stripped of it's 40_{16} bias and replaced by its ones complement if the fraction sign (bit 0) is negative. If the resultant exponent is greater than 10_{16}, or else is equal to 10_{16} and the most significant fraction bit (bit 8) is not a sign bit, an arithmetic exception condition is generated. If the resultant exponent is less than or equal to zero, or the contents of the register-pair R_S and R_S+1 equals zero, a result of zero is placed in the register-pair R_D and R_D+1. Otherwise, the floating-point fraction is sign-extended in bit positions 0-7, and converted to integer format by right shifting the assumed radix point a number of hexadecimal positions equal to the value of the exponent. This integer number is then shifted right or left arithmetically as necessary, to place the new radix position to the right of bit position 31 of R_S+1 (truncating any bits to the right of the radix point). The resultant 64 bit signed integer is placed in bit positions 0 through 31 of the GPR specified by R_D and in bit positions 0 through 31 of the GPR specified by R_D+1.

NOTES

1. The GPRs specified by R_S and R_D must both be even-numbered registers

2. The operand is expected to be normalized.

3. If an arithmetic exception occurs, the contents of the GPR pair specified by R_D are unchanged.

SUMMARY EXPRESSION

$\text{FIX}(R_S, R_S + 1) \rightarrow R_D, R_D + 1$

CONDITION CODE RESULTS

CC1: Is set if arithmetic exception occurs
CC2: Is set if (R_D, R_D+1) is greater than zero
CC3: Is set if (R_D, R_D+1) is less than zero
CC4: Is set if (R_D, R_D+1) is equal to zero

NONBASE AND BASE REGISTER MODE EXAMPLE

Memory Location:	1000
Hexadecimal Instruction:	3B2D (R_D = 6, R_S = 2)
Assembly Language Coding:	FIXD 2,6

Before	PSD1	GPR6,7	GPR2,3
	00001000 (Nonbase)	F3803000	41100000
	02001000 (Base)	20000000	00000000
After	PSD1	GPR6,7	GPR2,3
	20001002 (Nonbase)	00000000	41100000
	22001002 (Base)	00000001	00000000

This page intentionally left blank

6.2.12 Control Instructions

This group of instructions allows the mainframe to perform execute, no operation, halt, and wait operations.

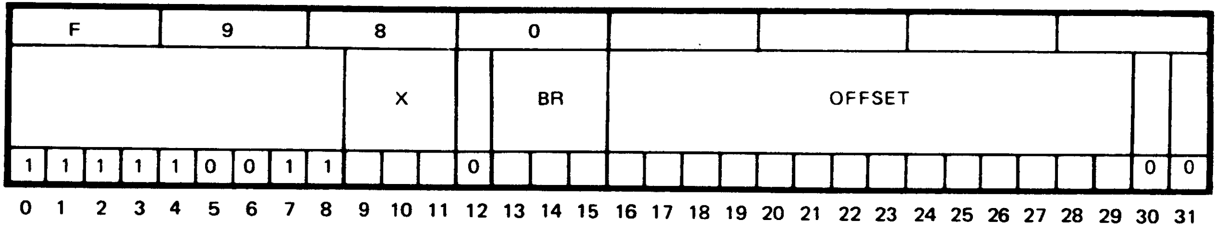
6.2.12.1 Instruction Format

Control instructions use the memory reference and interregister instruction formats. Several of the control instructions vary the basic interregister format in that certain portions are not used and are left zero.

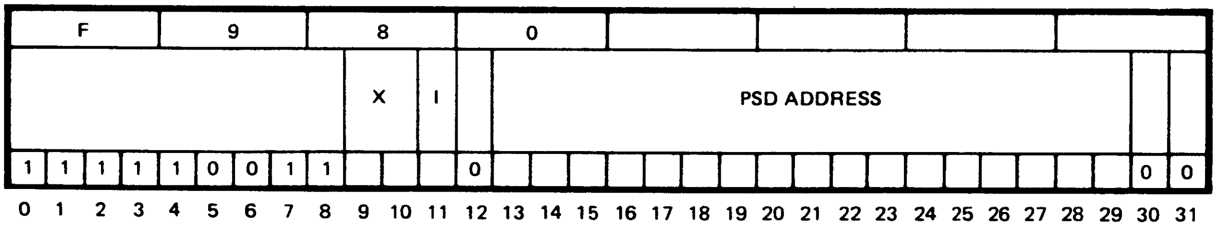
6.2.12.2 Condition Code

Condition code results for execute operations will be dependent on the instruction that was performed. All other control operations leave the current condition code unchanged.

LOAD PROGRAM STATUS DOUBLEWORD F980 — LPSD d,*m,x



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

830296

DEFINITION

Causes the contents of two successive memory words addressed by the instruction to be loaded into bits 0-31 and 32-49 of the program status doubleword.

SUMMARY EXPRESSION

$(EWL) \rightarrow PSD1_{0-31}$

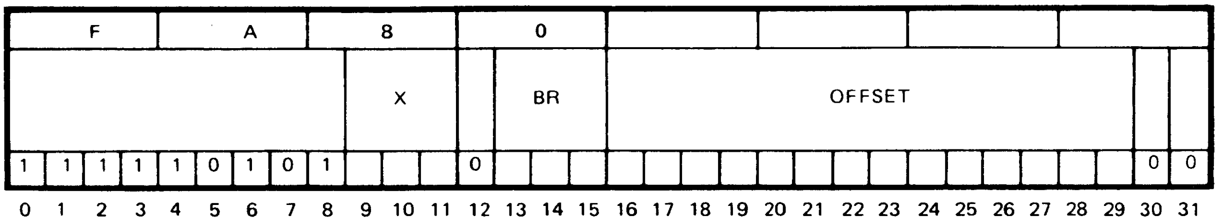
$(EWL+1) \rightarrow PSD2_{32-49}$

CONDITION CODE RESULTS

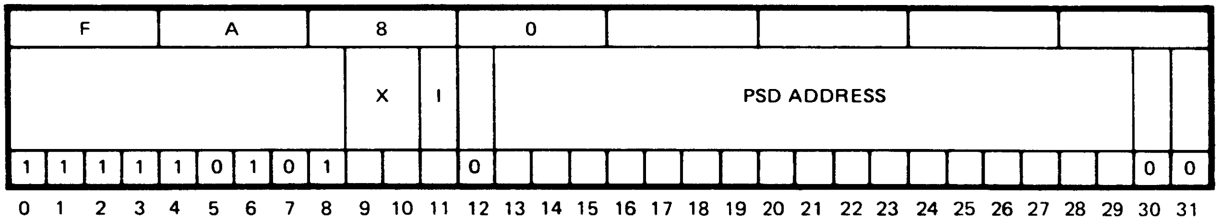
CC1: Changed by the PSD being loaded
CC2: Changed by the PSD being loaded
CC3: Changed by the PSD being loaded
CC4: Changed by the PSD being loaded

NOTES

1. LPSD is a privileged instruction.
2. The LPSD instruction causes the system to enter the mapped or unmapped mode in accordance with bit 32 in the new PSD that is being loaded.
3. This instruction does not modify the contents of the CPIX field or the contents of the map registers.
4. The block external interrupts will be changed in accordance with bits 48 and 49 of the PSD.
5. This instruction will enable the power fail trap or console attention trap if it is disabled.
6. The operand (PSD) of this instruction must be on a word boundary.



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

830297

DEFINITION

Causes the contents of two successive memory words, addressed by the instruction, to be loaded into the PSD words, and the map to be loaded in accordance with note 3.

SUMMARY EXPRESSION

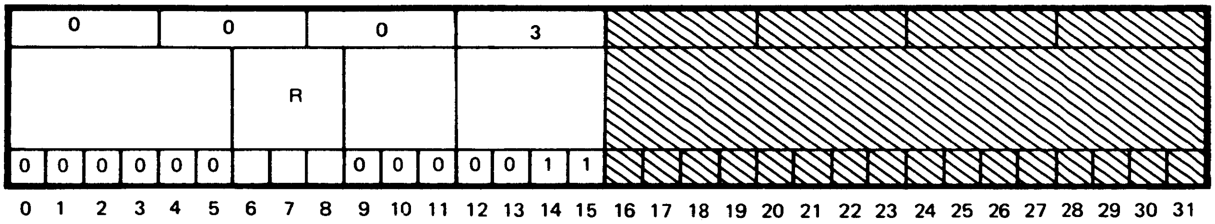
(EWL) → $PSD1_{0-31}$
(EWL+1) → $PDS2_{32-49}$
(MIDL) → Map Registers

CONDITION CODE RESULTS

CC1: Changed by the PSD being loaded
CC2: Changed by the PSD being loaded
CC3: Changed by the PSD being loaded
CC4: Changed by the PSD being loaded

NOTES

1. LPSDCM is a privileged instruction.
2. The LPSDCM instruction causes the system to enter the mapped or unmapped mode as defined by the contents of bit 32 in the new PSD that is being loaded.
3. The CPIX field (PSD_{50-61}) and the map registers are loaded only if map is enabled ($EWL+1_0=1$) and the retain current map bit is reset ($EWL+1_{15}=0$).
4. The block external interrupts mode is established by bits 48 and 49 of the new PSD.
5. This instruction will enable the power fail trap or console attention if it is disabled.
6. The operand (PSD) of this instruction must be on a word boundary.



830298

DEFINITION

The contents of control switches memory location 780, bits 0-31 are transferred to bit positions 0-31 of the general purpose register (GPR) specified by R.

SUMMARY EXPRESSION

$(CS_{0-31}) \rightarrow R_{0-31}$

CONDITION CODE RESULTS

CC1: Always zero
CC2: Is set if (R_{0-31}) is greater than zero
CC3: Is set if (R_{0-31}) is less than zero
CC4: Is set if (R_{0-31}) is equal to zero

NONBASE AND BASE REGISTER MODE EXAMPLE

The contents of the control switches, memory location 780, is transferred to GPR7. CC3 is set.

Memory Location:	06002
Hexadecimal Instruction:	0383 (R=7)
Assembly Language Coding:	LCS 7

Before	PSD1 00006002 (Non Base) 02006002 (Base)	GPR7 FFFFFFFF
	Control Switches (Memory location 780) 82000000	
After	PSD1 10006005 (Non Base) 12006005 (Base)	GPR7 82000000

EXECUTE REGISTER
C807

EXR
s

C				8				0				7																			
						R										UNASSIGNED															
1	1	0	0	1	0				0	0	0	0	1	1	1															0	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

830299

DEFINITION

The word in the general purpose register (GPR) specified by R is transferred to the instruction register to be executed as the next instruction. If this instruction is not a branch, the next instruction executed (following execution of the instruction in register R) is in the sequential memory location following the EXR instruction. If the GPR specified by R does contain a branch instruction, the program status doubleword (PSD) is changed accordingly.

NOTES

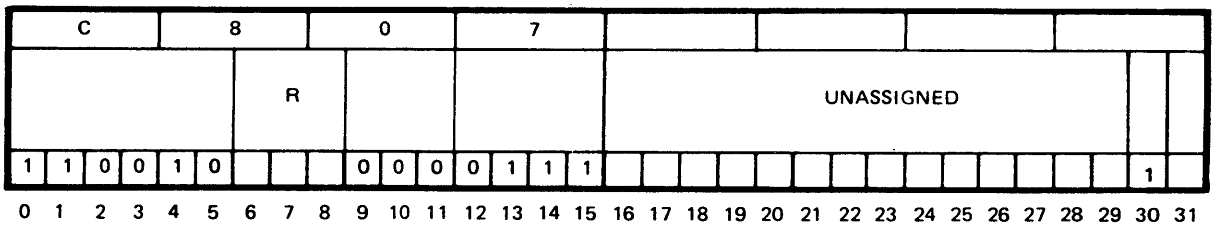
1. If two halfword instructions are in the GPR specified by R, only the left halfword instruction is executed.
2. An undefined instruction trap is generated if an EXR instruction attempts to execute an undefined instruction.

SUMMARY EXPRESSION

(R) → Instruction register

CONDITION CODE RESULTS

Defined by the executed instruction.



830300

DEFINITION

The contents of the least-significant halfword (bits 16-31) of the general purpose register (GPR) specified by R are transferred to the most-significant halfword position (bits 0-15) of the instruction register to be executed as the next instruction. The next instruction executed (following execution of the halfword instruction transferred to the instruction register) is in the sequential memory location following the EXRR instruction.

NOTE

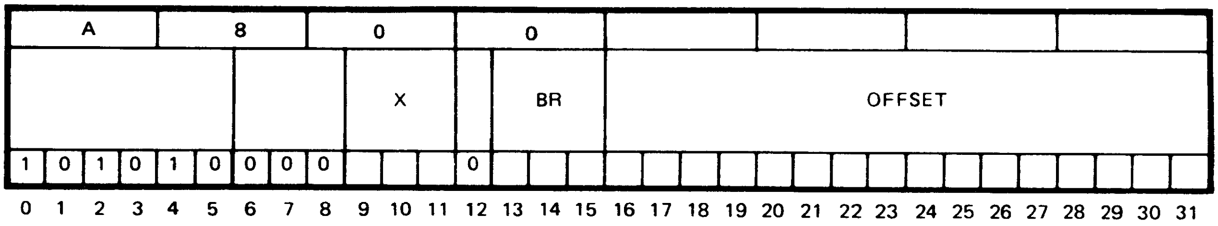
An undefined instruction trap is generated if an EXRR instruction attempts to execute an undefined instruction.

SUMMARY EXPRESSION

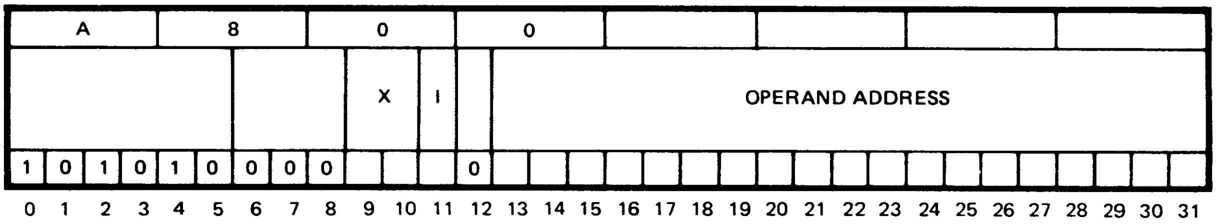
$(R_{16-31}) \rightarrow$ Instruction Register $_{0-15}$

CONDITION CODE RESULTS

Defined by the executed instruction.



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

830301

DEFINITION

The contents of the effective word location (EWL) specified by the effective address (EWA) is accessed and executed as the next instruction. If this instruction is not a branch, the next instruction executed (following execution of the instruction specified by the EWA) is in the next sequential memory location following the EXM instruction. If the instruction in memory specified by the EWA is a branch instruction, the program status doubleword (PSD) is changed accordingly.

NOTES

1. If two halfword instructions are in the memory location specified by the EWA, bit 30 of the EWA determines which halfword instruction is executed. When bit 30 equals zero, the left halfword is executed. When bit 30 equals one, the right halfword is executed.

2. An undefined instruction trap is generated if an EXM instruction attempts to execute an undefined instruction or another Execute instruction.

SUMMARY EXPRESSION

If $EA_{30}=0$

$(EWL_{0-15}) \rightarrow IW$

If $EA_{30}=1$

$(EWL_{16-31}) \rightarrow IW$

CONDITION CODE RESULTS

Defined by the executed instruction.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0				0				0				0				▨				▨				▨				▨			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨

830302

DEFINITION

The execution of this instruction causes computer operation to stop and lights the halt indicator on the system control panel. This instruction terminates input/output transfers and the servicing of priority interrupts. I/O in progress will be completed, but no interrupts will be serviced. Leaving a HALT condition requires depressing the RUN/HALT switch on the system control panel or execute a RUN command at the IOP console.

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

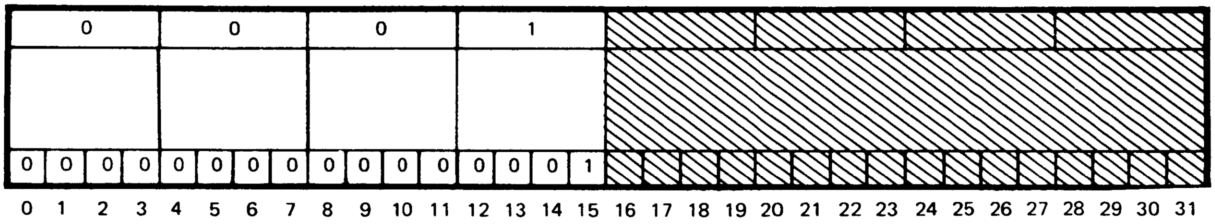
NOTES

1. HALT is a privileged instruction.

2. If the CPU halt trap is enabled, and the privileged bit is set, the execution of the halt instruction will cause a halt trap.

3. In the IPU the halted state may be terminated when the IPU receives a signal IPU (SIPU) trap from the CPU.

4. In the CPU and IPU, the receipt of a power down trap or IOP run command will cause the halt state to be terminated.



830303

DEFINITION

The execution of this instruction causes the central processing unit (CPU) to enter the idle mode and lights the wait indicator on the system control panel. Input/output or trap transfers and priority interrupt servicing continue. If an interrupt occurs during a wait condition, a return to the wait may occur after the interrupt is serviced.

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

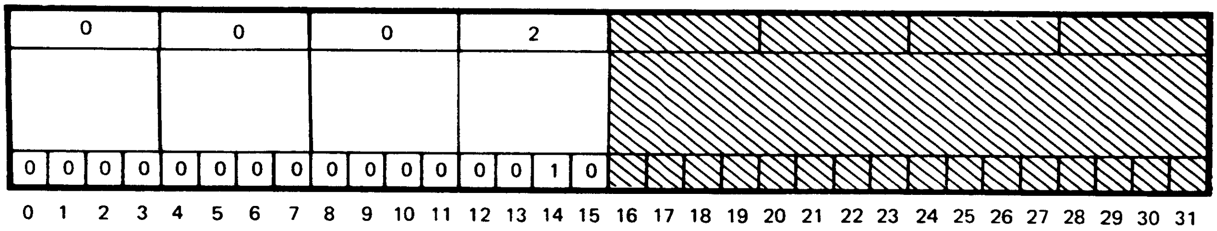
NOTES

1. If there is an attempt to execute a WAIT with interrupts blocked, the system check trap will be generated.

2. If the WAIT instruction is the first instruction of a trap or interrupt software handler or the first instruction following a DAI or DACI a system check trap will occur. These sequences are defined as uninterruptable and therefore the WAIT state cannot be terminated.

3. If a trap or interrupt occurs while the CPU is in a WAIT state the old PSD of the TCB or ICB points to the starting address of the WAIT instruction.

NO OPERATION
0002



830304

DEFINITION

The assembler generates the no operation instruction following a halfword instruction in order to force the next instruction to start on a word boundary, if the next instruction is a word instruction. The NOP instruction is also used whenever there is a need for an executable instruction that does not alter the machine status.

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

NOTES

1. In the CPU/IPU, if the NOP instructions are encountered in the right halfword, during sequential instruction execution the right halfword NOP is skipped and the halfword pair is treated as a single fullword. The execution time of this type of halfword pair is the execution time of the left halfword.

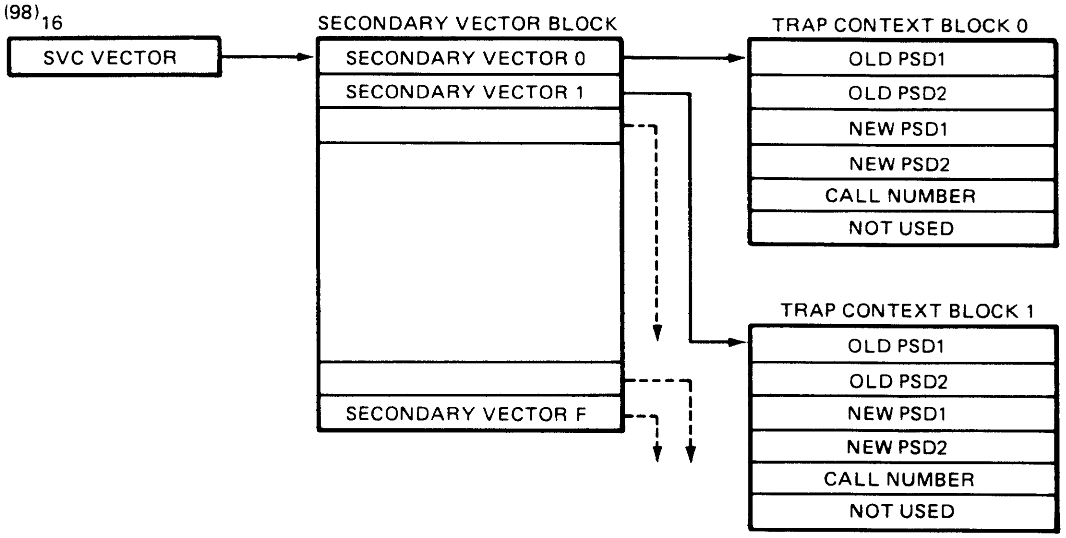
2. Since right halfword NOPs are skipped, if the left halfword instruction causes a trap or an interrupt occurs following the execution of the left halfword, the old PSD stored in the trap or interrupt TCB/ICB points to the next full word instruction unless the left halfword was a WAIT instruction.

C	8	0	6				
				INDEX	CALL NUMBER		
1 1 0 0	1 0 0 0	0 0 0 0	0 1 1 0				
0 1 2 3	4 5 6 7	8 9 10 11	12 13 14 15	16 17 18 19	20 21 22 23	24 25 26 27	28 29 30 31

DEFINITION

The execution of the SVC instruction causes a trap to the trap vector location for relative priority level 6. Bits 16-19 are used to index the initial interrupt vector to one of 16 locations. The secondary vector address contained in the indexed location points to a SVC vector block the content of which should point to a trap subroutine (new PSD in words 2 and 3 of the vector block).

The contents of bits 20-31 are referred to as the call number. This call number serves as an identifier parameter for software use.



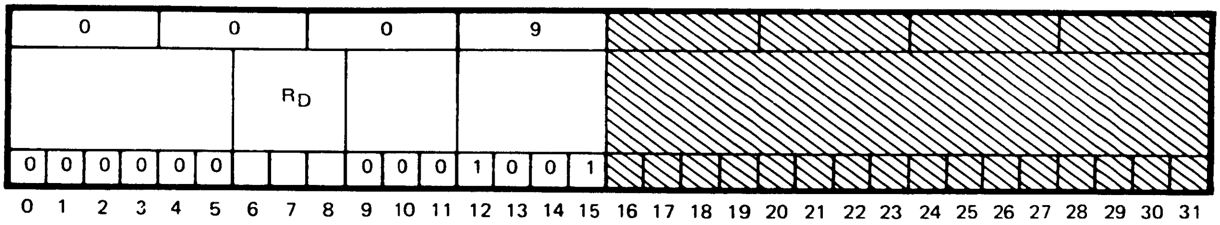
830305

CONDITION CODE RESULTS

CC1: Zero
CC2: Zero
CC3: Zero
CC4: Zero

As specified by the new PSD of the SVC TCB.

Condition code settings upon return to the next succeeding instruction depend upon action taken within the trap routine.



830307

DEFINITION

This instruction places the current operational status of the central processing unit (CPU) into register R_D. The CPU status in register R_D is defined as follows:

Bit 0	=0	Unprivileged mode
	=1	Privileged mode
Bits 1-4		(Always zeros)
Bit 5	=0	Extended addressing disabled
	=1	Extended addressing enabled
Bit 6	=0	Base register mode disabled
	=1	Base register mode enabled
Bit 7	=0	Arithmetic exception trap disabled
	=1	Arithmetic exception trap enabled
Bit 8	=0	Map disabled
	=1	Map enabled
Bits 9-19		(Always zero)
Bit 20	=0	Write to Writable Control Store/Alterable Control Store is disables (V6 ONLY)
	=1	Write to Writable Control Store/Alterable Control Store is enables (V6 ONLY)
Bit 21	=0	PROM mode enabled (V6 ONLY)
	=1	Alterable Control Store mode enabled (V6 ONLY)
Bit 22	=0	Floating Point Accelerator present and enabled (V6 ONLY)
	=1	Floating Point Accelerator disabled or not present (V6 ONLY)
Bit 23	=0	Privileged mode halt trap disabled
	=1	Privileged mode halt trap enabled
Bit 24	=0	Interrupts are unblocked
	=1	Interrupts are blocked
Bit 25	=0	Software traps are disabled (automatic trap halt is enabled)
Bit 26		(Always zero)
Bit 27	=0	Processor = CPU
	=1	Processor = IPU

Bits 28-31		CPU/IPU Model Indicator
	=0	32/55 CPU
	=1	32/75 CPU
	=2	32/27 CPU
	=3	32/67 CPU
	=4	32/87 CPU
	=5	32/97 CPU
	=6	V6 CPU
	=7	V9 CPU
	=8-F	Not defined

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2				C				0				9				▨				▨				▨				▨			
						R										▨															
0	0	1	0	1	1				0	0	0	1	0	0	1	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨

DEFINITION

The execution of this instruction causes certain operating characteristics of the central processing unit (CPU) to change as specified by the contents of R.

The contents of R are defined as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RESERVED										RESERVED										WWCS	ACS	HFPA	PRIV		TRAP	RESERVED					
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0		0	0	0	0	0	0

830306

Bits 0-19		Not used (reserved)
Bit 20	=0	Write to Writable Control Store/Alterable Control Store is disabled (V6 only)
	=1	Write to Writable Control Store/Alterable Control Store is enabled (V6 only)
Bit 21	=0	Enable PROM Mode (V6 only)
	=1	Enable Alterable Control Store Mode (V6 only)
Bit 22	=0	Enable High Speed Floating Point Accelerator (V6 only)
	=1	Disable High Speed Floating Point Accelerator (V6 only)
Bit 23	=0	Disable privileged mode halt trap
	=1	Enable privileged mode halt trap
Bit 24		Not used (reserved)
Bit 25	=0	Disable software trap handling (enable automatic trap halt)
	=1	Enable software trap handling
Bits 26-31		Not used (reserved)

CONDITION CODE RESULTS

CC1:	No change
CC2:	No change
CC3:	No change
CC4:	No change

NOTES

1. SETCPU is a privileged instruction.
2. The SETCPU should always be preceded by a RDSTS instruction and then either a ZBR or SBR to enable/disable the desired function.
3. Bits 20, 21, and 22 are for V6 CPU only.

ENABLE ARITHMETIC EXCEPTION TRAP
0008

0				0				0				8				▨				▨				▨				▨			
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

830308

DEFINITION

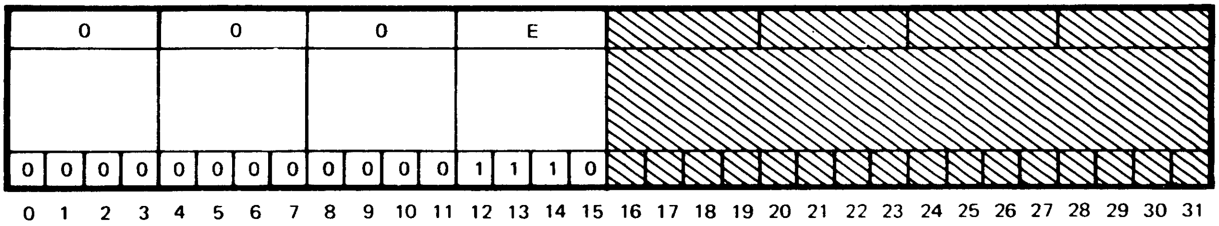
This instruction sets bit 7 of the program status doubleword (PSD) to enable the arithmetic exception trap.

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

DISABLE ARITHMETIC EXCEPTION TRAP 000E

DAE



830309

DEFINITION

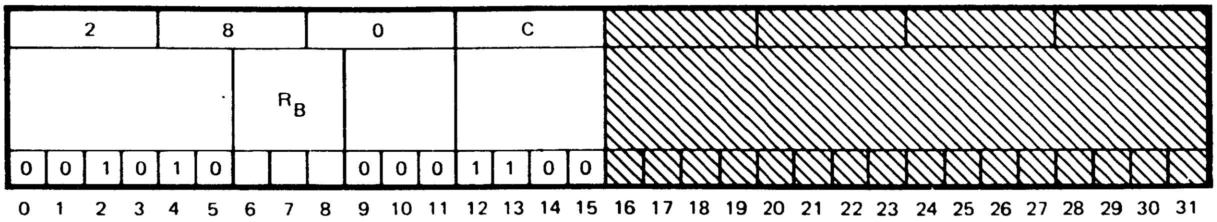
This instruction resets bit 7 of the program status doubleword (PSD) to disable the arithmetic exception trap.

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

TRANSFER PROGRAM COUNTER TO BASE REGISTER 280C

TPCBR d



830310

DEFINITION

This instruction transfers the contents of the program counter (PC) (bits 8-30 of program status doubleword) to bit position 8-30 of the base register specified by R_B. Bit positions 0-7 of specified register are set to zero.

SUMMARY OF EXPRESSION

$$(PSD_{8-30}) \rightarrow R_B8\text{-}30$$

$$\text{Zeros} \rightarrow R_B0\text{-}7$$

CONDITION CODE RESULTS

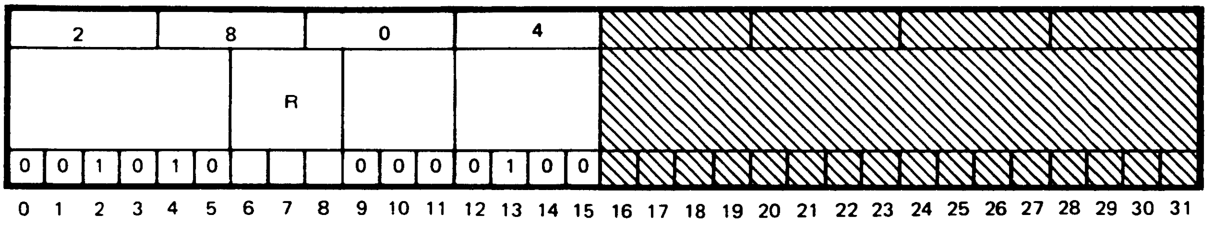
Condition codes remain unchanged.

NOTE

This instruction is used for the base register mode only.

BASE REGISTER MODE EXAMPLE

Memory Location:		1000
Hexadecimal Instruction:		2A0C (R_B=4)
Assembly Language Coding:		TPCBR 4
Before	PSD1	R_B4
	02001000	2468ABC1
After	PSD1	R_B4
	02001002	00001000



830312

DEFINTION

Transfer the contents of Condition Code bits CC1, CC2, CC3, and CC4 into bit positions 28-31 of the GPR specified by R.

SUMMARY OF EXPRESSIONS

$(PSW_{1-4}) \rightarrow R_{28-31}$

Zeros $\rightarrow$ R_{0-27}

CONDITION CODES

CC1: Unchanged
CC2: Unchanged
CC3: Unchanged
CC4: Unchanged

NOTE

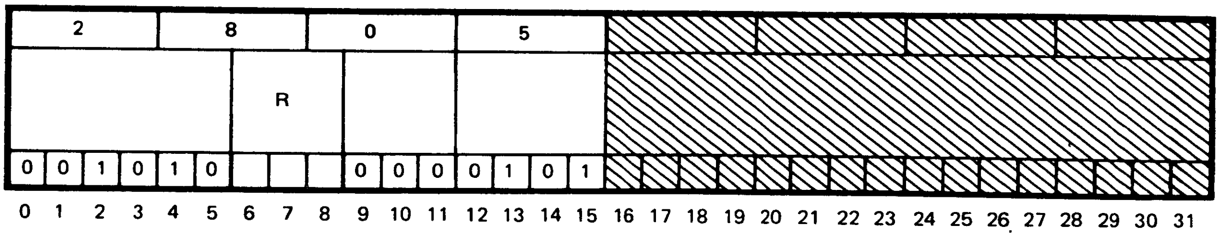
This instruction is valid in the base register mode only.

BASE REGISTER MODE EXAMPLE

Memory Location:		1000
Hexadecimal Instruction:		2A04 (R=4)
Assembly Language Coding:		TCCR 4
Before	PSD1 22001000	GPR4 00003456
After	PSD1 22001002	GPR4 00000004

TRANSFER GPR TO CONDITION CODES 2805

TRCC



DEFINITION

830313

Transfers the contents of bit positions 28-31 of the GPR specified by R to the Condition Code field (bits 1-4) of the Program Status Doubleword (PSD).

SUMMARY OF EXPRESSIONS

$(R_{28-31}) \rightarrow PSW_{1-4}$

CONDITION CODE RESULTS

CC1: Bit 28 of R
CC2: Bit 29 of R
CC3: Bit 30 of R
CC4: Bit 31 of R

NOTE

This instruction is used for the base register mode only.

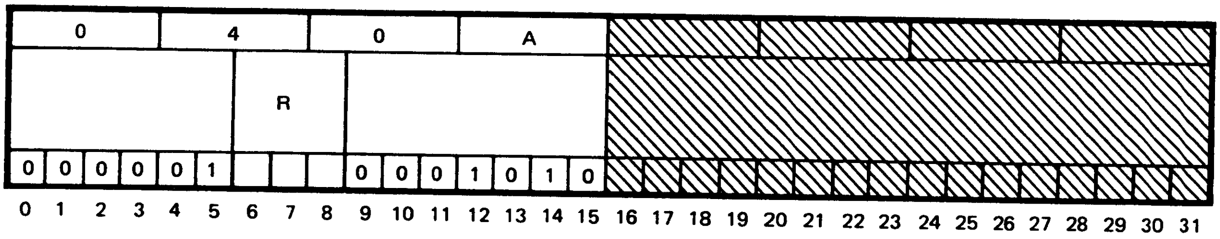
BASE REGISTER MODE EXAMPLE

Memory Location:	1000
Hexadecimal Instruction:	2A05 (R=4)
Assembly Language Coding:	TRCC 4

Before	PSD1 02001000	GPR4 00000004
After	PSD1 22001002	GPR4 00000004

CACHE MEMORY CONTROL (V6 ONLY)
040A

CMC
d



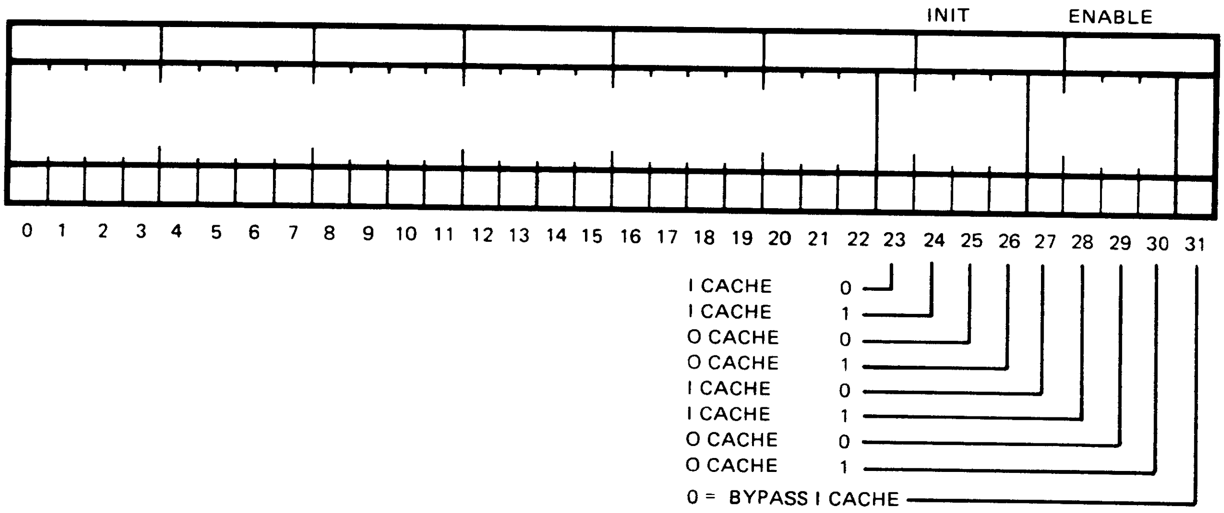
830315

DEFINITION

Bits labeled 'Enable' of the word in the general purpose register (GPR) specified by R are loaded into the Cache Memory Control Register (CMCR) of the CPU. Bits labeled 'Init' are used to control selective initialization of the corresponding cache memory bank using the patterns shown in note 4. A '1' in a bit position enables the operation. The control of the operation is independent of whether the units are on or off line as specified by the enable bits.

Bit 31 disables (0) or enables (1) the use of cache on instruction fetches. When bit 31 is off (0) instruction fetches bypass cache but memory return transfers (MRT) update cache.

GPR SPECIFIED BY R



(FOR BITS 27-30, 0= CACHE OFF AND 1=CACHE ON)
I CACHE = INSTRUCTION CACHE
O CACHE = OPERAND CACHE

830444

SUMMARY OF EXPRESSIONS

(R 27-31), → $(\mathrm{CMCR}_{0\text{-}4})$

CONDITION CODE RESULTS

No change

NOTES

1. CMC is a privileged instruction.

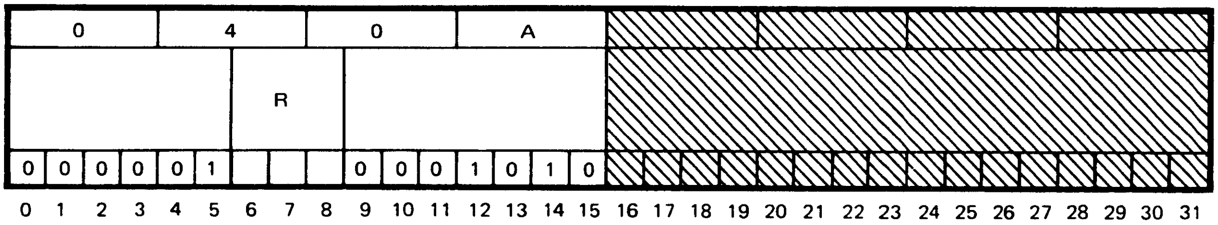
2. The CMC instruction operates differently for V6 and V9 CPU. This section describes the operation for the V6 CPU.

3. This instruction is for use by diagnostics and not for dynamic switching on/off line of cache units.

4. System reset initializes, enables and sets online both banks 0 and 1.

5. When cleared the cache RAMs contain undetermined data patterns.

6. If a bank is off, it must be initialized before it is enabled.



830315

DEFINITION

The Cache Memory Control (CMC) instruction is used to initialize and/or enable installed cache or shadow memory boards.

Bits 15-30 of the GPR specified by R are used to initialize and/or enable the resident cache boards in the CPU as shown below.

If shadow memory is installed, bits 16 through 18 will be used to initialize shadow memory units 1 through 3, respectively. Bits 20 through 22 will be used to enable the installed shadow memory units.

Bit 31 of R, when reset, cuases instructions to be bypassed from cache.

NOTES

1. If shadow memory is installed in a cache board slot, only the cache bank 1 init and enable bits are used. The respective cache bank 0 init and enable bits are ignored.
2. The CMC instruction operates differently for V6 and V9 CPU. This section describes the operation for the V9 CPU.
3. The CMC instruction is a privileged instruction.
4. This instruction is for diagnostics use only. All interrupts and traps are ignored.
5. System reset will clear and initialize all of cache memory. Shadow memory will remain unaffected by a system reset.
6. When shadow memory is initialized, the contents of the main memory locations covered by the shadow memory unit address range, are copied into the respective shadow memory units.
7. When the cache units are cleared, the respective index RAMS are set as follows:

Cache initialization patterns
(bit positions 8-19)

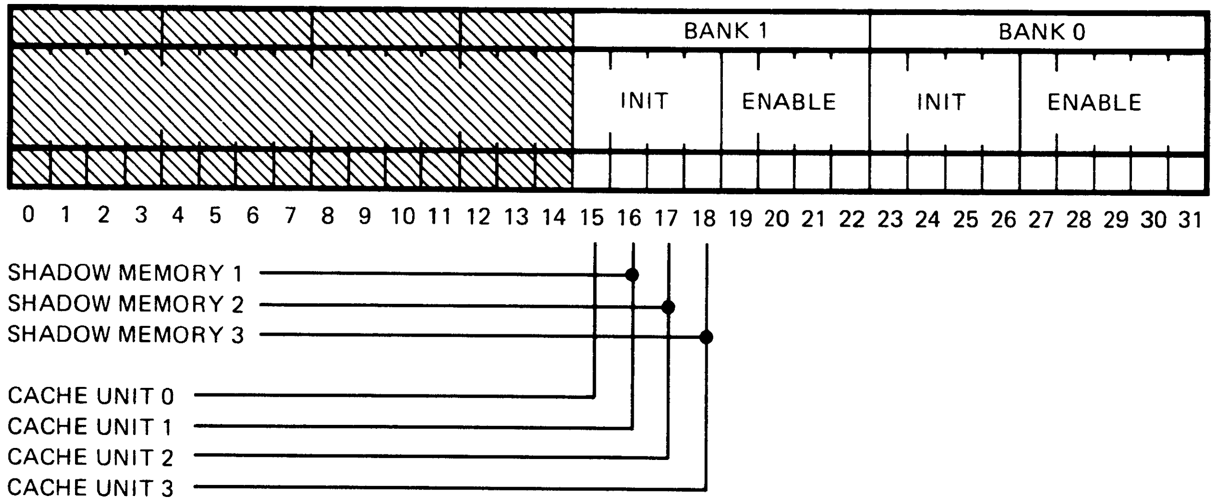
Cache Unit 0	-	Bank	0	-	1111	1111000X
		Bank	1	-	1111	1111001X
Cache Unit 1	-	Bank	0	-	1111	1111010X
		Bank	1	-	1111	1111011X
Cache Unit 2	-	Bank	0	-	1111	1111100X
		Bank	1	-	1111	1111101X
Cache Unit 3	-	Bank	0	-	1111	1111110X
		Bank	1	-	1111	1111111X

X = Don't care

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

Contents of R:



830982

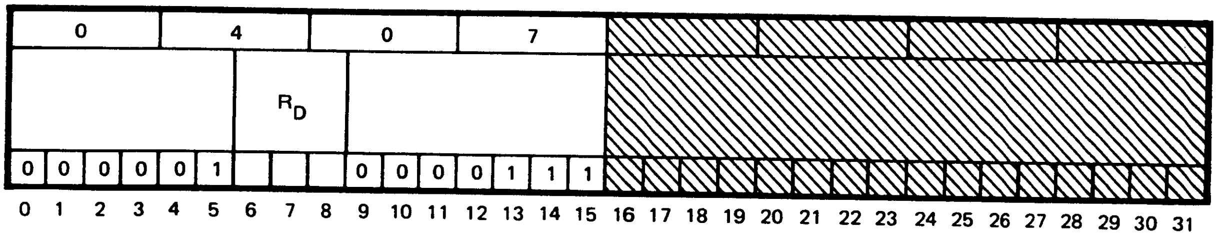
NOTE

Care must be taken when initializing and/or enabling the shadow memory. If a shadow memory unit is initialized, but not enabled, that respective unit may contain invalid data when it is enabled.

Bit 31 0 - Cache is bypassed on instruction fetches
1 - Normal cache operation

SHARED MEMORY CONTROL (V6 ONLY) 0407

SMC



DEFINITION

This instruction causes the contents of the general purpose register (GPR) specified by R_D to be transferred to the shared memory control logic of the CPU.

SUMMARY EXPRESSION

$R_D(1\text{-}12) \rightarrow \text{SMCL}$

CONDITION CODE RESULTS

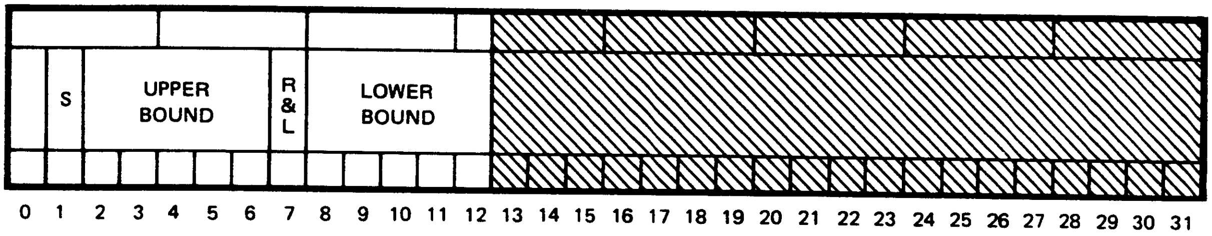
No change.

NOTES

1. SMC is valid for V6 only.

2. SMC is a privileged instruction.

3. The format of R_D is as follows:



830314

4. Bit 1 enables Shared Memory and Read and Lock within shared boundaries:

   Bit 1 = 0 Disable Memory Sharing (Bits 2-6, 8-12 disregard)
   = 1 Enable Memory Sharing (Bit 2-6, 8-12 provide upper and lower boundaries of Shared Memory)

5. Bits 8-12 specify the MSB's of the 24 bit Real Memory Address at which memory sharing is to begin. Bits 2-6 specify the MSB's of the 24 bit Real Memory Address at which memory sharing ends.

6. Bit 7 enables Memory Read & Lock to all memory

   Bit 7 = 0 Disable Read & Lock for non-shared memory
   = 1 Enable Read & Lock for all memory

7. Bit 7 must be set (Rd and Lk) for all CPU/IPU configurations.

8. Read and Lock causes SBM and ZBM to operate using memory (not cache) and prevents a second processor from accessing the target memory location for the duration of the SBM or ZBM instruction.

9. Shared memory boundaries cause the processor to turn-off cache and use memory for all memory accesses within the shared memory boundaries.

10. Shared memory boundaries must include all shared memory that can not echo memory writes from external ports to the processor cache.

LOWER AND UPPER BOUND SELECTION EXAMPLE

Lower Bound Bits 8-12	Upper Bound Bits 2-6	Lower Bound Address (Hex)	Upper Bound Address (Hex)
00000	00000	000000	07FFFC
00001	00001	080000	0FFFFC
00010	00010	100000	17FFFC
00011	00011	180000	1FFFFC
11110	11110	F00000	F7FFFC
11111	11111	F80000	FFFFFC
11111	00000	000000	FFFFFC

0				0				0				A				▨				▨				▨				▨			
0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

830316

DEFINITION

When the SIPU instruction is executed in the CPU the IPU trap is set in the IPU; when it is executed in the IPU the IPU trap is set in the CPU. The trap to the CPU is deferred if interrupts are blocked.

CONDITION CODE RESULTS

No change

ASSEMBLY LANGUAGE CODING

SIPU

NOTE

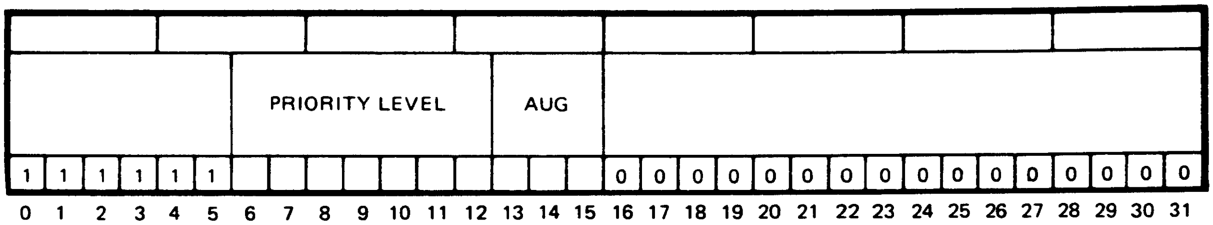
SIPU is an unprivileged instruction.

6.2.13 Interrupt Control Instructions

The interrupt control instructions are privileged instructions that control the interrupt functions of the CPU. The interrupt control instructions for nonextended I/O and RTOM interrupts (class 3 and class E protocols) will enable, disable, request, activate, or deactivate the interrupt operations to be performed on the priority level addressed. However, the interrupt control instructions for extended I/O channels (class F protocol) will enable, disable, activate, or deactivate operations from the CPU to an addressed channel. The other instructions in this group block or unblock external interrupts to the CPU. Interrupt control requests to the IPU are trapped as an undefined IPU instruction trap.

6.2.13.1 Instruction Format

Three special formats are used by the interrupt control instructions to accommodate the performance of these variations. The non-extended I/O instructions in this class have the same primary operation code of all ones, but each must designate the applicable interrupt priority level and contain an augmenting op code that corresponds to the specific instruction. The format used is as follows:

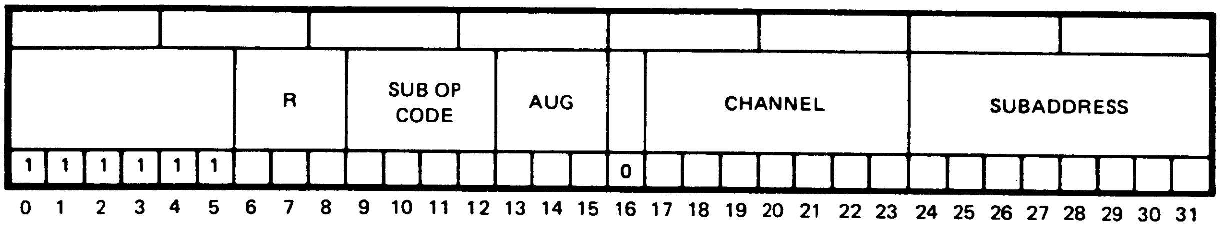


						PRIORITY LEVEL							AUG																		
1	1	1	1	1	1											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

830361

Bits 6-12	Binary priority level number of the relevant interrupt
Bits 13-15	Augmenting operation code
Bits 16-31	Unassigned

The channel-related interrupt control instructions (class F protocol), likewise, have the same primary operation code of all ones. However, the format used contains a channel address, a subaddress, an augment code, and a GPR designator, in addition to a subordinate op code in bits 9 through 12. The format is as follows:



830353

Bits 6-8 The R field, if nonzero, specifies a general purpose register the contents of which will be added to the channel and subaddress field (bits 16-31) to form the logical channel and subaddress.

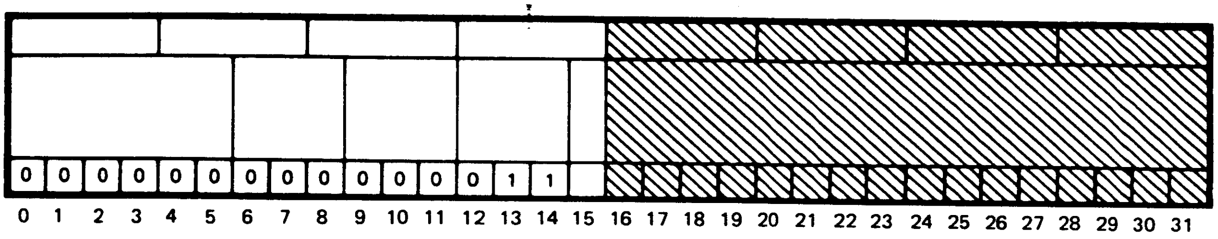
Bits 9-12 The subordinate operation code (SUB OP)

Bits 13-15 The augment code

Bits 16-31 A constant representing the logical channel and subaddress field. If the R field is zero, only the channel and subchannel fields will be used

Bits 16 Always zero

The block/unblock instructions for external interrupts use a halfword format that is all zeros in bits 0 through 11, and actually differ only in bit 15 of the four-bit designator contained in bits 12 through 15 (either hexadecimal 6 or hexadecimal 7).



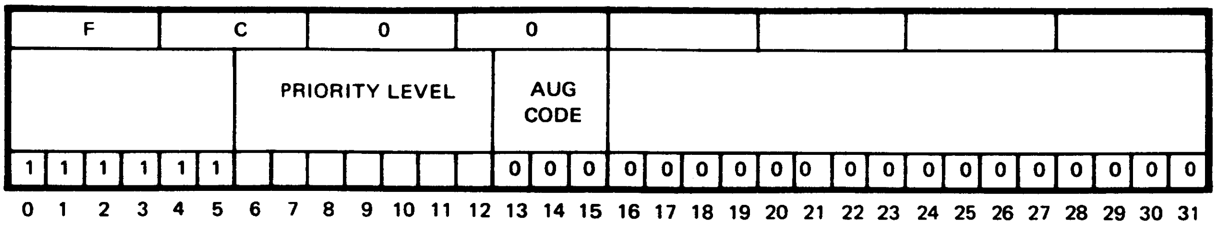
830356

6.2.13.2 Condition Code

Most interrupt control instructions leave the condition codes unchanged. Descriptions of the channel-related interrupt control instructions each contain specific comments on condition code disposition.

NOTE

Class F interrupt instructions are not included in this section, but are shown in the Extended I/O (class F) section.



830317

DEFINITION

The priority interrupt level specified by the priority level field (bits 6-12) in the instruction word (IW) is conditioned to respond to an interrupt signal. If bit position 0 of the PSD is reset to zero (unprivileged state), execution of this instruction will generate the privilege violation trap.

NOTES

1. Any stored requests for the specified level are eligible to become active.
2. Traps are always enabled.
3. This instruction has no affect on levels assigned to class F I/O and is treated as NOP for such levels.
4. EI is a privileged instruction.
5. In the V6 IPU only, EI may be used with class 6 or B interrupts otherwise an undefined IPU trap occurs.

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

ASSEMBLY LANGUAGE CODING

EI level

REQUEST INTERRUPT FC02

RI
V

<table>
<tr><td colspan="4">F</td><td colspan="4">C</td><td colspan="4">0</td><td colspan="4">2</td><td colspan="4"></td><td colspan="4"></td><td colspan="4"></td><td colspan="4"></td></tr>
<tr><td colspan="6"></td><td colspan="7">PRIORITY LEVEL</td><td colspan="3">AUG CODE</td><td colspan="16"></td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
</table>

830318

DEFINITION

An interrupt request signal is applied to the interrupt level specified by the priority level field (bits 6-12) in the instruction word (IW). This signal simulates the signal generated by the internal or external condition connected to the specified level. If bit position 0 of the PSD is reset to zero (unprivileged state), execution of this instruction will generate the privilege violation trap. The interrupt request signal is stored in the specified level whether or not it is enabled and/or active.

NOTES

1. For RIs on levels 0 or 1, the RTOM jumpers select either that levels 0 and 1 are enabled, or that software enables are required.

2. This instruction has no affect on levels assigned to class F I/O and is treated as NOP for such levels.

3. RI is a privileged instruction.

4. In the V6 IPU only, RI may be used with class 6 or B interrupts otherwise an undefined IPU trap occurs.

CONDITION CODE RESULTS

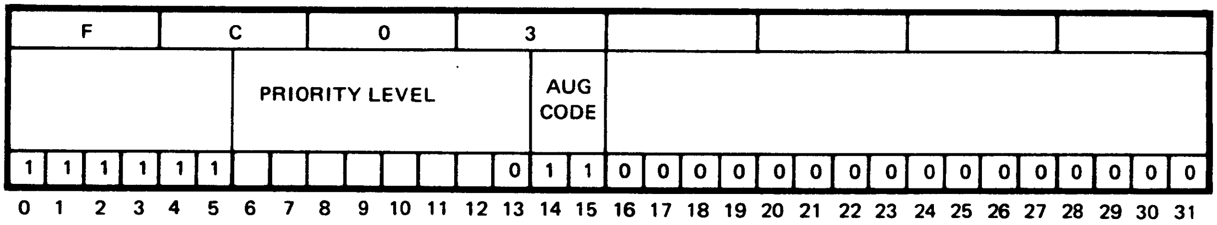
CC1: No change
CC2: No change
CC3: No change
CC4: No change

ASSEMBLY LANGUAGE CODING

RI level

ACTIVATE INTERRUPT FC03

AI
V



830319

DEFINITION

A signal is applied to set the active condition in the priority interrupt level specified by the priority level field (bits 6-12) in the instruction word (IW). The active level is set in the specified level whether or not that level is enabled. This condition prohibits this level and any lower levels not already in service from being serviced until this level is deactivated. However, request signals occurring at this or lower levels are stored for subsequent servicing. If bit position 0 of the PSD is reset to zero (unprivileged state), execution of this instruction will generate the privilege violation trap.

NOTES

1. This instruction has no affect on levels assigned to class F I/O and is treated as NOP for such levels.

2. AI is a privileged instruction.

3. In the V6 IPU only, AI may be used with class 6 of B interrupts otherwise an undefined IPU trap occurs.

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

ASSEMBLY LANGUAGE CODING

AI level

F				C				0				1																			
						PRIORITY LEVEL							AUG CODE																		
1	1	1	1	1	1								0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

830320

DEFINITION

The priority interrupt level specified by the priority level field (bits 6-12) in the instruction word (IW) is disabled and will not respond to an interrupt signal. If bit position 0 of the PSD is reset to zero (unprivileged state), execution of this instruction will generate the privilege violation trap. The active state of the interrupt is not affected.

NOTES

1. Any unserviced request signal at this level is cleared by execution of this instruction.

2. Traps are always enabled.

3. This instruction has no affect on levels assigned to class F I/O and is treated as NOP for such levels.

4. DI is a privileged instruction.

5. In the V6 IPU only, DI may be used with class 6 or B interrupts otherwise an undefined IPU trap occurs.

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

ASSEMBLY LANGUAGE CODING

DI level

F				C				0				4																			
						PRIORITY LEVEL							AUG CODE																		
1	1	1	1	1	1								1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

830321

DEFINITION

A signal is applied to reset the active condition in the priority interrupt level specified by the priority level field (bits 6-12) in the instruction word (IW). The specified level is set inactive whether the level is enabled or disabled. Execution of the deactivate interrupt instruction does not clear any request signals on the specified level or any other level. If bit position 0 of the PSD is reset to zero (unprivileged state), execution of this instruction will generate the privilege violation trap.

NOTES

1. This instruction has no affect on levels assigned to class F I/O and is treated as a NOP for such levels.

2. DAI and the following instruction are executed as an uninterruptible pair.

3. DAI is a privileged instruction.

4. In the V6 IPU only, DAI may be used with class 6 of B interrupts otherwise an undefined IPU trap occurs.

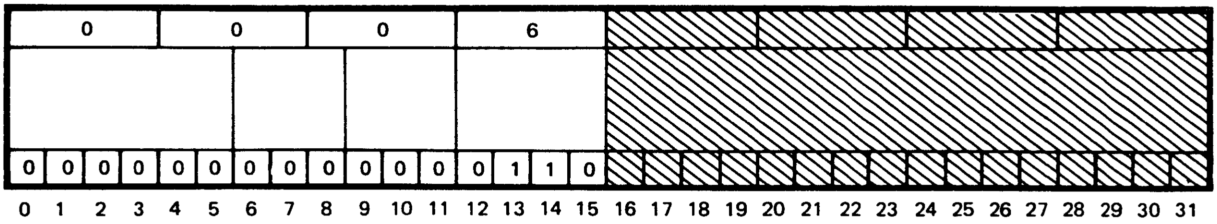
CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

ASSEMBLY LANGUAGE CODING

DAI level

BLOCK EXTERNAL INTERRUPTS 0006 BEI



830322

DEFINITION

The execution of this instruction prevents the CPU from sensing all interrupt requests generated by the I/O channel and RTOM or IOP. Also the IPU and console attention traps are inhibited.

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

ASSEMBLY LANGUAGE CODING

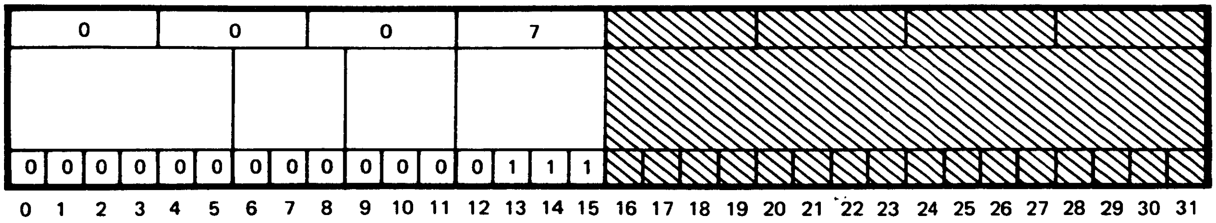
BEI

NOTES

1. BEI is a privileged instruction.

2. Causes an undefined IPU trap if executed in the IPU.

UNBLOCK EXTERNAL INTERRUPTS UEI
0007



830323

DEFINITION

The execution of this instruction allows the central processing unit to sense all interrupt requests generated by the I/O channel and RTOM or IOP.

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

ASSEMBLY LANGUAGE CODING

UEI

NOTES

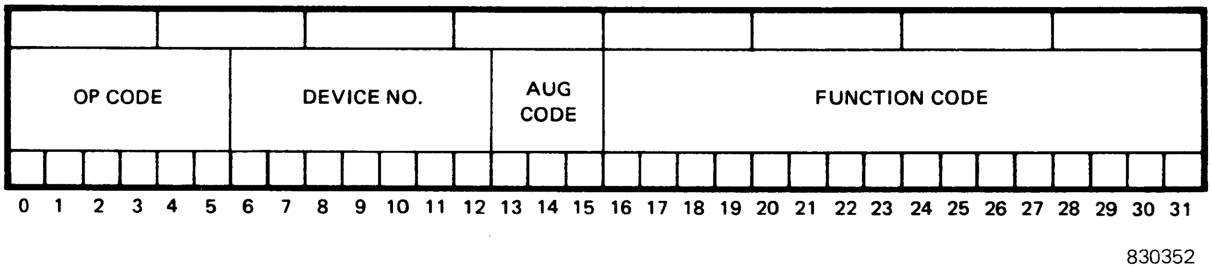
1. UEI is a privileged instruction.
2. This instruction may be executed in the V6 IPU.

6.2.14 Input/Output Instructions

The input/output (I/O) instructions consist of two distinct groups. In the first group, the command device (CD) and test device (TD) instructions provide the capability to conduct command and test operations to peripheral devices. These two instructions cause a 16-bit "function code" to be sent to the peripheral device specified by the device number. In the second group, I/O instructions provide various other capabilities as designated by each subordinate operation code. The V9 IPU cannot execute I/O instructions. If attempted, an IPU undefined instruction trap will be generated.

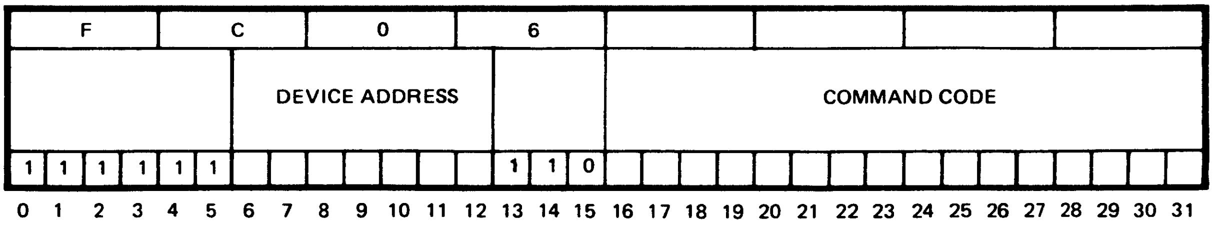
6.2.14.1 Command Device and Test Device

The following instruction format is used for the command device and test device instructions only:



Bits 0-5	Operation code
Bits 6-12	Peripheral device identifying number
Bits 13-15	Augmenting operation code
Bits 16-31	16-bit "function" code

During execution of a test device instruction, the condition code is set to indicate the result of the test being performed. The command device instruction leaves the current condition code unchanged.



830324

DEFINITION

The contents of the command code field (bits 16-31) are transferred to the device controller channel (DCC) specified by the device address contained in bit positions 6-12 of the instruction word.

CONDITION CODE RESULTS

CC1: No change
CC2: No change
CC3: No change
CC4: No change

ASSEMBLY EXAMPLE

	Dev Addr	Comm Code	Command
CD	X'7A',	X'8000'	Output data to device 7A
CD	X'78',	X'9000'	Input data from device 78

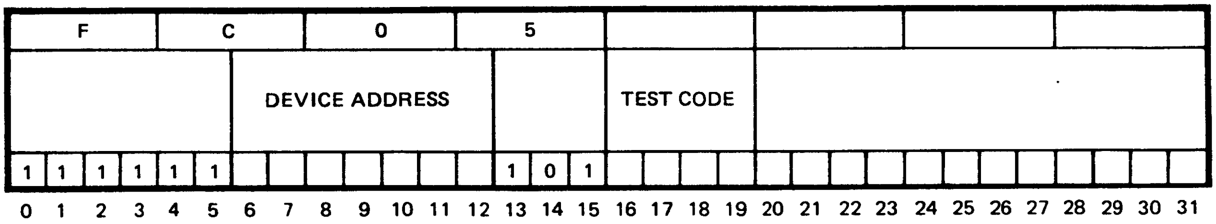
NOTES

1. This instruction is for class 3 and class E I/O processors only.

2. If a CD instruction to a class F channel is attempted, a system check trap will occur.

3. CD is a privileged instruction.

4. This instruction in the V6 IPU is only for class B (interval timer), otherwise the IPU interprets it as an undefined IPU instruction.



830325

DEFINITION

The contents of the test code field (bits 16-19) are transferred to the device controller channel (DCC) specified by the device address contained in bit positions 6-12 of the instruction word. The device test defined by the test code is performed in the DCC, and the test results are loaded into condition code bits 1-4 (CC_{1-4}).

A TD having a unique test code is available with most peripheral devices. Execution of a TD with this code causes a snapshot of all device and DCC status to be stored in memory. The individual peripheral device reference manuals define the operation of this instruction with each device.

CONDITION CODE RESULTS

Test results defined for specific peripheral device (refer to Figure 5-7).

ASSEMBLY EXAMPLE

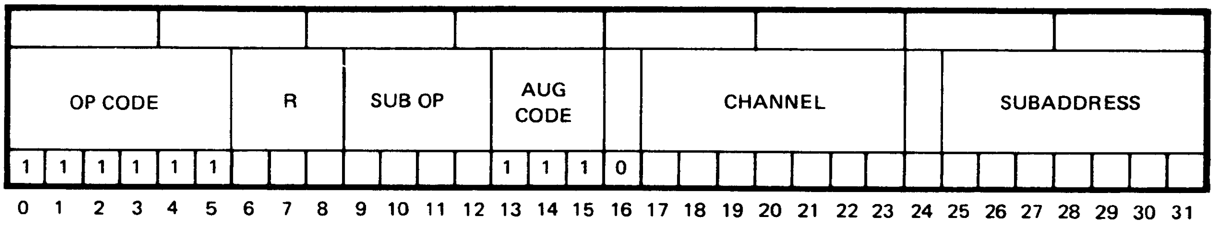
	Dev Addr	Comm Code	Command
TD	X'10',	X'8000'	Request the controller status for unit 10
TD	X'10',	X'2000'	Request the device status for unit 10

NOTES

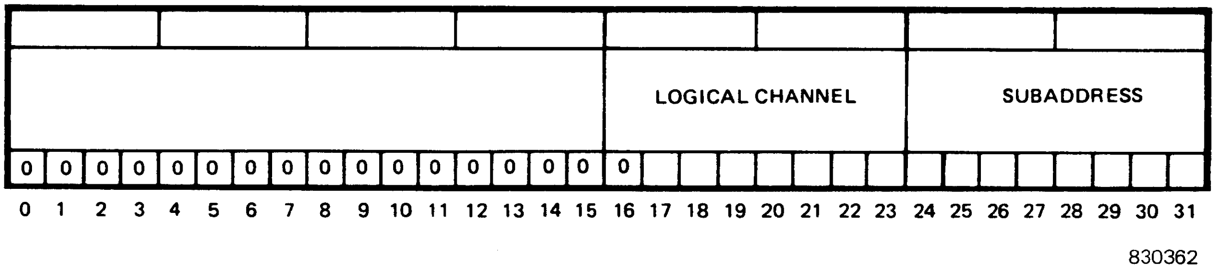
1. This instruction is for class 3 and class E I/O processors only.
2. If a TD instruction to a class F channel is attempted, a system check trap will occur.
3. TD is a privileged instruction.
4. This instruction in the V6 IPU is only for class B (interval timer), otherwise the IPU interprets it as an undefined IPU instruction.

6.2.15 Class F I/O Instructions

All class F I/O instructions will be in the following format:



OP CODE, bits 0-5, and AUG CODE, bits 13-15, must contain ones. The R field (bits 6-8), if nonzero, specifies a general purpose register the contents of which will be added to the channel and subaddress field (bits 16-31) to form the logical channel and subaddress. If R is specified as zero, only the channel and subaddress fields will be used. The format of the computed logical channel and subaddress is:



The subaddress will be ignored by the channel if the operation does not apply to a controller or device.

The sub op field (bits 9-12) specifies the type of operation that is to be performed as described below:

Bits 9-12	Sub op
0 0 0 0 - X'0'	Unassigned
0 0 0 1 - X'1'	Unassigned
0 0 1 0 - X'2'	Start I/O (SIO)
0 0 1 1 - X'3'	Test I/O (TIO)
0 1 0 0 - X'4'	Stop I/O (STPIO)
0 1 0 1 - X'5'	Reset channel (RSCHNL)
0 1 1 0 - X'6'	Halt I/O (HIO)
0 1 1 1 - X'7'	Grab controller (GRIO)
1 0 0 0 - X'8'	Reset controller (RSCTL)
1 0 0 1 - X'9'	Enable write channel WCS (ECWCS)
1 0 1 0 - X'A'	Unassigned
1 0 1 1 - X'B"	Write channel WCS (WCWCS)
1 1 0 0 - X'C'	Enable channel interrupt (ECI)
1 1 0 1 - X'D'	Disable channel interrupt (DCI)
1 1 1 0 - X'E'	Activate channel interrupt (ACI)
1 1 1 1 - X'F'	Deactivate channel interrupt (DACI)

NOTES

1. In CPU, the channel must be ICL'd as Class F.

2. Condition Codes must be tested after each instruction.

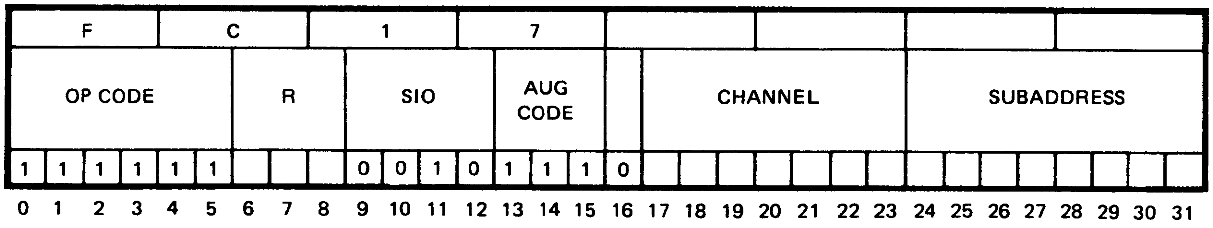
3. CD, TD, EI, DI, AI, DAI, and RI cannot be executed to a class F channel.

4. Class F I/O instructions are all privileged instructions.

5. If these instructions are executed for other than Class F I/O devices/channels, a system check trap will occur.

6. In the V6 IPU, the channel must be ICL'd as class 7. The V9 does not have I/O capability.

7. These instructions cannot be the target of an execute instruction.



DEFINITION

Start I/O (SIO) will be used to begin I/O execution or to return appropriate condition codes and status if I/O execution could not be accomplished.

Bits 0-5	Specifies the operation code.
Bits 6-8	Specifies a general purpose register; if this GPR is not R0, then its contents will be added to the channel and subaddress field to form the effective logical channel and subaddress.
Bits 9-12	Specifies the operation as SIO.
Bits 13-15	Specifies the augment code.
Bits 16-31	Specifies the constant that will be added to the contents of the GPR specified by R to form the effective logical channel and subaddress. If R is zero, bits 16-31 alone will be used to specify the logical channel and subaddress.

CONDITION CODE RESULTS

CC1	CC2	CC3	CC4	
0	0	0	0	Request activated, will echo status
0	0	0	1	Channel busy
0	0	1	0	Channel inoperable or undefined
0	0	1	1	Subchannel busy
0	1	0	0	Status stored (SIO rejected)
0	1	0	1	Unsupported transaction
0	1	1	0	Unassigned
0	1	1	1	Unassigned
1	0	0	0	Request accepted and queued, no echo status

ASSEMBLY LANGUAGE CODING

SIO R, '(Constant)

NOTE

1. Condition codes, after execution of a SIO, will be set and can be tested by a subsequent conditional branch instruction to ascertain if the I/O was accepted.

2. Start I/O is a privileged instruction.

<table>
<tr><td colspan="4">F</td><td colspan="4">C</td><td colspan="4">1</td><td colspan="4">F</td><td colspan="4"></td><td colspan="4"></td><td colspan="4"></td><td colspan="4"></td></tr>
<tr><td colspan="6">OP CODE</td><td colspan="3">R</td><td colspan="4">TIO</td><td colspan="3">AUG CODE</td><td></td><td colspan="7">CHANNEL</td><td colspan="8">SUBADDRESS</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td></td><td></td><td></td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
</table>

830327

DEFINITION

Test I/O (TIO) will be used to test the controller state and to return appropriate condition codes and status reflecting the state of the addressed controller and/or device. Channel implementation will dictate the depth to which the channel must test to determine current state.

Bits 0-5	Specifies the operation code.
Bits 6-8	Specifies a general purpose register; if this GPR is not R0, then its contents will be added to the channel and subaddress field to form the effective logical channel and subaddress.
Bits 9-12	Specifies the operation as TIO.
Bits 3-15	Specifies the augment code.
Bits 16-31	Specifies a constant that will be added to the contents of the GPR specified by R to form the effective logical channel and subaddress. If R is zero, bits 16-31 alone will be used to specify the logical channel and subaddress.

CONDITION CODE RESULTS

CC1	CC2	CC3	CC4	
0	0	0	0	Request activated, will echo status
0	0	0	1	Channel busy
0	0	1	0	Channel inoperable or undefined
0	0	1	1	Subchannel busy
0	1	0	0	Status stored
0	1	0	1	Unsupported transaction
0	1	1	0	Unassigned
0	1	1	1	Unassigned
1	0	0	0	Request accepted and queued, no echo status

ASSEMBLY LANGUAGE CODING

TIO R '(Constant)

NOTE

1. Condition codes, after execution of the TIO, will be set and can be tested by a subsequent conditional branch instruction to ascertain channel/controller/device state.

2. Test I/O is a privileged instruction.

<table>
<tr><td colspan="4">F</td><td colspan="4">C</td><td colspan="4">2</td><td colspan="4">7</td><td colspan="4"></td><td colspan="4"></td><td colspan="4"></td><td colspan="4"></td></tr>
<tr><td colspan="6">OP CODE</td><td colspan="3">R</td><td colspan="4">STPIO</td><td colspan="3">AUG CODE</td><td></td><td colspan="7">CHANNEL</td><td colspan="8">SUBADDRESS</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td></td><td></td><td></td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
</table>

830328

DEFINITION

Stop I/O (STPIO) terminates the current I/O operation after the completion of the action specified by the current IOCD. The STPIO applies only to the addressed subchannel, its only function is to suppress command and data chain flags in the current IOCD.

Bits 0-5 Specifies the operation code.

Bits 6-8 Specifies a general purpose register; if this GPR is not R0, then its contents will be added to the channel and subaddress field to form the effective logical channel and subaddress.

Bits 9-12 Specifies the operation as STPIO.

Bits 13-15 Specifies the augment code.

Bits 16-31 Specifies a constant that will be added to the contents of the GPR specified by R to form the effective logical channel and subaddress. If R is zero, bits 16-31 alone will be used to specify the logical channel and subaddress.

CONDITION CODE RESULTS

CC1	CC2	CC3	CC4	
0	0	0	0	Request activated, will echo status
0	0	0	1	Channel busy
0	0	1	0	Channel inoperable or undefined
0	0	1	1	Subchannel busy
0	1	0	0	Status stored
0	1	0	1	Unsupported transaction
0	1	1	0	Unassigned
0	1	1	1	Unassigned
1	0	0	0	Request accepted and queued, no echo status

ASSEMBLY LANGUAGE CODING

STPIO R, '(Constant)'

NOTE

1. Condition codes, after execution of a STPIO, will be set and can be tested by a subsequent conditional branch instruction to ascertain channel/controller/device state.

2. Stop I/O is a privileged instruction.

<table>
<tr><td colspan="4">F</td><td colspan="4">C</td><td colspan="4">2</td><td colspan="4">F</td><td colspan="4"></td><td colspan="4"></td><td colspan="4"></td><td colspan="4"></td></tr>
<tr><td colspan="6">OP CODE</td><td colspan="3">R</td><td colspan="4">RSCHNL</td><td colspan="3">AUG CODE</td><td colspan="8">CHANNEL</td><td colspan="8">SUBADDRESS</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td></td><td></td><td></td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
</table>

830329

DEFINITION

Reset channel (RSCHNL) causes the addressed channel to cease and reset all activity and to return to the idle state. The channel will also reset the subchannels. No controller or device will be affected. Any requesting or active interrupt level will be reset.

Bits 0-5	Specifies the operation code.
Bits 6-8	Specifies a general purpose register; if this GPR is not R0, then its contents will be added to the channel and subaddress field to form the effective logical channel and subaddress.
Bits 9-12	Specifies the operation as RSCHNL.
Bits 13-15	Specifies the augment code.
Bits 16-31	Specifies the constant that will be added to the contents of R to form the effective logical channel and subaddress. If R is zero, bits 16-31 alone will be used to specify the logical channel and subaddress.

CONDITION CODE RESULTS

CC1	CC2	CC3	CC4	
0	0	0	0	Request activated, will echo status
0	0	1	0	Channel inoperable or undefined
1	0	0	0	Request accepted and queued, no echo status

ASSEMBLY LANGUAGE CODING

RSCHNL R, '(Constant)'

NOTE

1. Condition codes, after execution of a RSCHNL, will be set and can be tested by a subsequent conditional branch instruction to ascertain channel/controller/device state.

2. Reset channel is a privileged instruction.

3. The channel remains busy for an extended period of time following RSCHNL.

F				C				3				7																			
OP CODE						R			HIO				AUG CODE				CHANNEL							SUBADDRESS							
1	1	1	1	1	1				0	1	1	0	1	1	1	0															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

830330

DEFINITION

Halt I/O (HIO) causes an immediate but orderly termination in the controller. The device end condition will notify the software of the actual termination in the controller, thus indicating its availability for new requests. If the Halt I/O caused the generation of status relating to the terminated I/O operation, then the device end condition for the termination of the I/O operation will be the only device end condition generated.

Bits 0-5	Specifies the operation code.
Bits 6-8	Specifies a general purpose register; if this GPR is not R0, then its contents will be added to the channel and subaddress field to form the effective logical channel and subaddress.
Bits 9-12	Specifies the operation as HIO.
Bits 13-15	Specifies the augment code.
Bits 16-31	Specifies the constant that will be added to the contents of R to form the effective logical channel and subaddress. If R is zero, bits 16-31 alone will be used to specify the logical channel and subaddress.

CONDITION CODE RESULTS

CC1	CC2	CC3	CC4	
0	0	0	0	Request activated, will echo status
0	0	0	1	Channel busy
0	0	1	0	Channel inoperable or undefined
0	0	1	1	Subchannel busy
0	1	0	0	Status stored
0	1	0	1	Unsupported transaction
0	1	1	0	Unassigned
0	1	1	1	Unassigned
1	0	0	0	Request accepted and queued, no echo status

ASSEMBLY LANGUAGE CODING

HIO R, '(Constant)'

NOTE

1. Condition codes, after execution of the HIO, will be set and can be tested by a subsequent conditional branch instruction to determine if the HIO was successfully executed.
2. Halt I/O is a privileged instruction.

GRAB CONTROLLER FC3F — GRIO s,v

F				C				3				F																			
OP CODE						R			GRIO				AUG CODE				CHANNEL								SUBADDRESS						
1	1	1	1	1	1				0	1	1	1	1	1	1	0															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

830331

DEFINITION

Grab controller (GRIO) will cause the addressed controller to release itself from the currently assigned channel and to reserve itself for the grabbing channel.

Bits 0-5 Specifies the operation code.

Bits 6-8 Specifies a general purpose register; if this GPR is not R0, then its contents will be added to the channel and subaddress field to form the effective logical channel and subaddress.

Bits 9-12 Specifies the operation as GRIO.

Bits 13-15 Specifies the augment code.

Bits 16-31 Specifies the constant that will be added to the contents of the GPR specified by R to form the effective logical channel and subaddress. If R is zero, bits 16-31 alone will be used to specify the logical channel and subaddress.

CONDITION CODE RESULTS

CC1	CC2	CC3	CC4	
0	0	0	0	Request activated, will echo status
0	0	0	1	Channel busy
0	0	1	0	Channel inoperable or undefined
0	0	1	1	Subchannel busy
0	1	0	0	Status stored
0	1	0	1	Unsupported transaction
0	1	1	0	Unassigned
0	1	1	1	Unassigned
1	0	0	0	Request accepted and queued, no echo status

ASSEMBLY LANGUAGE CODING

GRIO R, '(Constant)'

NOTE

1. Condition codes, after execution of a GRIO, will be set and can be tested by a subsequent conditional branch instruction to determine if the GRIO was successfuly executed.

2. Grab controller is a privileged instruction.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
F				C				4				7																			
						R			RSCTL				AUG CODE				CHANNEL							SUBADDRESS							
1	1	1	1	1	1				1	0	0	0	1	1	1	0															

830332

DEFINITION

Reset controller (RSCTL) causes the addressed controller to be completely reset. In addition, the subchannel and all pending and generated status conditions are cleared.

Bits 0-5	Specifies the operation code.
Bits 6-8	Specifies a general purpose register; if this GPR is not R0, then its contents will be added to the channel and subaddress fields to form the effective logical channel and subaddress.
Bits 9-12	Specifies the operation as RSCTL.
Bits 13-15	Specifies the augment code.
Bits 16-31	Specifies the constant that will be added to the contents of the GPR specified by R to form the effective logical channel and subaddress. If R is zero, bits 16-31 alone will be used to specify the logical channel and subaddress.

CONDITION CODE RESULTS

CC1	CC2	CC3	CC4	
0	0	0	0	Request activated, will echo status
0	0	0	1	Channel busy
0	0	1	0	Channel inoperable or undefined
0	0	1	1	Subchannel busy
0	1	0	0	Status stored
0	1	0	1	Unsupported transaction
0	1	1	0	Unassigned
0	1	1	1	Unassigned
1	0	0	0	Request accepted and queued, no echo status

ASSEMBLY LANGUAGE CODING

RSCTL R, '(Constant)'

NOTE

1. Condition Codes, after execution of a RSCTL, will be set and can be tested by a subsequent conditional branch instruction to determine if the RSCTL was successfully executed.

2. Reset controller is a privileged instruction.

This page intentionally left blank

6.2.16 Class F I/O Writable Control Storage (WCS) Instructions

Writable control storage (WCS) is an option available for use with the class F I/O controller. The WCS consists of one or two random access memory (RAM) logic boards, each containing 2K x 64 bits of RAM memory. The WCS supplements the firmware in the class F I/O controller.

6.2.16.1 Instruction Format

The following format is used for class F I/O controller-associated WCS instructions.

OP CODE	R	CWCS CODE	AUG CODE	CONSTANT
0 1 2 3 4 5	6 7 8	9 10 11 12	13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

830357

Bits 0-5 Define the operation code.

Bits 6-8 Specify a GPR; if this GPR is not R0, then its contents will be added to the constant to form the effective logical channel and subaddress.

Bits 9-12 Specify the channel WCS operation code.

Bits 13-15 Define the augmenting operation code.

Bits 16-31 Specify a constant that will be added to the contents of the GPR designated by R to form the effective logical channel and subaddress. If R is zero, the constant alone will be used to specify the logical channel and subaddress.

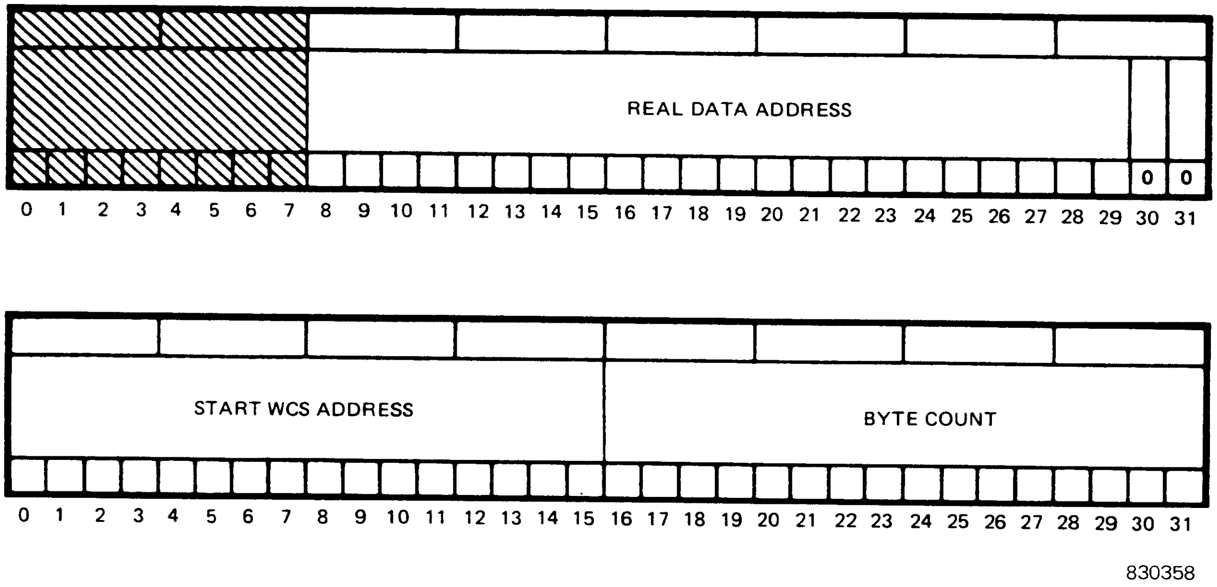
6.2.16.2 Condition Code

When using the class F I/O controller WCS, the condition codes are changed in accordance with the WCS instructions.

6.2.16.3 WCS Programming

Programming of the class F I/O controller associated WCS is presented in the individual I/O processor publications.

6.2.16.4 IOCD Format for Class F I/O WCS



Real Data Address: Bits 8-31 (MSW) will contain the address of the physical memory location for the first word to be loaded.

Start WCS Address: Bits 0-15 (LSW) will contain the address of the location in WCS in which the first word is to be loaded.

Byte Count: Bits 16-31 (LSW) will contain the number of bytes to be loaded.

F				C				4				F																			
OP CODE						R			ECWCS				AUG CODE				CHANNEL							SUBADDRESS							
1	1	1	1	1	1				1	0	0	1	1	1	1	0															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

830333

DEFINITION

Enable channel WCS load (ECWCS) sets an interlock within the central processing unit (CPU) to enable the loading of WCS. The ECWCS must be the first instruction of a two-instruction sequence the second instruction being WCWCS.

Bits 0-5	Specifies the operation code.
Bits 6-8	Specifies a general purpose register; if this GPR is not R0, then its contents will be added to the channel and subaddress field to form the effective logical channel and subaddress.
Bits 9-12	Specifies the operation as ECWCS.
Bits 13-15	Specifies the augment code.
Bits 16-31	Specifies the constant that will be added to the contents of the GPR specified by R to form the effective logical channel and subaddress. If R is zero, bits 16-31 alone will be used to specify the logical channel and subaddress.

CONDITION CODE RESULTS

CC1	CC2	CC3	CC4	
0	0	0	0	Request activated, will echo status
0	0	0	1	Channel busy
0	0	1	0	Channel inoperable or undefined
0	0	1	1	Subchannel busy
0	1	0	0	Status stored
0	1	0	1	Unsupported transaction
0	1	1	0	Unassigned
0	1	1	1	Unassigned
1	0	0	0	Request accepted and queued, no echo status

NOTES

1. Condition codes after the execution of the ECWCS instruction will be set and can be tested by a subsequent conditional branch instruction to ascertain whether the ECWCS instruction was successfully executed.

2. ECWCS is a privileged instruction.

WRITE CHANNEL WCS FC5F

WCWCS

F				C				5				F																			
OP CODE						R			WCWCS				AUG CODE				CHANNEL							SUBADDRESS							
1	1	1	1	1	1				1	0	1	1	1	1	1	0															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

830334

DEFINITION

Write channel WCS (WCWCS) causes the loading of the channel WCS. The WCWCS must be the second of an instruction pair executed to the class F I/O controller, the first being ECWCS; no other I/O instructions to the class F I/O controller to be loaded can intervene.

Bits 0-5	Specifies the operation code.
Bits 6-8	Specifies a general purpose register; if this GPR is not R0, then its contents will be added to the channel and subaddress field to form the effective logical channel and subaddress.
Bits 9-12	Specifies the operation as WCWCS.
Bit 13-15	Specifies the augment code.
Bits 16-31	Specifies a constant that will be added to the contents of the GPR specified by R to form the effective logical channel and subaddress. If R is zero, bits 16-31 alone will be used to specify the logical channel and subaddress.

CONDITION CODE RESULTS

CC1	CC2	CC3	CC4	
0	0	0	0	Request activated, will echo status
0	0	0	1	Channel busy
0	0	1	0	Channel inoperable or undefined
0	0	1	1	Subchannel busy
0	1	0	0	Status stored
0	1	0	1	Unsupported transaction
0	1	1	0	Unassigned
0	1	1	1	Unassigned
1	0	0	0	Request accepted and queued, no echo status

NOTES

1. The information that is required by the WCS load will be passed to the class F I/O controller by a parameter list. The IOCD address location specified for this controller will be initialized by software prior to the execution of this instruction. The subaddress field will be ignored. The IOCD format is shown in paragraph 6.2.16.1.

2. If the WCWCS instruction is not preceded by an ECWCS instruction, a system check trap will occur.

3. WCWCS is a privileged instruction.

This page intentionally left blank

F				C				6				7																			
OP CODE						R			ECI				AUG CODE				CHANNEL								SUBADDRESS						
1	1	1	1	1	1				1	1	0	0	1	1	1	0								0							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

830336

DEFINITION

The enable channel interrupt enables the addressed channel to request interrupts from the CPU.

Bits 0-5	Specify the operation code.
Bits 6-8	Specify a general purpose register; if this GPR is not R0, then its contents will be added to the channel and subaddress field to form the logical channel and subaddress.
Bits 9-12	Specify the operation as ECI.
Bits 13-15	Specify the augment code.
Bits 16-31	Specify the constant that will be added to the contents of the GPR specified by R to form the effective logical channel and subaddress. If R is zero, the constant alone will be used to specify the logical channel and subaddress.

NOTES

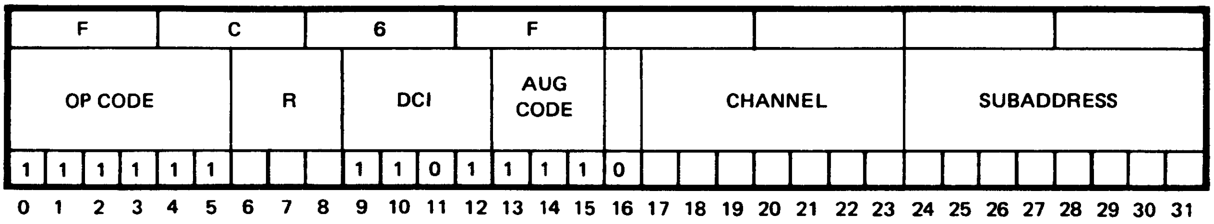
1. Condition codes after execution of the ECI will be set and can be tested by a subsequent conditional branch instruction to determine if the ECI was accepted by the channel.
2. In CPU, if this instruction is executed for other than a Class F I/O device channel, an undefined instruction trap will occur.
3. ECI is a privileged instruction.
4. The subaddress is not relevant for this instruction.
5. In the V6 IPU, if this instruction is executed for other than class 7, an undefined IPU trap occurs.

CONDITION CODE RESULTS

CC1	CC2	CC3	CC4	
0	0	0	0	Request activated, will echo status
0	0	0	1	Channel busy
0	0	1	0	Channel inoperable or undefined
0	0	1	1	Subchannel busy
0	1	0	0	Status stored
0	1	0	1	Unsupported transaction
0	1	1	0	Unassigned
0	1	1	1	Unassigned
1	0	0	0	Request accepted and queued, no echo status

ASSEMBLY LANGUAGE CODING

ECI R, '(Constant)'



830337

DEFINITION

The disable channel interrupt causes the addressed channel to be disabled from requesting interrupts from the CPU.

Bits 0-5	Specify the operation code.
Bits 6-8	Specify a general purpose register; if this GPR is not R0, then its contents will be added to the channel and subaddress field to form the effective logical channel and subaddress.
Bits 9-12	Specify the operation as DCI.
Bits 13-15	Specify the augment code.
Bits 16-31	Specify the constant that will be added to the contents of the GPR specified by R to form the effective logical channel and subaddress. If R is zero, the constant alone will be used to specify the logical channel and subaddress.

NOTES

1. Condition codes after execution of the DCI will be set and can be tested by a subsequent conditional branch instruction to determine if the DCI was accepted.
2. In CPU, if this instruction is executed for other than a Class F I/O device channel, an undefined instruction trap will occur.
3. DCI is a privileged instruction.
4. This instruction does not clear pending requests.
5. The subaddress is not relevant for this instruction.
6. In the V6 IPU, if this instruction is executed for other than class 7, an undefined IPU trap occurs.

CONDITION CODE RESULTS

CC1	CC2	CC3	CC4	
0	0	0	0	Request activated, will echo status
0	0	0	1	Channel busy
0	0	1	0	Channel inoperable or undefined
0	0	1	1	Subchannel busy
0	1	0	0	Status stored
0	1	0	1	Unsupported transaction
0	1	1	0	Unassigned
0	1	1	1	Unassigned
1	0	0	0	Request accepted and queued, no echo status

ASSEMBLY LANGUAGE CODING

DCI R, '(Constant)'

F				C				7				7																			
OP CODE						R			ACI				AUG CODE				CHANNEL							SUBADDRESS							
1	1	1	1	1	1				1	1	1	0	1	1	1	0															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

830335

DEFINITION

The activate channel interrupt will cause the addressed channel to begin actively contending with other interrupt levels, causing a blocking of its level, and all lower priority levels, from requesting an interrupt. If a request is currently pending in the channel, the request interrupt is removed but the interrupt level remains in contention.

Bits 0-5	Specify the operation code.
Bits 6-3	Specify a general purpose register; if this GPR is not R0, then its contents will be added to the channel and subaddress field to form the effective logical channel and subaddress.
Bits 9-12	Specify the operation as an ACI.
Bits 13-15	Specify the augment code.
Bits 16-31	Specify a constant that will be added to the contents of the GPR specified by R to form the effective logical channel and subaddress. If R is zero, the constant alone will be used to specify the logical channel subaddress.

NOTES

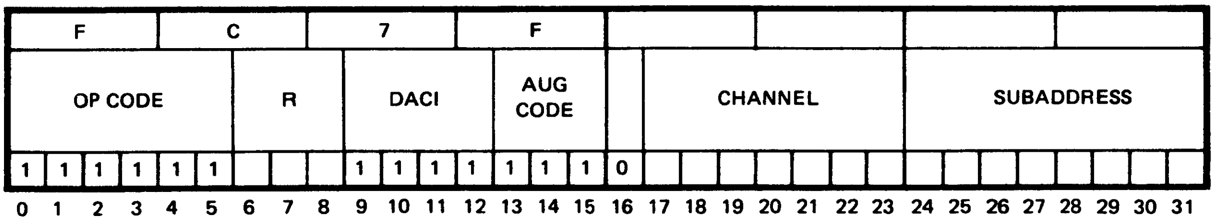
1. Condition codes, after execution of the ACI, will be set and can be tested by a subsequent conditional branch instruction to determine if the ACI was accepted by the channel.
2. In CPU, if this instruction is executed for other than a Class F I/O device channel, an undefined instruction trap will occur.
3. ACI is a privileged instruction.
4. The subaddress is not relevant for this instruction.
5. In the V6 IPU, if this instruction is executed for other than class 7, an undefined IPU trap occurs.

CONDITION CODE RESULTS

CC1	CC2	CC3	CC4	
0	0	0	0	Request activated, will echo status
0	0	0	1	Channel busy
0	0	1	0	Channel inoperable or undefined
0	0	1	1	Subchannel busy
0	1	0	0	Status stored
0	1	0	1	Unsupported transaction
0	1	1	0	Unassigned
0	1	1	1	Unassigned
1	0	0	0	Request accepted and queued, no echo status

ASSEMBLY LANGUAGE CODING

ACI R, '(Constant)'



830338

DEFINITION

The deactivate channel interrupt will cause the addressed channel to remove its interrupt level from contention. If a request interrupt is currently queued, the deactivate will cause the queued request to actively request if the channel is enabled.

Bits 0-5	Specify the operation code.
Bits 6-8	Specify a general purpose register; if this GPR is not R0, then its contents will be added to the channel and subaddress field to form the effective logical channel and subaddress.
Bits 9-12	Specify the operation as DACI.
Bits 13-15	Specify the augment code.
Bits 16-31	Specify the constant that will be added to the contents of the GPR specified by R to form the effective logical channel and subaddress. If R is zero, the constant alone will be used to specify the logical channel and subaddress.

NOTES

1. Condition codes after execution of the DACI will be set and can be tested by a subsequent conditional branch instruction to determine if the DACI was successfully executed.

2. In CPU, if this instruction is executed for other than a Class F I/O device channel, an undefined instruction trap will occur.

3. The DACI and following instruction are executed as an uninterruptible pair.

4. DACI is a privileged instruction.

5. The subaddress is not relevant for this instruction.

6. In the V6 IPU, if this instruction is executed for other than class 7, an undefined IPU trap occurs.

CONDITION CODE RESULTS

CC1	CC2	CC3	CC4	
0	0	0	0	Request activated, will echo status
0	0	0	1	Channel busy
0	0	1	0	Channel inoperable or undefined
0	0	1	1	Subchannel busy
0	1	0	0	Status stored
0	1	0	1	Unsupported transaction
0	1	1	0	Unassigned
0	1	1	1	Unassigned
1	0	0	0	Request accepted and queued, no echo status

ASSEMBLY LANGUAGE CODING

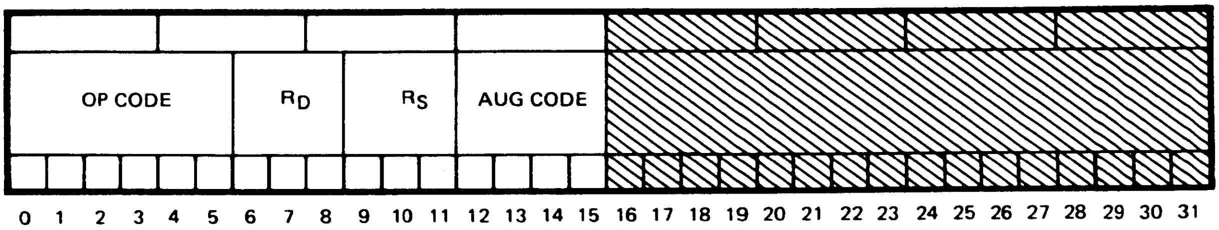
```
DACI R,'(Constant)'
```

6.2.17 Alterable Control Storage/Writable Control Storage Instructions (V6 ONLY)

Alterable control storage (ACS) comprises a 4K x 64 bit RAM bank which may be utilized to dynamically modify or 'patch' V6 CPU microcode. Writable control storage (WCS) is used with the V6 CPU only. WCS consists of two banks of 4K x 64 bit RAMs and is used to supplement firmware in the CPU.

6.2.17.1 Instruction Format

The V6 CPU associated ACS/WCS format is as follows:



830585

Bits 0-5 Define the operation code.

Bits 6-8 Varies in usage as follows:

Instruction	Usage
WWCS	Specifies the register containing the ACS/WCS address.
RWCS	Specifies the register containing the logical address in main memory that is to receive ACS/WCS contents.

Bit 9-11 Varies in usage as follows:

Instruction	Usage
WWCS	Specifies the register containing the logical address in main memory containing the information to be loaded into ACS/WCS.
RWCS	Specifies the register containing the ACS/WCS address.

Bits 12-15 Define the augmenting operation code.

Bits 16-31 Not used. This is a halfword instruction.

6.2.17.2 Condition Code

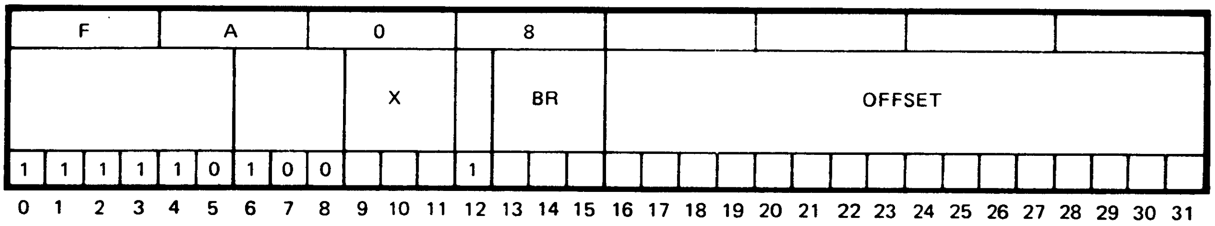
Condition codes remain unchanged for the RWCS and WWCS instructions, but may be changed by the JWCS instruction.

6.2.17.3 ACS/WCS Programming

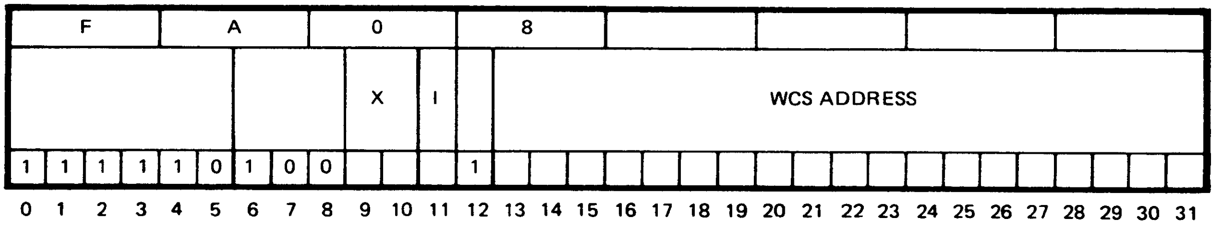
Programming the V6 CPU associated ACS/WCS is accomplished by the use of the Write WCS (WWCS) instruction. The contents of the WCS are in the form of microinstructions, which are used to augment the firmware in the V6 CPU. It is beyond the scope of this publication to provide the microinstruction techniques used in the implementation of WCS. The WCS is organized in 64 bits by 4K modules, allowing up to two modules to be used (8K x 64 bits).

Accessing the V6 CPU associated WCS is accomplished through the use of the Jump to WCS (JWCS) instruction. More complete information of the programming of the WCS is contained in the Writable Control Storage Users Manual 310-000980-000.

JUMP TO WRITABLE CONTROL STORAGE (V6 ONLY) JWCS
FA08 *m,x



BASE REGISTER FORMAT



NONBASE REGISTER FORMAT

830339

DEFINITION

This instruction causes an unconditional branch into WCS at the location specified by the effective WCS address.

The WCS effective address is determined after indirection and/or indexing have been resolved. The effective WCS address must be in the range of:

1000_H WCS_{EA} $2FFF_H$ (8K WCS Option)

If the effective WCS address is not within the above limits or if the WCS option is not present an address specification trap will occur. JWCS is an unprivileged instruction.

CONDITION CODE RESULTS

The condition codes may or may not be changed by the WCS microcode.

NOTE

JWCS is valid for V6 only.

READ WRITABLE CONTROL STORAGE (V6 ONLY) — RWCS

000B — **s,d**

0				0				0				B				▨				▨				▨				▨			
						R_D			R_S							▨															
0	0	0	0	0	0							1	0	1	1	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

830340

DEFINITION

The "Microword" in the PROM/ACS/WCS location specified by the address contained in R_S is accessed and transferred to the main memory location specified by the logical address contained in the GPR specified by R_D. The contents of R_D must be a logical word address that specifies the first word of a double word in main memory. F and C bits are ignored. The contents of R_S must be a valid PROM/ACS/WCS address in bits 16-31. If the PROM/ACS/WCS address specified by R_S bit 0=0, then the microword will be read from the PROM. If the PROM/ACS/WCS address is < 1000_H and R_S bit 0=1, then the microword will be read from ACS. If the PROM/ACS/WCS address is $\geq 1000_H$ then the microword will be read from WCS. Reads from PROM/ACS/WCS can be made while in PROM or ACS modes.

CONDITION CODE RESULTS

The condition codes are unchanged by this instruction.

NOTES

1. The WCS address specified in $R_{S_{16-31}}$ must be in the range of:

   $1000_H \leq R_{S_{16-31}} \leq 2FFF_H$

2. An address specification trap will occur if any of the following conditions exist:
   a. WCS option is not present.
   b. WCS target address is greater than 2FFF.
   c. WCS target address resides in a 4K bank that is marked non-present or inoperable.
   d. Memory address is not doubleword bound.

3. RWCS is an unprivileged instruction.

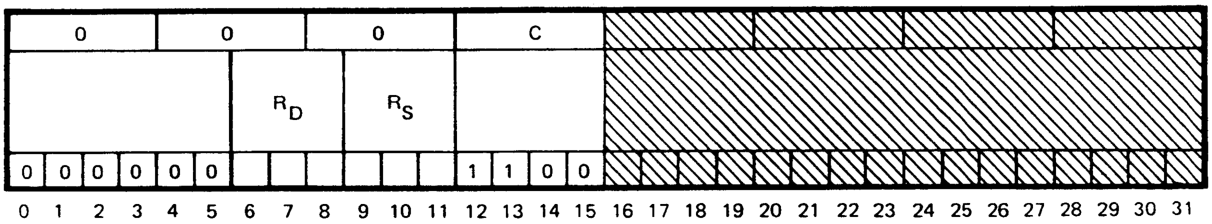
4. If $0 \leq R_{S16-31} \leq FFF_H$ and $R_{S0} = 0$ then read PROM at R_{S16-31}.

5. If $0 \leq R_{S16-31} \leq FFF_H$ and $R_{S0} = 1$ then read ACS at R_{S16-31}.

6. RWCS is valid for V6 only.

WRITE WRITABLE CONTROL STORAGE (V6 ONLY) 000C

WWCS s,d



830341

DEFINITION

The double word in memory specified by the logical address contained in the GPR specified by R_S is accessed and transferred to the ACS/WCS location specified by R_D. The contents of R_S must be a logical word address that specifies the first word of a double word. F and C bits are ignored. The contents of R_D must be a valid ACS/WCS location in bits 16-31. CPU Status Register bit 20 must be set (=1) in order to enable Writes to ACS or WCS. CPU Status Register bit 21 must be zero (=0), PROM mode, in order to write to ACS.

If the ACS/WCS location specified by R_D is $< 1000_H$, the Double word in Memory is transferred to ACS. If the ACS/WCS location specified by R_D is $\geq 1000_H$, the Double word in Memory is transferred to WCS.

CONDITION CODE RESULTS

The condition codes are unchanged by this instruction.

NOTES

1. Writes to ACS while in ACS or ROM Simulator Mode will generate a System Check Trap.

2. The WCS address specified in $R_{D_{16-31}}$ must be in the range of:

   $1000_H \leq R_{D_{16-31}} \leq 2FFF_H$

3. An address specification trap will occur if any of the following conditions exist:
   a. WCS option is not present.
   b. WCS target address is greater than 2FFF.
   c. WCS target address resides in a 4K bank that is marked non-present or inoperable.
   d. Memory address is not doubleword bound.

4. If $0 \leq R_{D16-31} \leq FFF_H$ then write to ACS.

5. If bit 20 of the CPU Status Register =0, an undefined instruction trap will occur.

6. WWCS is a privileged instruction.

7. WWCS is valid for V6 only.

APPENDIX A

INSTRUCTION SET
FUNCTIONALLY GROUPED BY SEQUENTIAL PAGE NUMBER

LOAD/STORE INSTRUCTIONS

Op Code	Mnemonic	Instruction	Page
AC08	LB	Load Byte	6-12
AC00	LH	Load Halfword	6-14
AC00	LW	Load Word	6-16
AC00	LD	Load Doubleword	6-18
B008	LMB	Load Masked Byte	6-20
B000	LMH	Load Masked Halfword	6-22
B000	LMW	Load Masked Word	6-24
B000	LMD	Load Masked Doubleword	6-26
B408	LNB	Load Negative Byte	6-28
B400	LNH	Load Negative Halfword	6-30
B400	LNW	Load Negative Word	6-32
B400	LND	Load Negative Doubleword	6-34
C800	LI	Load Immediate	6-36
D000 (NBR)	LEA	Load Effective Address	6-38
8000	LEAR	Load Effective Address Real	6-40
5000 (BR) 3400 (NBR)	LA	Load Address	6-42
5808 (BR)	LABR	Load Address Base Register	6-44
5800 (BR)	SUABR	Subtract Address Base Register	6-45
CC00	LF	Load File	6-46
CC08	LFBR	Load Base File	6-48
5C00 (BR)	LWBR	Load Base Register	6-50
D408	STB	Store Byte	6-52
D400	STH	Store Halfword	6-54
D400	STW	Store Word	6-56
D400	STD	Store Doubleword	6-58
D808	STMB	Store Masked Byte	6-60
D800	STMH	Store Masked Halfword	6-62
D800	STMW	Store Masked Word	6-64
D800	STMD	Store Masked Doubleword	6-66
DC00	STF	Store File	6-68
DC08	STFBR	Store Base File	6-70
5400 (BR)	STWBR	Store Base Register	6-72
F808	ZMB	Zero Memory Byte	6-74
F800	ZMH	Zero Memory Halfword	6-76
F800	ZMW	Zero Memory Word	6-78
F800	ZMD	Zero Memory Doubleword	6-80
0C00	ZR #	Zero Register	6-82

# Indicates halfword instruction
BR Base register
NBR Non base register

REGISTER TRANSFER INSTRUCTIONS

Op Code	Mnemonic	Instruction	Page
2C0F	TSCR # *	Transfer Scratchpad to Register	6-86
2C0E	TRSC # *	Transfer Register to Scratchpad	6-88
2C00	TRR #	Transfer Register to Register	6-90
2C08	TRRM #	Transfer Register to Register Masked	6-92
2C04	TRN #	Transfer Register Negative	6-94
2C0C	TRNM #	Transfer Register Negative Masked	6-96
2C03	TRC #	Transfer Register Complement	6-98
2C0B	TRCM #	Transfer Register Complement Masked	6-100
2C05	XCR #	Exchange Registers	6-102
2C0D	XCRM #	Exchange Registers Masked	6-104
2800	TRSW #	Transfer Register to PSD	6-106
2C02 (BR)	TBRR #	Transfer BR to GPR	6-108
2C01 (BR)	TRBR #	Transfer GPR to BR	6-109
2802 (BR)	XCBR #	Exchange Base Register	6-110
040B	RPSWT #	Read Processor Status Word Two (V6 Only)	6-112
040B	RPSWT #	Read Processor Status Word Two (V9 Only)	6-112B

MEMORY MANAGEMENT INSTRUCTIONS

000D (NBR)	SEA #	Set Extended Addressing	6-114
000F (NBR)	CEA #	Clear Extended Addressing	6-115
2C07	LMAP # *	Load Map	6-116
2C0A	TMAPR # *	Transfer Map to Register	6-117

BRANCH INSTRUCTIONS

EC00	BU	Branch Unconditionally	6-120
F000	BCF	Branch Condition False	6-122
EC00	BCT	Branch Condition True	6-124
F000	BFT	Branch Function True	6-126
F880	BL	Branch and Link	6-128
2808 (BR)	CALL #	Procedure Call	6-130
5C08 (BR)	CALLM	Procedure Call Memory	6-134
5C08 (BR)	BSUB	Branch Subroutine	6-138
2808 (BR)	BSUBM	Branch Subroutine Memory	6-142
280E (BR)	RETURN	Procedure Return	6-146
F400	BIB	Branch After Incrementing by a Byte	6-148
F420	BIH	Branch After Incrementing by a Halfword	6-150
F440	BIW	Branch After Incrementing by a Word	6-152
F460	BID	Branch After Incrementing by a Doubleword	6-154

COMPARE INSTRUCTIONS

9008	CAMB	Compare Arithmetic with Memory Byte	6-158
9000	CAMH	Compare Arithmetic with Memory Halfword	6-160

*	Indicates privileged instruction
#	Indicates halfword instruction
BR	Base register
NBR	Non base register

COMPARE INSTRUCTIONS (Continued)

Op Code	Mnemonic	Instruction	Page
9000	CAMW	Compare Arithmetic with Memory Word	6-162
9000	CAMD	Compare Arithmetic with Memory Doubleword	6-164
1000	CAR #	Compare Arithmetic with Register	6-166
C805	CI	Compare Immediate	6-168
9408	CMMB	Compare Masked with Memory Byte	6-170
9400	CMMH	Compare Masked with Memory Halfword	6-172
9400	CMMW	Compare Masked with Memory Word	6-174
9400	CMMD	Compare Masked with Memory Doubleword	6-176
1400	CMR #	Compare Masked with Register	6-178

LOGICAL INSTRUCTIONS

Op Code	Mnemonic	Instruction	Page
8408	ANMB	AND Memory Byte	6-182
8400	ANMH	AND Memory Halfword	6-184
8400	ANMW	AND Memory Word	6-186
8400	ANMD	AND Memory Doubleword	6-188
0400	ANR #	AND Register and Register	6-190
8808	ORMB	OR Memory Byte	6-192
8800	ORMH	OR Memory Halfword	6-194
8800	ORMW	OR Memory Word	6-196
8800	ORMD	OR Memory Doubleword	6-198
0800	ORR #	OR Register and Register	6-200
0808	ORRM #	OR Register and Register Masked	6-202
8C08	EOMB	Exclusive OR Memory Byte	6-204
8C00	EOMH	Exclusive OR Memory Halfword	6-206
8C00	EOMW	Exclusive OR Memory Word	6-208
8C00	EOMD	Exclusive OR Memory Doubleword	6-210
0C00	EOR #	Exclusive OR Register and Register	6-212
0C08	EORM #	Exclusive OR Register and Register Masked	6-214

SHIFT OPERATION INSTRUCTIONS

Op Code	Mnemonic	Instruction	Page
6000 (NBR)	NOR #	Normalize	6-218
6400 (NBR)	NORD #	Normalize Double	6-220
1008 (BR)	SACZ #	Shift and Count Zeros	6-222
6800 (NBR)	SCZ #		
1C40 (BR)	SLA #	Shift Left Arithmetic	6-224
6C40 (NBR)			
1C60 (BR)	SLL #	Shift Left Logical	6-226
7040 (NRB)			
2440 (BR)	SLC #	Shift Left Circular	6-228
7440 (NBR)			
2040 (BR)	SLAD #	Shift Left Arithmetic Double	6-230
7840 (NBR)			

# Indicates halfword instruction
BR Base register
NBR Non base register

SHIFT OPERATION INSTRUCTIONS (Continued)

Op Code		Mnemonic		Instruction	Page
2060 7C40	(BR) (NBR)	SLLD	#	Shift Left Logical Double	6-232
1C00 6C00	(BR) (NBR)	SRA	#	Shift Right Arithmetic	6-234
1C20 7000	(BR) (NBR)	SRL	#	Shift Right Logical	6-236
2400 7400	(BR) (NBR)	SRC	#	Shift Right Circular	6-238
2000 7800	(BR) (NBR)	SRAD	#	Shift Right Arithmetic Double	6-240
2020 7C00	(BR) (NBR)	SRLD	#	Shift Right Logical Double	6-242

BIT MANIPULATION INSTRUCTIONS

9808		SBM		Set Bit in Memory	6-246
1800		SBR	#	Set Bit in Register	6-248
9C08		ZBM		Zero Bit in Memory	6-250
1804 1C00	(BR) (NBR)	ZBR	#	Zero Bit in Register	6-252
A008		ABM		Add Bit in Memory	6-254
1808 2000	(BR) (NBR)	ABR	#	Add Bit in Register	6-256
A408		TBM		Test Bit in Memory	6-258
180C 2400	(BR) (NBR)	TBR	#	Test Bit in Register	6-260

FIXED-POINT ARITHMETIC INSTRUCTIONS

B808	ADMB		Add Memory Byte	6-264
B800	ADMH		Add Memory Halfword	6-266
B800	ADMW		Add Memory Word	6-268
B800	ADMD		Add Memory Doubleword	6-270
3800	ADR	#	Add Register to Register	6-272
3808	ADRM	#	Add Register to Register Masked	6-274
E808	ARMB		Add Register to Memory Byte	6-276
E800	ARMH		Add Register to Memory Halfword	6-278
E800	ARMW		Add Register to Memory Word	6-280
E800	ARMD		Add Register to Memory Doubleword	6-282
C801	ADI		Add Immediate	6-284
BC08	SUMB		Subtract Memory Byte	6-286
BC00	SUMH		Subtract Memory Halfword	6-288
BC00	SUMW		Subtract Memory Word	6-290
BC00	SUMD		Subtract Memory Doubleword	6-292
3C00	SUR	#	Subtract Register from Register	6-294
3C08	SURM	#	Subtract Register from Register Masked	6-296

# Indicates halfword instruction
BR Base register
NBR Non base register

FIXED-POINT ARITHMETIC INSTRUCTIONS (Continued)

Op Code	Mnemonic	Instruction	Page
C802	SUI	Subtract Immediate	6-298
C008	MPMB	Multiply by Memory Byte	6-300
C000	MPMH	Multiply by Memory Halfword	6-302
C000	MPMW	Multiply by Memory Word	6-304
3802 (BR) 4000 (NBR)	MPR #	Multiply Register by Register	6-306
C803	MPI	Multiply Immediate	6-308
C408	DVMB	Divide by Memory Byte	6-310
C400	DVMH	Divide by Memory Halfword	6-312
C400	DVMW	Divide by Memory Word	6-314
380A (BR) 4400 (NBR)	DVR #	Divide Register by Register	6-316
C804	DVI	Divide Immediate	6-318
0004	ES #	Extend Sign	6-320
0005	RND #	Round Register	6-322

FLOATING-POINT ARITHMETIC INSTRUCTIONS

E008	ADFW	Add Floating-Point Word	6-328
3801	ADRFW #	Add Floating-Point Word Register to Register	6-330
E008	ADFD	Add Floating-Point Doubleword	6-332
3809	ADRFD #	Add Floating-Point Doubleword Register to Register	6-334
E000	SUFW	Subtract Floating-Point Word	6-336
3803	SURFW #	Subtract Floating-Point Word Register to Register	6-338
E000	SUFD	Subtract Floating-Point Doubleword	6-340
380B	SURFD #	Subtract Floating-Point Doubleword Register to Register	6-342
E408	MPFW	Multiply Floating-Point Word	6-344
3806	MPRFW #	Multiply Floating-Point Word Register to Register	6-346
E408	MPFD	Multiply Floating-Point Doubleword	6-348
380E	MPRFD #	Multiply Floating-Point Doubleword Register to Register	6-350
E400	DVFW	Divide Floating-Point Word	6-352
3804	DVRFW #	Divide Floating-Point Word Register to Register	6-354
E400	DVFD	Divide Floating-Point Doubleword	6-356
380C	DVRFD #	Divide Floating-Point Doubleword Register to Register	6-358

#	Indicates halfword instruction
BR	Base register
NBR	Non base register

FLOATING-POINT CONVERSION INSTRUCTIONS

Op Code		Mnemonic			Instruction	Page
3807		FLTW	#		Float Integer Word to Floating-Point Word	6-360
380F		FLTD	#		Float Integer Doubleword to Floating-Point Doubleword	6-361
3805		FIXW	#		Fix Floating-Point Word to Integer Word	6-362
380D		FIXD	#		Fix Floating-Point Doubleword to Integer Doubleword	6-364

CONTROL INSTRUCTIONS

F980		LPSD		*	Load Program Status Doubleword	6-368
FA80		LPSDCM		*	Load Program Status Doubleword & Change Map	6-369
0003		LCS	#		Load Control Switches	6-370
C807		EXR			Execute Register	6-371
C807		EXRR			Execute Register Right	6-372
A800		EXM			Execute Memory	6-373
0000		HALT	#	*	Halt	6-374
0001		WAIT	#		Wait	6-375
0002		NOP	#		No Operation	6-376
C806		SVC			Supervisor Call	6-377
0009		RDSTS	#		Read CPU Status	6-378
2C09		SETCPU	#	*	Set CPU Mode	6-380
0008		EAE	#		Enable Arithmetic Exception Trap	6-381
000E		DAE	#		Disable Arithmetic Exception Trap	6-382
280C	(BR)	TPCBR	#		Transfer Program Counter to Base Register	6-383
2804	(BR)	TCCR	#		Transfer Condition Codes to GPR	6-384
2805	(BR)	TRCC	#		Transfer GPR to Condition Codes	6-385
040A		CMC	#	*	Cache Memory Control (V6 Only)	6-386
040A		CMC	#	*	Cache Memory Control (V9 Only)	6-387A
0407		SMC	#		Shared Memory Control (V6 Only)	6-388
000A		SIPU	#		Signal IPU	6-390

INTERRUPT CONTROL INSTRUCTIONS

FC00		EI		*	Enable Interrupt	6-394
FC02		RI		*	Request Interrupt	6-395
FC03		AI		*	Activate Interrupt	6-396
FC01		DI		*	Disable Interrupt	6-397
FC04		DAI		*	Deactivate Interrupt	6-398
0006		BEI	#	*	Block External Interrupts	6-399
0007		UEI	#	*	Unblock External Interrupts	6-400

INPUT/OUTPUT INSTRUCTIONS

FC06		CD		*	Command Device	6-402
FC05		TD		*	Test Device	6-403

*	Indicates privileged instruction
#	Indicates halfword instruction
BR	Base register

CLASS F I/O INSTRUCTIONS

Op Code	Mnemonic		Instruction	Page
FC17	SIO	*	Start I/O	6-406
FC1F	TIO	*	Test I/O	6-407
FC27	STPIO	*	Stop I/O	6-408
FC2F	RSCHNL	*	Reset Channel	6-409
FC37	HIO	*	Halt I/O	6-410
FC3F	GRIO	*	Grab Controller	6-411
FC47	RSCTL	*	Reset Controller	6-412
FC4F	ECWCS	*	Enable Channel WCS Load	6-414
FC5F	WCWCS	*	Write Channel WCS	6-415
FC67	ECI	*	Enable Channel Interrupt	6-417
FC6F	DCI	*	Disable Channel Interrupt	6-418
FC77	ACI	*	Activate Channel Interrupt	6-419
FC7F	DACI	*	Deactivate Channel Interrupt	6-420

WRITABLE CONTROL STORAGE INSTRUCTIONS

FA08	JWCS	*	Jump To Writable Control Storage (V6 Only)	6-424
000B	RWCS	# *	Read Writable Control Storage (V6 Only)	6-425
000C	WWCS	# *	Write Writable Control Storage (V6 Only)	6-426

Indicates halfword instruction
* Indicates privileged instruction

APPENDIX B

INSTRUCTION SET
GROUPED BY MNEMONIC
IN ALPHABETICAL ORDER

Mnemonic	Instruction	Execution Time (Microseconds)			Page
		V6	V6+FPA	V9	
ABM	Add Bit in Memory	.450+I1		.075+I1	6-254
ABR	Add Bit in Register	.300		.075	6-256
ACI	Activate Channel Interrupt	29.250			6-419
ADFD	Add Floating-Point Doubleword	3.600-15.90	1.500	.300	6-332
ADFW	Add Floating-Point Word	4.200-7.950	.950	.225	6-328
ADI	Add Immediate	.150		.075	6-284
ADMB	Add Memory Byte	.150+I1		.075	6-264
ADMD	Add Memory Doubleword	.300+I1		.150	6-270
ADMH	Add Memory Halfword	.150+I1		.075	6-266
ADMW	Add Memory Word	.150+I1		.075	6-268
ADR	Add Register to Register	.150		.075	6-272
ADRFD	Add Floating-Point Doubleword Register to Register	3.600-15.90	1.800	.375	6-334
ADRFW	Add Floating-Point Word Register to Register	4.200-7.950	1.050	.300	6-330
ADRM	Add Register to Register Masked	.300		.225	6-274
AI	Activate Interrupt	19.500		18.525	6-396
ANMB	AND Memory Byte	.450+I1		.225	6-182
ANMD	AND Memory Doubleword	.300+I1		.150	6-188
ANMH	AND Memory Halfword	.450+I1		.225	6-184
ANMW	AND Memory Word	.150+I1		.075	6-186
ANR	AND Register and Register	.150		.075	6-190
ARMB	Add Register to Memory Byte	.450+I1		.150+I1	6-276
ARMD	Add Register to Memory Doubleword	1.050+I1		.150+I1	6-282
ARMH	Add Register to Memory Halfword	.450+I1		.150+I1	6-278
ARMW	Add Register to Memory Word	.300+I1		.075+I1	6-280
BCF	Branch Condition False	.150+I2		.075+I2	6-122
BCT	Branch Condition True	.150+I2		.075+I2	6-124
BEI	Block External Interrupts	.300		.525	6-399
BFT	Branch Function True	3.000+I2		.150+I2	6-126
BIB	Branch After Incrementing by a Byte	.300+I2		.150+I2	6-148
BID	Branch After Incrementing by a Doubleword	.300+I2		.150+I2	6-154

* Most of the instruction times given are single case instruction times. They should not be considered best case, worst case, or typical (see notes on page B-7).

Mnemonic	Instruction	Execution Time (Microseconds) V6	 V6+FPA	 V9	Page
BIH	Branch After Incrementing by a Halfword	.300+I2		.150+I2	6-150
BIW	Branch After Incrementing by a Word	.300+I2		.150+I2	6-152
BL	Branch and Link	.750		.375	6-128
BSUB	Branch Subroutine	3.450		1.500	6-138
BSUBM	Branch Subroutine Memory	3.450			6-142
BU	Branch Unconditionally	.450		.225	6-120
CALL	Procedure Call	13.200		4.275	6-130
CALLM	Procedure Call Memory	13.200		4.750	6-134
CAMB	Compare Arithmetic with Memory Byte	.150+I1		.075	6-158
CAMD	Compare Arithmetic with Memory Doubleword	.300+I1		.075	6-164
CAMH	Compare Arithmetic with Memory Halfword	.150+I1		.075	6-160
CAMW	Compare Arithmetic with Memory Word	.150+I1		.075	6-162
CAR	Compare Arithmetic with Register	.150		.075	6-166
CD	Command Device	10.350-20.700		23.625	6-402
CEA	Clear Extended Addressing	1.500		.525	6-115
CI	Compare Immediate	.150		.075	6-168
CMC	Cache Memory Control (V6 Only)	4.200		---	6-386
CMC	Cache Memory Control (V9 Only)	---		600.000	6-387A
CMMB	Compare Masked with Memory Byte	.300		.150	6-170
CMMD	Compare Masked with Memory Doubleword	.600		.225	6-176
CMMH	Compare Masked with Memory Halfword	.300		.150	6-172
CMMW	Compare Masked with Memory Word	.300		.150	6-174
CMR	Compare Masked with Register	.300		.150	6-178
DACI	Deactivate Channel Interrupt	32.100		21.225	6-420
DAE	Disable Arithmetic Exception Trap	.450		.075	6-382
DAI	Deactivate Interrupt	21.150		7.125	6-398
DCI	Disable Channel Interrupt	25.650		34.275	6-418
DI	Disable Interrupt	15.450		10.725	6-397
DVFD	Divide Floating-Point Doubleword	36.150-37.80	8.100	3.075	6-356
DVFW	Divide Floating-Point Word	11.400-12.15	4.950	1.800	6-352
DVI	Divide Immediate	12.300-13.50		1.725	6-318
DVMB	Divide by Memory Byte	12.300-13.50		1.725	6-310
DVMH	Divide by Memory Halfword	12.300-13.50		1.725	6-312
DVMW	Divide by Memory Word	12.300-13.50		1.725	6-314
DVR	Divide Register by Register	3.000		1.725	6-316

* Most of the instruction times given are single case instruction times. They should not be considered best case, worst case, or typical (see notes on page B-7).

		Execution Time (Microseconds)			
Mnemonic	Instruction	V6	V6+FPA	V9	Page
DVRFD	Divide Floating Point Doubleword Register to Register	36.300-40.50	8.400	3.225	6-358
DVRFW	Divide Floating-Point Word Register to Register	11.400-12.15	4.500	1.950	6-354
EAE	Enable Arithmetic Exception Trap	.450		.075	6-381
ECI	Enable Channel Interrupt	24.600		33.375	6-417
ECWCS	Enable Channel WCS Load	8.400		.900	6-414
EI	Enable Interrupt	14.850		11.775	6-394
EOMB	Exclusive OR Memory Byte	.150+I1		.075	6-204
EOMD	Exclusive OR Memory Doubleword	.300+I1		.150	6-210
EOMH	Exclusive OR Memory Halfword	.150+I1		.075	6-206
EOMW	Exclusive OR Memory Word	.150+I1		.075	6-208
EOR	Exclusive OR Register and Register	.150		.075	6-212
EORM	Exclusive OR Register and Register Masked	.300		.150	6-214
ES	Extend Sign	.600		.150	6-320
EXM	Execute Memory	1.500		.0+I3	6-373
EXR	Execute Register	1.650		.225	6-371
EXRR	Execute Register Right	1.650		.225	6-372
FIXD	Fix Floating-Point Doubleword to Integer Doubleword	1.650-5.700		.525	6-364
FIXW	Fix Floating-Point Word to Integer Word	1.500-2.400		.450	6-362
FLTD	Float Integer Doubleword to Floating-Point Doubleword	1.950-9.450		.525	6-361
FLTW	Float Integer Word to Floating-Point Word	1.350-3.450		.300	6-360
GRIO	Grab Controller	21.000		28.800	6-411
HALT	Halt	.150		.075	6-374
HIO	Halt I/O	24.450		15.450	6-410
JWCS	Jump to Writable Control Store (V6 Only)	2.550		---	6-424
LA	Load Address	.150		.075	6-42
LABR	Load Address Base Register	.300+I1		.150+I1	6-44
LB	Load Byte	.150+I1		.075+I1	6-12
LCS	Load Control Switches	.900		.450	6-370
LD	Load Doubleword	.300+I1		.150	6-18
LEA	Load Effective Address	.150+I1		.075	6-38
LEAR	Load Effective Address Real	.300+I1		.300+I1	6-40
LF	Load File	(.600)N+.150		.450+.225(N-1)	6-46
LFBR	Load Base File	(.600)N-.150			6-48
LH	Load Halfword	.150+I1		.075	6-14
LI	Load Immediate	.150		.075	6-36

Most of the instruction times given are single case instruction times. They should not be considered best case, worst case, or typical (see notes on page B-7).

		Execution Time (Microseconds)			
Mnemonic	Instruction	V6	V6+FPA	V9	Page
LMAP	Load Map	218.550		43.5(M=256)	6-116
LMB	Load Masked Byte	.150+I1		.150	6-20
LMD	Load Masked Doubleword	.300+I1		.225	6-26
LMH	Load Masked Halfword	.150+I1		.150	6-22
LMW	Load Masked Word	.150+I1		.150	6-24
LNB	Load Negative Byte	.150+I1		.075	6-28
LND	Load Negative Doubleword	.300+I1		.150	6-34
LNH	Load Negative Halfword	.150+I1		.075	6-30
LNW	Load Negative Word	.150+I1		.075	6-32
LPSD	Load Program Status Doubleword	4.350		2.550	6-368
LPSDCM	Load Program Status Doubleword and Change Map	4.200-69.00		12.225(M=12)	6-369
LW	Load Word	.150+I1		.075	6-16
LWBR	Load Base Register	.150+I1		.150+I1	6-50
MPFD	Multiply Floating-Point Doubleword	15.000-25.20	2.550	1.725(.533*)	6-348
MPFW	Multiply Floating-Point Word	6.300-7.200	1.350	.900(.308*)	6-344
MPI	Multiply Immediate	6.450	1.800	.675(.173*)	6-308
MPMB	Multiply by Memory Byte	6.750	1.800	.525(.173*)	6-300
MPMH	Multiply by Memory Halfword	7.350	1.800	.675(.173*)	6-302
MPMW	Multiply by Memory Word	7.950	1.800	.900(.173*)	6-304
MPR	Multiply Register by Register	7.950	1.950	.900(.173*)	6-306
MPRFD	Multiply Floating-Point Doubleword-Register to Register	15.000-25.20	2.850	1.875(.533*)	6-350
MPRFW	Multiply Floating-Point Word Register to Register	6.300-7.200	1.500	.975(.308*)	6-346
NOP	No Operation	.150		.075	6-376
NOR	Normalize	1.200-6.450		.450	6-218
NORD	Normalize Double	1.500-15.00		.525	6-220
ORMB	OR Memory Byte	.150+I1		.075	6-192
ORMD	OR Memory Doubleword	.300+I1		.150	6-198
ORMH	OR Memory Halfword	.150+I1		.075	6-194
ORMW	OR Memory Word	.150+I1		.075	6-196
ORR	OR Register and Register	.150		.075	6-200
ORRM	OR Register and Register Masked	.300		.150	6-202
RDSTS	Read CPU Status	1.950		.900	6-378
RETURN	Procedure Return	5.400-8.100		3.375	6-146
RI	Request Interrupt	15.600		17.400	6-395
RND	Round Register	.900		.150	6-322
RPSWT	Read Processor Status Word Two (V6 Only)	1.050-1.350		---	6-112
RPSWT	Read Processor Status Word Two (V9 Only)	---		.225	6-112B

Most of the instruction times given are single case instruction times. They should not be considered best case, word case, or typical (see notes on page B-7).

* With multiply accelerator option installed.

		Execution Time (Microseconds)			
Mnemonic	Instruction	V6	V6+FPA	V9	Page
RSCHNL	Reset Channel			12.975	6-409
RSCTL	Reset Controller			28.975	6-412
RWCS	Read Writable Control Storage (V6 Only)	5.100		---	6-425
SACZ	Shift and Count Zeros	1.050+(.30)n		.375	6-222
SBM	Set Bit in Memory	.450+I1		.150	6-246
SBR	Set Bit in Register	.450		.150	6-248
SCZ	Shift and Count Zeros	1.050+(.30)n		.375	6-222
SEA	Set Extended Addressing	1.500		.525	6-114
SETCPU	Set CPU Mode	3.900		.450	6-380
SIO	Start I/O			18.000	6-406
SIPU	Signal IPU (in CPU)	.450		.225	6-390
	(in IPU)			7.275	6-390
SLA	Shift Left Arithmetic	(.150)n+.600		.075	6-224
SLAD	Shift Left Arithmetic Double	(.150)n+.900		.225	6-230
SLC	Shift Left Circular	(.150)n+.450		.075	6-228
SLL	Shift Left Logical	(.150)n+.450		.075	6-226
SLLD	Shift Left Logical Double	(.150)n+.600		.150	6-232
SMC	Shared Memory Control (V6 Only)	2.850		---	6-388
SRA	Shift Right Arithmetic	(.150)n+.450		.075	6-234
SRAD	Shift Right Arithmetic Double	(.150)n+.600		.150	6-240
SRC	Shift Right Circular	(.150)n+.450		.075	6-238
SRL	Shift Right Logical	(.150)n+.450		.075	6-236
SRLD	Shift Right Logical Double	(.150)n+.600		.150	6-242
STB	Store Byte	.300+I1		.075+I1	6-52
STFBR	Store Base File	(.300)n+.450		(.150)N+.225	6-70
STD	Store Doubleword	.900+I1		.150+I1	6-58
STF	Store File	(.600)N+.750		(.150)N+.225	6-68
STH	Store Halfword	.300+I1		.075+I1	6-54
STMB	Store Masked Byte	.300+I1		.150+I1	6-60
STMD	Store Masked Doubleword	.900+I1		.225+I1	6-66
STMH	Store Masked Halfword	.300+I1		.150+I1	6-62
STMW	Store Masked Word	.300+I1		.150+I1	6-64
STPIO	Stop I/O			12.525	6-408
STW	Store Word	.300+I1		.075+I1	6-56
STWBR	Store Base Register	.150+I1		.075+I1	6-72
SUABR	Subtract Address Base Register	.150+I1		.150+I1	6-45
SUFD	Subtract Floating-Point Doubleword	3.750-17.40	1.500	.300	6-340
SUFW	Subtract Floating-Point Word	4.350-7.500	.900	.225	6-336
SUI	Subtract Immediate	.150+I1		.075	6-298
SUMB	Subtract Memory Byte	.150+I1		.075	6-286
SUMD	Subtract Memory Doubleword	.300+I1		.150	6-292
SUMH	Subtract Memory Halfword	.150+I1		.075	6-288
SUMW	Subtract Memory Word	.150+I1		.075	6-290

Most of the instruction times given are single case instruction times. They should not be considered best case, worst cast, or typical (see notes on page B-7).

		Execution Time (Microseconds)			
Mnemonic	Instruction	V6	V6+FPA	V9	Page
SUR	Subtract Register from Register	.150		.075	6-294
SURFD	Subtract Floating-Point Doubleword Register to Register	3.750-17.40	1.800	.450	6-342
SURFW	Subtract Floating-Point Word Register to Register	4.350-7.500	1.050	.375	6-338
SURM	Subtract Register from Register Masked	.300		.225	6-296
SVC	Supervisor Call	9.750-74.55		13.275	6-377
TBM	Test Bit in Memory	.300		.075	6-258
TBR	Test Bit in Register	.300		.075	6-260
TBRR	Transfer BR to GPR	.150		.075	6-108
TCCR	Transfer Condition Codes to GPR	1.500		.300	6-384
TD	Test Device	21.000		12.975	6-403
TIO	Test I/O			21.825	6-407
TMAPR	Transfer Map to Register	25.500		.675	6-117
TPCBR	Transfer Program Counter to Base Register	5.400		.300+I1	6-383
TRBR	Transfer GPR to BR	.150+I1		.150+I1	6-109
TRC	Transfer Register Complement	.150		.075	6-98
TRCC	Transfer GPR to Condition Codes	1.150		.150	6-385
TRCM	Transfer Register Complement Masked	.300		.150	6-100
TRN	Transfer Register Negative	.150		.075	6-94
TRNM	Transfer Register Negative Masked	.300		.225	6-96
TRR	Transfer Register to Register	.150		.075	6-90
TRRM	Transfer Register to Register Masked	.300		.150	6-92
TRSC	Transfer Register to Scratchpad	.300		.225	6-88
TRSW	Transfer Register to PSD	.900		.300	6-106
TSCR	Transfer Scratchpad to Register	.450		.150	6-86
UEI	Unblock External Interrupts	.300		.525	6-400
WAIT	Wait				6-375
WCWCS	Write Channel WCS			14.850	6-415
WWCS	Write Writable Control Store				6-426
	(V6 Only)	5.400		---	6-426
XCR	Exchange Registers	.450		.150	6-102
XCBR	Exchange Base Registers	.450+I1		.225+I1	6-110
XCRM	Exchange Registers Masked	.600		.225	6-104
ZBM	Zero Bit in Memory	.600+I1		.150+I1	6-250
ZBR	Zero Bit in Register	.450		.150	6-252
ZMB	Zero Memory Byte	.300+I1		.075+I1	6-74
ZMD	Zero Memory Doubleword	.900+I1		.150+I1	6-80
ZMH	Zero Memory Halfword	.300+I1		.075+I1	6-76
ZMW	Zero Memory Word	.300+I1		.075+I1	6-78
ZR	Zero Register	.150		.075	6-82

* Most of the instruction times given are single case instruction times. They should not be considered best case, worst case, or typical (see notes on page B-7).

NOTES

1. In the V6, I1 (.150 usec) is the additional time required when the preceding instruction is a memory write. In the V9, I1 (.150 usec) is the additional time required when the next instruction is a full word instruction.

2. I2 (.300 usec fot V6 and .150 usec for V9) is the additional time required by the I unit to refill its instruction pipeline when the branch instruction is taken.

3. No additional time (I3) is required when the preceding instruction takes three or more cycles to be executed. When it takes one or two cycles to be executed, .150 usec and .075 usec of additional time are required respectively.

4. N indicates the number of registers loaded/stored.

5. M indicates the number of map registers loaded.

6. n indicates the number of bit shifts required.

APPENDIX C

INSTRUCTION SET GROUPED BY OP CODE IN HEXADECIMAL ORDER

Mnemonic		Instruction	Page	Op Code
HALT		Halt	6-374	0000
WAIT		Wait	6-375	0001
NOP		No Operation	6-376	0002
LCS		Load Control Switches	6-370	0003
ES		Extend Sign	6-320	0004
RND		Round Register	6-322	0005
BEI		Block External Interrupts	6-399	0006
UEI		Unblock External Interrupts	6-400	0007
EAE		Enable Arithmetic Exception Trap	6-381	0008
RDSTS		Read CPU Status	6-378	0009
SIPU		Signal IPU	6-390	000A
RWCS		Read Writable Control Store (V6 Only)	6-425	000B
WWCS		Write Writable Control Store (V6 Only)	6-426	000C
SEA	(NBR)	Set Extended Addressing	6-114	000D
DAE		Disable Arithmetic Exception Trap	6-382	000E
CEA	(NBR)	Clear Extended Addressing	6-115	000F
ANR		AND Register and Register	6-190	0400
SMC		Shared Memory Control (V6 Only)	6-388	0407
CMC		Cache Memory Control (V6 Only)	6-386	040A
CMC		Cache Memory Control (V9 Only)	6-387A	040A
RPSWT		Read Processor Status Word Two (V6 Only)	6-112	040B
RPSWT		Read Processor Status Word Two (V9 Only)	6-112B	040B
ORR		OR Register and Register	6-200	0800
ORRM		OR Register and Register Masked	6-202	0808
EOR		Exclusive OR Register and Register	6-212	0C00
ZR		Zero Register	6-82	0C00
EORM		Exclusive OR Register and Register Masked	6-214	0C08
CAR		Compare Arithmetic with Register	6-166	1000
SACZ	(BR)	Shift and Count Zeros	6-222	1008
CMR		Compare Masked with Register	6-178	1400
SBR		Set Bit in Register	6-248	1800
ZBR	(BR)	Zero Bit in Register	6-252	1804
ABR	(BR)	Add Bit in Register	6-256	1808
TBR	(BR)	Test Bit in Register	6-260	180C
SRA	(BR)	Shift Right Arithmetic	6-234	1C00
ZBR	(NBR)	Zero Bit in Register	6-252	1C00
SRL	(BR)	Shift Right Logical	6-236	1C20
SLA	(BR)	Shift Left Arithmetic	6-224	1C40
SLL	(BR)	Shift Left Logical	6-226	1C60
ABR	(NBR)	Add Bit in Register	6-256	2000
SRAD	(BR)	Shift Right Arithmetic Double	6-240	2000

BR Base Register
NBR Non-Base Register

Mnemonic	Instruction	Page	Op Code
SRLD (BR)	Shift Right Logical Double	6-242	2020
SLAD (BR)	Shift Left Arithmetic Double	6-230	2040
SLLD (BR)	Shift Left Logical Double	6-232	2060
SRC (BR)	Shift Right Circular	6-238	2400
TBR (NBR)	Test Bit in Register	6-260	2400
SLC (BR)	Shift Left Circular	6-228	2440
TRSW	Transfer Register to PSD	6-106	2800
XCBR (BR)	Exchange Base Registers	6-110	2802
TCCR (BR)	Transfer Condition Codes to GPR	6-384	2804
TRCC (BR)	Transfer GPR to Condition Codes	6-385	2805
BSUB (BR)	Branch Subroutine	6-138	2808
CALL (BR)	Procedure Call	6-130	2808
TPCBR (BR)	Transfer Program Counter to Base Register	6-383	280C
RETURN(BR)	Procedure Return	6-146	280E
TRR	Transfer Register to Register	6-90	2C00
TRDR (BR)	Transfer GPR to BR	6-109	2C01
TBRR (BR)	Transfer BR to GPR	6-108	2C02
TRC	Transfer Register Complement	6-98	2C03
TRN	Transfer Register Negative	6-94	2C04
XCR	Exchange Registers	6-102	2C05
LMAP	Load Map	6-116	2C07
TRRM	Transfer Register to Register Masked	6-92	2C08
SETCPU	Set CPU Mode	6-380	2C09
TMAPR	Transfer Map to Register	6-117	2C0A
TRCM	Transfer Register Complement Masked	6-100	2C0B
TRNM	Transfer Register Negative Masked	6-96	2C0C
XCRM	Exchange Registers Masked	6-104	2C0D
TRSC	Transfer Register to Scratchpad	6-88	2C0E
TSCR	Transfer Scratchpad to Register	6-86	2C0F
LA (NBR)	Load Address	6-42	3400
ADR	Add Register to Register	6-272	3800
ADRFW	Add Floating-Point Word Register to Register	6-330	3801
MPR (BR)	Multiply Register by Register	6-306	3802
SURFW	Subtract Floating-Point Word Register to Register	6-338	3803
DVRFW	Divide Floating-Point Word Register to Register	6-354	3804
FIXW	Fix Floating-Point Word Integer to Integer Word	6-362	3805
MPRFW	Multiply Floating-Point Word Register to Register	6-350	3806
FLTW	Float Integer Word to Floating-Point Word	6-360	3807
ADRM	Add Register to Register Masked	6-274	3808
ADRFD	Add Floating-Point Doubleword Register to Register	6-334	3809
DVR (BR)	Divide Register by Register	6-316	380A

BR Base Register
NBR Non-Base Register

Mnemonic	Instruction	Page	Op Code
SURFD	Subtract Floating-Point Doubleword Register to Register	6-342	380B
DVRFD	Divide Floating-Point Doubleword Register to Register	6-358	380C
FIXD	Fix Floating-Point Doubleword to Integer Doubleword	6-364	380D
MPRFD	Multiply Floating-Point Doubleword Register to Register	6-350	380E
FLTD	Float Integer Doubleword to Floating-Point Doubleword	6-361	380F
SUR	Subtract Register from Register	6-294	3C00
SURM	Subtract Register from Register Masked	6-296	3C08
MPR (NBR)	Multiply Register by Register	6-306	4000
DVR (NBR)	Divide Register by Register	6-316	4400
LA (BR)	Load Address	6-42	5000
STWBR (BR)	Store Base Register	6-72	5400
SUABR (BR)	Subtract Address Base Register	6-45	5800
LABR (BR)	Load Address Base Register	6-44	5808
LWBR (BR)	Load Base Register	6-50	5C00
BSUBM (BR)	Branch Subroutine Memory	6-142	5C08
CALLM (BR)	Procedure Call Memory	6-134	5C08
NOR (NBR)	Normalize	6-218	6000
NORD (NBR)	Normalize Double	6-220	6400
SCZ (NBR)	Shift and Count Zeros	6-222	6800
SRA (NBR)	Shift Right Arithmetic	6-234	6C00
SLA (NBR)	Shift Left Arithmetic	6-224	6C40
SRL (NBR)	Shift Right Logical	6-236	7000
SLL (NBR)	Shift Left Logical	6-226	7040
SRC (NBR)	Shift Right Circular	6-238	7400
SLC (NBR)	Shift Left Circular	6-228	7440
SRAD (NBR)	Shift Right Arithmetic Double	6-240	7800
SLAD (NBR)	Shift Left Arithmetic Double	6-230	7840
SRLD (NBR)	Shift Right Logical Double	6-242	7C00
SLLD (NBR)	Shift Left Logical Double	6-232	7C40
LEAR	Load Effective Address Real	6-40	8000
ANMH	AND Memory Halfword	6-184	8400
ANMW	AND Memory Word	6-186	8400
ANMD	AND Memory Doubleword	6-188	8400
ANMB	AND Memory Byte	6-182	8408
ORMH	OR Memory Halfword	6-194	8800
ORMW	OR Memory Word	6-196	8800
ORMD	OR Memory Doubleword	6-198	8800
ORMB	OR Memory Byte	6-192	8808
EOMH	Exclusive OR Memory Halfword	6-206	8C00
EOMW	Exclusive OR Memory Word	6-208	8C00
EOMD	Exclusive OR Memory Doubleword	6-210	8C00
EOMB	Exclusive OR Memory Byte	6-204	8C08
CAMH	Compare Arithmetic with Memory Halfword	6-160	9000

BR Base Register
NBR Non-Base Register

Mnemonic	Instruction	Page	Op Code
CAMW	Compare Arithmetic with Memory Word	6-162	9000
CAMD	Compare Arithmetic with Memory Doubleword	6-164	9000
CAMB	Compare Arithmetic with Memory Byte	6-158	9008
CMMH	Compare Masked with Memory Halfword	6-172	9400
CMMW	Compare Masked with Memory Word	6-174	9400
CMMD	Compare Masked with Memory Doubleword	6-176	9400
CMMB	Compare Masked with Memory Byte	6-170	9408
SBM	Set Bit in Memory	6-246	9808
ZBM	Zero Bit in Memory	6-250	9C08
ABM	Add Bit in Memory	6-254	A008
TBM	Test Bit in Memory	6-258	A408
EXM	Execute Memory	6-373	A800
LH	Load Halfword	6-14	AC00
LW	Load Word	6-16	AC00
LD	Load Doubleword	6-18	AC00
LB	Load Byte	6-12	AC08
LMH	Load Masked Halfword	6-22	B000
LMW	Load Masked Word	6-24	B000
LMD	Load Masked Doubleword	6-26	B000
LMB	Load Masked Byte	6-20	B008
LNH	Load Negative Halfword	6-30	B400
LNW	Load Negative Word	6-32	B400
LND	Load Negative Doubleword	6-34	B400
LNB	Load Negative Byte	6-28	B408
ADMH	Add Memory Halfword	6-266	B800
ADMW	Add Memory Word	6-268	B800
ADMD	Add Memory Doubleword	6-270	B800
ADMB	Add Memory Byte	6-264	B808
SUMH	Subtract Memory Halfword	6-288	BC00
SUMW	Subtract Memory Word	6-290	BC00
SUMD	Subtract Memory Doubleword	6-292	BC00
SUMB	Subtract Memory Byte	6-286	BC08
MPMH	Multiply by Memory Halfword	6-302	C000
MPMW	Multiply by Memory Word	6-304	C000
MPMB	Multiply by Memory Byte	6-300	C008
DVMH	Divide by Memory Halfword	6-312	C400
DVMW	Divide by Memory Word	6-314	C400
DVMB	Divide by Memory Byte	6-310	C408
LI	Load Immediate	6-36	C800
ADI	Add Immediate	6-284	C801
SUI	Subtract Immediate	6-298	C802
MPI	Multiply Immediate	6-308	C803
DVI	Divide Immediate	6-318	C804
CI	Compare Immediate	6-168	C805
SVC	Supervisor Call	6-377	C806
EXRR	Execute Register Right	6-372	C807
EXR	Execute Register	6-371	C807
LF	Load File	6-46	CC00

BR Base Register
NBR Non-Base Register

Mnemonic	Instruction	Page	Op Code
LFBR	Load Base File	6-48	CC08
LEA	Load Effective Address	6-38	D000
STH	Store Halfword	6-54	D400
STW	Store Word	6-56	D400
STD	Store Doubleword	6-58	D400
STB	Store Byte	6-52	D408
STMH	Store Masked Halfword	6-62	D800
STMW	Store Masked Word	6-64	D800
STMD	Store Masked Doubleword	6-66	D800
STMB	Store Masked Byte	6-60	D808
STF	Store File	6-68	DC00
STFBR	Store Base File	6-70	DC08
SUFW	Subtract Floating-Point Word	6-336	E000
SUFD	Subtract Floating-Point Doubleword	6-340	E000
ADFW	Add Floating-Point Word	6-328	E008
ADFD	Add Floating-Point Doubleword	6-332	E008
DVFW	Divide Floating-Point Word	6-352	E400
DVFD	Divide Floating-Point Doubleword	6-356	E400
MPFW	Multiply Floating-Point Word	6-344	E408
MPFD	Multiply Floating-Point Doubleword	6-348	E408
ARMH	Add Register to Memory Halfword	6-278	E800
ARMW	Add Register to Memory Word	6-280	E800
ARMD	Add Register to Memory Doubleword	6-282	E800
ARMB	Add Register to Memory Byte	6-276	E808
BU	Branch Unconditionally	6-120	EC00
BCT	Branch Condition True	6-124	EC00
BCF	Branch Condition False	6-122	F000
BFT	Branch Function True	6-126	F000
BIB	Branch After Incrementing Byte	6-148	F400
BIH	Branch After Incrementing Halfword	6-150	F420
BIW	Branch After Incrementing Word	6-152	F440
BID	Branch After Incrementing Doubleword	6-154	F460
ZMH	Zero Memory Halfword	6-76	F800
ZMW	Zero Memory Word	6-78	F800
ZMD	Zero Memory Doubleword	6-80	F800
ZMB	Zero Memory Byte	6-74	F808
BL	Branch and Link	6-128	F880
LPSD	Load Program Status Doubleword	6-368	F980
JWCS	Jump to Writable Control Store (V6 Only)	6-424	FA08
LPSDCM	Load Program Status Doubleword and Change Map	6-369	FA80
EI	Enable Interrupt	6-394	FC00
DI	Disable Interrupt	6-397	FC01
RI	Request Interrupt	6-395	FC02
AI	Activate Interrupt	6-396	FC03
DAI	Deactivate Interrupt	6-398	FC04
TD	Test Device	6-403	FC05
CD	Command Device	6-402	FC06
SIO	Start I/O	6-406	FC17
TIO	Test I/O	6-407	FC1F
STPIO	Stop I/O	6-408	FC27
RSCHNL	Reset Channel	6-409	FC2F

Mnemonic	Instruction	Page	Op Code
HIO	Halt I/O	6-410	FC37
GRIO	Grab Controller	6-411	FC3F
RSCTL	Reset Controller	6-412	FC47
ECWCS	Enable Channel WCS Load	6-414	FC4F
WCWCS	Write Channel WCS	6-415	FC5F
ECI	Enable Channel Interrupt	6-417	FC67
DCI	Disable Channel Interrupt	6-418	FC6F
ACI	Activate Channel Interrupt	6-419	FC77
DACI	Deactivate Channel Interrupt	6-420	FC7F

APPENDIX D

INSTRUCTION SET
V6 CPU COMPARED TO V9 CPU

V6 INSTRUCTIONS

The following instructions are valid for V6 only:

SMC	Shared Memory Control
JWCS	Jump to Writable Control Storage
RWCS	Read Writable Control Storage
WWCS	Write Writable Control Storage

SAME INSTRUCTION, DIFFERENT IMPLEMENTATION

The following instructions operate differently in the V6 and V9 CPUs:

RPSWT	Read Processor Status Word Two
CMC	Cache Memory Control

OTHER MAJOR DIFFERENCE:

Trap implementation

INSTRUCTION SET

V6 AND V9 COMPARED TO CONCEPT 32 CPUs

INSTRUCTION DIFFERENCES BETWEEN VARIOUS CPUs

32/67	32/87	32/97	V6	V9
SMC	n/a	n/a	SMC	n/a
JWCS	n/a	n/a	JWCS	n/a
RWCS	n/a	n/a	RWCS	n/a
WWCS	n/a	n/a	WWCS	n/a

V6 and V9 are fully supported in Base Mode only.

Gould Inc., Computer Systems Division
6901 W. Sunrise Blvd.
P. O. Box 409148
Fort Lauderdale, FL 33340-9148
Telephone (305) 587-2900

GOULD
Electronics

Users Group Membership Application

USER ORGANIZATION: ______________________________

REPRESENTATIVE(S): ______________________________

____

____

ADDRESS: ______________________________

____

TELEX NUMBER: ______________ PHONE NUMBER: ______________

NUMBER AND TYPE OF GOULD CSD COMPUTERS: ______________________________

____

OPERATING SYSTEM AND REV. LEVEL: ______________________________

____

APPLICATIONS (Please Indicate)

1. EDP
   - A. Inventory Control
   - B. Engineering & Production Data Control
   - C. Large Machine Off-Load
   - D. Remote Batch Terminal
   - E. Other

2. Communications
   - A. Telephone System Monitoring
   - B. Front End Processors
   - C. Message Switching
   - D. Other

3. Design & Drafting
   - A. Electrical
   - B. Mechanical
   - C. Architectural
   - D. Cartography
   - E. Image Processing
   - F. Other

4. Industrial Automation
   - A. Continuous Process Control Op.
   - B. Production Scheduling & Control
   - C. Process Planning
   - D. Numerical Control
   - E. Other

5. Laboratory and Computational
   - A. Seismic
   - B. Scientific Calculation
   - C. Experiment Monitoring
   - D. Mathematical Modeling
   - E. Signal Processing
   - F. Other

6. Energy Monitoring & Control
   - A. Power Generation
   - B. Power Distribution
   - C. Environmental Control
   - D. Meter Monitoring
   - E. Other

7. Simulation
   - A. Flight Simulators
   - B. Power Plant Simulators
   - C. Electronic Warfare
   - D. Other

8. Other

Please return to:

Users Group Representative

Date: ______________

Gould Inc., Computer Systems Division Users Group. . .

The purpose of the Gould CSD Users Group is to help create better User/User and User/Gould CSD communications.

There is no fee to join the Users Group. Simply complete the Membership Application on the reverse side and mail to the Users Group Representative. You will automatically receive Users Group Newsletters, Referral Guide and other pertinent Users Group activity information.

Fold and Staple for Mailing

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 947 FT. LAUDERDALE, FL

POSTAGE WILL BE PAID BY ADDRESSEE

GOULD INC., COMPUTER SYSTEMS DIVISION
ATTENTION: USERS GROUP REPRESENTATIVE
6901 W. SUNRISE BLVD.
P. O. BOX 409148
FT. LAUDERDALE FL 33340-9970

(Detach Here)

Fold and Staple for Mailing

GOULD
Electronics