Honeywell

MULTICS DIFFERENCES MANUAL

DPS 8 70/M

HONEYWELL CONFIDENTIAL & PROPRIETARY

This document and the information contained herein are confidential to and the property of Honeywell Information Systems, Inc. and are made available only to Honeywell employees for the sole purpose of conducting Honeywell's business. This document, any copy thereof and the information contained herein shall be maintained in strictest confidence; shall not be copied in whole or in part except as authorized by the employee's manager; and shall not be disclosed or distributed (a) to persons who are not Honeywell employees, or (b) to Honeywell employees for whom such information is not necessary in connection with their assigned responsibilities. Upon request, or when the employee in possession of this document no longer has need for the document for the authorized Honeywell purpose, this document and any copies thereof shall be returned to the employee's manager. There shall be no exceptions to the terms and conditions set forth herein except as authorized in writing by the responsible Honeywell Vice President.

58009997-040

A ISSUED
AUGUST 1983

WCPU68LA DIFFERENCE MANUAL

PREFACE

The purpose of this manual is to describe at a high level the hardware differences that are unique to the (WCPU68LA) DPS 8/70M processors relative to the current (WDPS68MA/SA;MB/SB) L68 processors. Because this manual describes the hardware differences at a high level it presumes that the reader has a thorough knowledge of the L68 processor. Some explanations may require the use of the Logic Block Diagrams.

This manual describes changes made to the CU and APU. These changes were made to reduce the processor board count and increase the processor speed.

These CU and APU changes resulted in minor changes to a few of the processor instructions.

The board count in the CU was reduced, a hardware controlled cache added and instruction buffer management changed. The board count in the APU was reduced and the PTW and SDW associative memories increased to 64 words each.

The following notice is provided in accordance with the United States Federal Communications Commission's (FCC) regulations.

Warning: This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instructions manual, may cause interference to radio communications. It has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the inteference.

HONEYWELL CONFIDENTIAL & PROPRIETARY

RECORD OF REVISIONS

REV	DATE	AUTHORIZATION	PAGES AFFECTED
A ISSUED	AUGUST 1983	PHAFPD951	58009997-040 58009997-014, 1F 58009997-033, 1F 58009997-001, 1F 58009997-017, 1 thru 4F 58009997,019, 1F 58009997, 1 thru 136F

TABLE OF CONTENTS

HONEYWELL CONFIDENTIAL & PROPRIETARY

MANUAL CONTENTS

MANUAL CONTENTS (CONT.)

HONEYWELL CONFIDENTIAL AND PROPRIETARY

MANUAL CONTENTS (CONT.)

LIST OF ILLUSTRATIONS

HONEYWELL CONFIDENTIAL AND PROPRIETARY

LIST OF ILLUSTRATIONS (CONT.)

LIST OF TABLES

HONEYWELL CONFIDENTIAL AND PROPRIETARY

1.0 INTRODUCTION

The purpose of this manual is to describe at a high level the hardware differences that are unique to the level (WCPU68LA) DPS 8/70M processors relative to current level (WDPS68MA/SA;MB/SB) L68 processors. This new 68/DPS 8 processor will support MULTICS and GCOS III in the emulation mode or GCOS III via manual switch setting.

1.1 PROCESSOR UNIT FUNCTIONS

The general major functions of each principal logical unit of the processor are listed below and the differences are described in subsequent sections of this manual.

1.1.1 OPERATIONS UNIT (OU)

This unit does-

- o Fixed and floating-binary operations
- o Shifting
- o Boolean operations

This unit is unchanged.

1.1.2 DECIMAL UNIT (DU)

This unit does-

- o Decimal arithmetic
- o Character and bit string operations

This unit is unchanged.

1.1.3 CONTROL UNIT (CU)

This unit does-

- o Address modification
- o Controls mode of operation (privileged, normal, etc.)
- o Interrupt recognition
- o Decodes instruction and indirect words

- o Controls timer register loading and decrementing
- o Contains cache memory

This unit has been considerably changed to reduce the MULTICS processor board count and to take advantage of the performance improvements made possible by cache memory.

The following characteristics change this unit.

- o Improved processor performance
- o Cache integral part of CU
- o Automatic cache clearing and 8K word cache size
- o Size and contents of history registers
- o Decrease of number of processor to SCU ports from 8 to 4
- o PROM to store ID codes and identify separately priced software
- o Maintenance panel interface

1.1.4 APPEND UNIT (APU)

This unit-

- o Controls data input/output to main memory
- o Performs main memory selection and interlace
- o Does address appending
- o Controls fault recognition
- o Interfaces with cache

The append unit has been changed to take advantage of the performance improvement made to the Control Unit and Associative Memory.

The following characteristics change this unit.

- o Improved processor performance
- o Size of history registers

HONEYWELL CONFIDENTIAL & PROPRIETARY

- o Decrease in number of CPU to SCU ports from 8 to 4
- o Configuration panel interface

1.1.5 ASSOCIATIVE MEMORY ASSEMBLY

The Associative Memory Assembly consists of a set of registers used to hold pointers to Page Table Words (PTW) and Segment Descriptor Words (SDW) that have recently been used to develop absolute address. The number of SDWAM and PTWAM registers have been increased to 64 from 16.

1.2 DEFINITIONS OF NOTATION AND SYMBOLS

The following notations and symbols are used in this manual:

1.2.1 MAIN MEMORY ADDRESSES

Y	=	an 18-bit computed address as generated during address preparation.
Y	=	a 24-bit main memory address of the instruction operand after all address preparation (including appending) is complete.
Y-pair	=	a pair of main memory locations with successive addresses, the smaller address being even. When Y is even, it designates the pair Y (even), Y+1; and when it is odd, the pair Y-1, Y (odd). The main memory location with the smaller (even) address contains the most significant part of a double-word operand or the first of a pair of instructions.
Y-blockn	=	a block of main memory locations of 4-, 8-, 16-, or 32-word extent. For a block of n-word extent, the processor forces Y-blockn to a 0 modulo n address and performs address incrementing through the block accordingly, stopping when the address next reaches a value 0 module n.
Y-charnk	=	a character or string of characters in main memory of character size n bits as described by the kth operand descriptor. n is specified by the data type field of operand descriptor k and may have values 4, 6, or 9.

Y-bit*k*	=	a bit or string of bits in main memory as described by the *k*th operand descriptor.

1.2.2 INDEX VALUES

When reference is made to the elements of a string of characters or bits in main memory, the notation shown in "Register Position and Contents" below is used. The index used to show traversing a string of extent *n* may take any of the values in the interval (1,*n*) unless noted otherwise. The elements of a main memory block are traversed explicitly by using the index as an addend to the given block address, (e.g.. Y-block8+m and Y-block4+2m+1).

1.2.3 ABBREVIATIONS AND SYMBOLS

A	Accumulator register
ARn	Address register n (n = 0, 1, 2, ..., 7)
AQ	Combined accumulator-quotient register
BAR	Base address register
C()	"Contents of"
CA	Computed address
DSBR	Descriptor segment base register
DSBR.ADDR	Address field of DSBR
DSBR.BND	Bound field of DSBR
DSBR.STACK	Stack base field of DSBR
DSBR.U	Unpaged flag of DSBR
E	Exponent register
EA	Combined exponent-accumulator register
EAQ	Combined exponent-accumulator-quotient register
ERN	Effective ring number
ESN	Effective segment number
FCT	Fault counter-counts instruction
HOLD	Holding register
IC	Instruction counter
IR	Indicator register
PPR	Procedure pointer register
PPR.PRR	Procedure ring register of PPR
PPR.PSR	Procedure segment register of PPR
PPR.IC	Instruction counter register of PPR (same as IC above)
PPR.P	Privileged flag of PPR
PRn	Pointer register n (n = 0, 1, 2, ..., 7)
PRn.RNR	Ring number register of PRn
PRn.SNR	Segment number register of PRn
PRn.WORDNO	Word address register of PRn
PRn.CHAR	Character address register of PRn
PRn.BITNO	Bit offset register of PRn
PRR	Procedure ring register
PSR	Procedure segment register

HONEYWELL CONFIDENTIAL & PROPRIETARY

Q	Quotient register
PTWAM	Page table word associative memory
SDWAM	Segment descriptor word associative memory
RALR	Ring alarm register
TPR	Temporary pointer register
TPR.CA	Computer address register of TPR (same as CA above)
TPR.TRR	Temporary ring register of TPR
TPR.TSR	Temporary segment register of TPR
TPR.TBR	Temporary bit register of TPR
TR	Timer register
TRR	Temporary ring register
Xn	Index register n (n = 0, 1, 2, ..., 7)
Z	Temporary pseudo-result of a nonstore comparative operation

1.2.4 REGISTER POSITIONS AND CONTENTS

In the definitions that follow, "R" stands for any of the registers listed above as well as for main memory words, word-pairs, word-blocks, and bit- or character-strings.

R_i	The ith bit, character, or byte position of R
R(i)	The ith register of a set of n registers named R
$R_{i,j}$	The bit, character or byte positions i through j or R
C(R)	The contents of the full register R
C(R)	The contents of the ith bit, character, or byte of R
$C(R)_{i,j}$	The contents of the bits, characters, or bytes i through j of R
xx...x	A string of binary bits (0's or 1's) of any necessary length

When the description of an instruction specifies a change for a part of a register or main memory location, it is understood that the part of the register or main memory location not mentioned remains unchanged.

1.2.5 OTHER SYMBOLS

->	replaces
::	compare with

&	the Boolean connective AND
:	the Boolean connective OR
$\oplus$	the Boolean connective NON-EQUIVALENCE (or EXCLUSIVE OR)
$\overline{XXX}$	the logical inverse (ones complement) of the quantity XXX
$\neq$	not equal
n**m	indicates an exponent (n and m are integers); for example, the fifth power of 2 is represented as 2**5.
X	multiplication; for example, C(Y) times C(Q) is represented as C(Y) X C(Q).
/	division; for example, C(Y) divided by C(A) is represented as C(Y) / C(A).
::	concatenation; for example, string 1:: string 2.
....	the absolute value of the value between vertical bars (no algebraic sign). For example the absolute, value of C(A) plus C(Y) is represented as: C(A) + C(Y) .
$C(R)_{mod\underline{n}}$	A coined notation for remaindering or modulo arithmetic; for example C(REG) modulo 9 is represented as $C(REG)_{mod9}$.

2.0 DIFFERENCES UNIQUE TO DPS 8 PROCESSORS

The hardware for all four processor ports is built into and is functional in all processors. In pre-DPS 8 processors, the port boards were separately-installable options.

The 8K word cache is built into all processors.

When operating in GCOS III mode (mode switch set for GCOS), the system is functionally identical to Level 66/DPS 8 (NON NSA).

3.0 DIFFERENCES UNIQUE TO LEVEL 68/DPS 8

This section lists differences from MULTICS systems which are unique to the Level 68/DPS 8 system.

HONEYWELL CONFIDENTIAL & PROPRIETARY

3.1 PROCESSOR NUMBERS, PROCESSOR TO SCU PORTS

The maximum number of processors designated per system is extended from 4 to 8. Processor numbers will be designated by three bits. The maximum number of SCU's per system is reduced from 8 to 4. The previous recognition of processor ports E, F, G, and H is dropped.

Several instructions which specified the processor port selection by $C(TPR.CA)_{1-2}$(see Section 1.2, Definitions of Notations and Symbols) will now ignore bit position 0. Port selection for L68/DPS 8 is specified by $C(TPR.CA)_{1-2}$. The affected instructions are:

Op Code	Mnemonic	Name
633(0)	rccl	Read Calendar Clock
233(0)	rmcm	Read Memory Controller Mask Register
451(0)	smic	Set Memory Controller Interrupt Cells
553(0)	smcm	Set Memory Controller Mask Register

Processor number will occupy three bits in the control unit data, bit positions 27-29 of word 2, rather than two bits. The affected instructions are:

Op Code	Mnemonic	Name
657(0)	scu	Store Control Unit
513(0)	rcu	Restore Control Unit

The reduction in the number of processor ports and the rearrangement of data fields in read switches instruction allows elimination of instructions rsw3 and rsw4.

3.2 ASSOCIATIVE MEMORIES

The Segment Descriptor Word and Page Table Word associative memories are extended to 64 words from 16 words. The 64 word associative memories are organized as four level set-associative with LRU replacement.

Existing T&D instructions which loaded the associative memories are deleted. They are not required for the new T&D tests. The deleted instructions are:

Op Code	Mnemonic	Name
257(1)	lptp	Load Page Table Pointers
172(1)	lptr	Load Page Table Registers
257(0)	lsdp	Load Segment Descriptor Pointers
232(1)	lsdr	Load Segment Descriptor Registers

The T&D instructions which stored contents of the associative memory are modified to accommodate the increase of size to 64 words. These are:

Op Code	Mnemonic	Name
557(1)	sptp	Store Page Table Pointers
154(1)	sptr	Store Page Table Registers
557(0)	ssdp	Store Segment Descriptor Pointers
254(0)	ssdr	Store Segment Descriptor Registers

3.3 HARDWARE-CONTROLLED CACHE

3.3.1 SUMMARY

The Level 68/DPS 8 processor has an 8K word hardware controlled cache. The design eliminates the need for certain cache clears and cache bypass modes that were required in the previous 2K word software controlled cache.

The requirement for coexistence of the 2K cache with the 8K cache in a multiprocessor system has led to the introduction of an additional cache bypass control. This is provided in the processor's cache mode register. This will permit coexistence without requiring a change in the 2K cache processor or restricting the types of multiple memory connections.

A synchronizing function, whose purpose is to ensure integrity of gated shared data, is provided in the instruction repertoire. This function is added to the stc2 and stacq instructions. This addition is needed to support 8K cache processors in multiprocessor, multi-SCU systems.

An additional performance improvement involves the group of read-alter-rewrite instructions. These will no longer automatically bypass cache, as in the 2K cache processor, unless a cache bypass mode is in effect. This cache mode is totally controlled by the cache bypass bit, SDW.C, in the segment descriptor.

Full utilization of the performance capabilities require changes to MULTICS software. Among these are replacement of open gate instructions with stc2 or stacq, and changes to SDW.C to specify cache use instead of bypass. Software considerations must also be given to the 2K-8K coexistence, and the eventual phasing out of this requirement as migration from 2K to 8K cache processors occurs.

3.3.2 DETAILS

The cache bypass option in the segment descriptor word is retained. An overriding bypass enable, bit 68 of the cache mode register is added. The cache mode is set as follows:

SDW.C	CMR	RESULTANT CACHE MODE
Use Cache	X	Use Cache
Bypass Cache	Bypass Cache	Bypass Cache
Bypass Cache	Use Cache	Use Cache

All close gate instructions, ldac, ldqc, stac, stacq, and sznc will automatically bypass cache. Two features are added to ensure integrity of gated shared data; one is added during the close gate operation and the other during the open gate operation. The instruction following the close gate instruction will bypass cache if the instruction is a read or a read-alter-rewrite. The open gate operation must be performed with either a stc2 or stacq which includes the synchronizing function. The synchronizing function forces the processor to delay the open gate operation until it is notified by the SCU that write completes have occurred and write notifications requesting cache block clears have been sent to the other processors for all write instructions that the processor has previously issued.

Read-alter-rewrite instructions will no longer automatically bypass cache. Cache behavior for these instructions is determined fully by SDW.C. If the bypass cache mode is set, these instructions will bypass cache and issue read-lock-write-unlock commands to memory. If a cache directory match occurs, the location is cleared.

All accesses to memory by SDW and PTW associative memory hardware will continue to bypass cache. Operations used will be reads for SDWs, read-alter-rewrites with lock for PTWs and setting the page used bit, and writes for setting the page modified and used bits. For writes, the hardware will also disable the key line so that the SCU lock is honored. This is consistent with dynamic PTW modification by software which also bypasses cache and uses read-alter-rewrite instructions.

The instructions which cleared the associative memories and also cleared cache or selective portions of cache are changed to eliminate the cache clear function. Bit C $(TPR.CA)_{15}$ is ignored. These instructions will also include disable/enable capabilities for each quarter of the associative memories.

The associative memory replacement is LRU; (Least Recently Used); this is also reset upon a camp or cams so that repeatable behavior occurs under test conditions. Cache mode register bit 56, which had previously controlled cache bypass for operands is disregarded. All other cache control bits are continued. However, maintenance panel cache control function is restricted to cache half enable/disable functions.

The cache memory replacement method is also LRU.

3.4 HISTORY REGISTERS

On prior Level 66 systems, the processor history registers were 16 instructions or steps deep. This has proved to be less than desired. These registers are now implemented 64 steps deep in the DPS 8 CPU. The OU and DU registers are combined into a 72-bit register which is pulled in a single access. Many of the OU and DU bits were rarely used and were eliminated.

As an aid to software debug, a capability for history register strobe on "transfer" was added.

In order to read the history registers and place them in the store, the scpr instruction must be used. The TAG field of the scpr indicates which of the 64 register groups will be stored, the CU group, the OU/DU group, the APU group. In order to store a full group of 64 registers, an scpr instruction must be executed by the processor 64 times in succession.

The TAG field code is shown below:

Select CU History Register - TAG field = 010000 $(20)_8$
Select OU/DU History Register - TAG field = 100000 $(40)_8$
Select APU History Register - TAG field = 000000 $(00)_8$

The lcpr instruction with the tags of 03_8 and 07_8 shall load all four groups of history registers with zero's and one's, respectively.

3.4.1 CU HISTORY REGISTER FORMAT

Bit	Field	Function
00	FPIA	Prepare Instruction Address
01	FPOA	Prepare Operand Address
02	DRIW	Request Indirect Word
03	FSIW	Restore Indirect Word
04	FPOT	Prepare Operand Tally
05	FPON	Prepare Operand Next
06	DRAW	Read-Alter Word
07	FSAW	Store Read-Alter Word
08		Remember TRA/GO
09	XDE	Execute Double from Even
10	XDO	Execute Double from Odd
11	FIC	Odd Instruction from Current Pair
12	RPTS	Repeat Flag
13	PORTF	Memory Cycle Went to Port
14	MATCHF	Memory Cycle Went to Cache/Direct
15	XIP/ADR	Prepare Interrupt Address
16	FLT/ADR	Prepare Fault Address
17	MASTER MODE	Processor in Master Mode
18-35	Op code and Tag A copy of bits 18-35 of the instruction being executed.	
36-59	24-Bit Real Memory Address	
60-64	Command Register A copy of the Processor Command Register (Command Lines A-E)	
65	FXEC-INT	Execute Interrupt Cycle
66	RB-INS-FETCH	Instruction Fetch
67	RB-CACHE-RD	Cache Read
68	RB-NONMCACHE	Memory Read
69	RB-STORE	Store
70	PC-BUSY	Port Busy

3.4.2 OU/DU HISTORY REGISTERS FORMAT

Bit	Field	Function	
00	FANDL1	Alpha-Numeric Load Desc. 1	C
01	FANDL2	Alpha-Numeric Load Desc. 2	C
02	FANSTR	Alpha-Numeric Store	C

03	FLDWRT1	Load-Rewrite Register 1	C
04	FLDWRT2	Load-Rewrite Register 2	C
05	FNLD1	Numeric Load Desc. 1	C
06	FNLD2	Numeric Load Desc. 2	C
07	FEND-SEQ	End Sequence Flag	
08	FDUD	Decimal Unit Idle	C
09	FGSTR	General Store	C
10	DEND-SEQ	End of Sequence	C
11	(NINE	9-Bit Character Operation	
12	(SIX	6-Bit Character Operation	
13	(FOUR	4-Bit Character Operation	
14	(BIT	Bit Operation	
15	(WORD	Word Operation	
16	(PTRA00	Select PTR 1	
17	(PTRA01	Select PTR 2	
18	(PTRA02	Select PTR 3	
19	FPOP	Prepare Operand Pointer	
20	GFGEAA-M		C
21	FGLDP1-2	Load Pointer 1+2	C
22	FGEMA-E	Multiply Gates	C
23	FGBDABC	Binary to Decimal Gates	C
24	FGSP1-5	Align Cycles	C
25	FSWEQ		C
26	FGCH	Character Cycle	C
27	DFRST		
28	(EXH	Exhaust	
29	FGADD	Add Cycle	C
30	FINTRPTD	Interrupted	C
31	DCODE 0		
32	1		
33	2		
34	3		
35	4		

36-53 A copy of the Instruction Counter
54-62 A copy of OU Op Code Register - (RS0 - RS8)

63	ZERO	Indicator Register
64	SIGN	Indicator Register
65	CARRY	Indicator Register
66	OVFL	Indicator Register
67	EOVFL	Indicator Register
68	EUFL	Indicator Register
69	OFLM	Indicator Register
70	HEX MODE	Indicator Register
71	DTRGO	Transfer Go

C - Data Stored in Complement Form

3.4.3 APPENDING UNIT HISTORY REGISTER (APUHR) FORMAT

The contents and number of the APUHRs remain the same, the contents are repeated here for convenience. The APUHRs are handled as a rotating queue controlled by the appending unit history register counter. The counter is always set to the number of the oldest entry and advances by one for each history register reference (data entry or scpr instruction).

The appending unit history register shows the condition in the appending unit at the end of an address preparation cycle in append mode. The registers will hold the conditions for the last 16 such address preparation cycles. Entries are made according to controls set in the mode register.

Bit	Field	Function
History Register #1		
0-14	ESN	Effective segment number (TPR.TSR)
15	PIAPGBSY	Instruction fetch across a page boundary
16	PIAOOSB	Instruction fetch cycle out of segment boundary
17	FDSPTW	Descriptor segment PTW fetch
18	MDSPTW	Descriptor segment PTW modification
19	FSDWP	SDW fetch from paged descriptor segment
20	FPTW	PTW fetch
21	FPTW2	PTW+1 fetch (prepaging for certain EIS instructions)
22	MPTW	PTW modification
23	FANP	Final address fetch from nonpaged segment
24	FAP	Final address fetch from paged segment
25	SDWAMM	SDWAM match occurred
26	SDWMF	SDWAM match occurred and used
27-28	BSY	Data Source for ESN 00 = from PPR.PSR 01 = from PRn.TSR 10 = from TPR.SNR 11 = from TPR.CA
29	PTWMF	PTWAM match occurred and used
30	MTCHPTW	PTWAM match occurred
31-34	PTWADDR	$(TPR.CA)_{4,7}$ Address select for PTWAM
35	FLT	Access violation or directed fault on this cycle

36-59	ADD	24-bit absolute main memory address from this cycle
60-62	TRR	Ring number from this cycle (TPR.TRR)
63	SDWERR	Multiple match or parity error in SDWAM
64-65	SDWLVL	SDWAM Level selected 00 = A 01 = B 10 = C 11 = D
66	CA	Cache used on this cycle
67	PTWERR	Multiple match or parity error in PTWAM
68-69	PTWLVL	PTWAM Level selected 00 = A 01 = B 10 = C 11 = D
70	FHLD	Access violation or directed fault is waiting

History Register #2

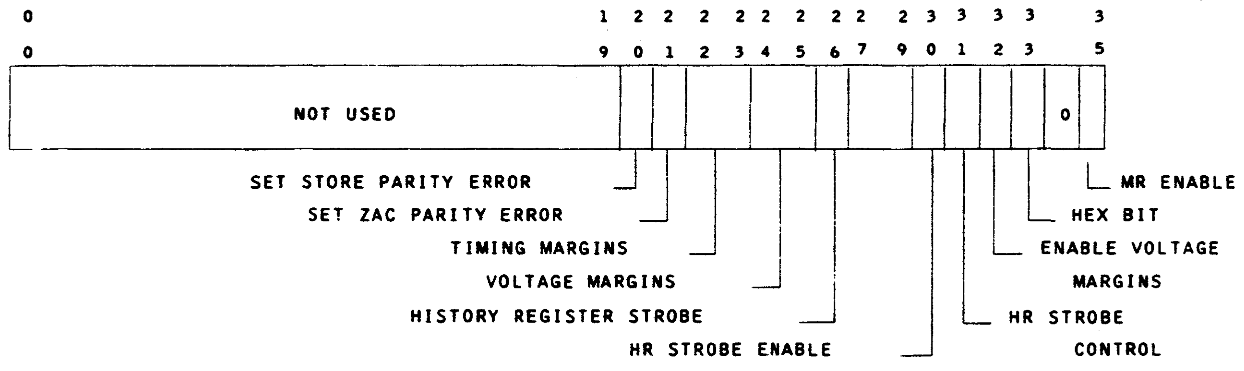
00-17	ADDR	Computed address, TPR.CA
18-27	OPCODE	Operation code from current instruction word.
28	I	Interrupt inhibit bit from current instruction word.
29	P	Pointer register flag bit from current instruction word
30-35	TAG	Current address modifier. This field is replaced by TAG field of indirect words as they are fetched during indirect chains.
36-71		Not used.

3.5 MODE SWITCH

The mode switch on Level 68/DPS 8 has two positions, for GCOS mode and MULTICS mode, instead of three. The absolute mode is eliminated. The mode switch is customer-available.

3.6 MODE REGISTER (MR) FORMAT

Even word of Y-pair as stored by Store Central Processor Register (scpr), TAG = 06.



An assemblage of flags and registers from the control unit. The mode register and the cache mode register are both stored into the Y-pair by the Store Central Processor Register (scpr), TAG = 06. The mode register is loaded with the Load Central Processor Register (lcpr), TAG = 04, instruction.

The mode register controls the operation of those features of the processor that are capable of being enabled and disabled. The functions of the constituent flags and registers are:

Bit	Field	Function
0-19		Unassigned
20	SET STORE PARITY ERROR	Causes incorrect data parity to be sent to SCU for next Store instruction, reset bit 20
21	SET ZAC PARITY ERROR	Causes incorrect ZAC parity to be sent to SCU for next Store instruction, reset bits 20
22-23	TIMING MARGINS	Set timing margins if set ON. If VOLT (bit 32) is set ON and the margin control switch on the processor maintenance panel is in PROG position, set processor timing margins as follows:

22,23	Margin
0,0	normal
0,1	slow
1,0	normal
1,1	fast

24-25	VOLTAGE MARGINS	Set +5 voltage margins if set ON. If VOLT (bit 32) is set ON and the margin control switch on the processor maintenance panel is in the PROG position, set +5 voltage margins as follows:

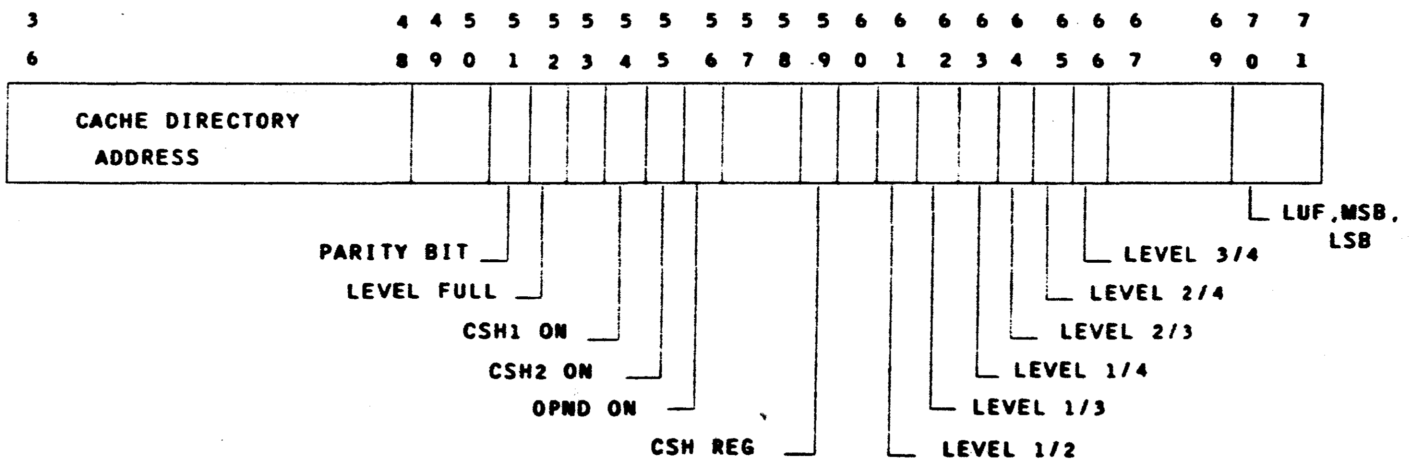
24,25	Margin
0,0	normal
0,1	slow
1,0	high
1,1	normal

26	HISTORY REGISTER STROBE	If reset, history register is normally strobed. If set, the history register is strobed on transfers made.
27-29		Unassigned
30	HR STROBE ENABLE	Enable strobe for CU, OU, VU and DU history registers.
31	HR STROBE CONTROL	Controls reset of bit 30. Bit 31 = 0 Reset bit 30. On ONC fault only. Bit 31 = 1 Reset bit 30. All faults will reset bit 30, lock history registers and leave cache on except ONC, LOCKUP, IPR, Parity, Command, Store or Shutdown.
32	ENABLE VOLTAGE MARGINS	Control ability of software to set voltage margins. Test mode indicator. This bit is set ON whenever the TEST/NORMAL switch on the processor maintenance panel is in TEST position and is set OFF otherwise. It serves to enable the program control of voltage and timing margins.
33	HEX BIT	Hexadecimal exponent permission bit.

34		Unassigned
35	MR ENABLE	Enable mode register. When this bit is set ON, all other bits and controls of the mode register are active. When this bit is set OFF, the mode register controls are disabled.

3.7 CACHE MODE REGISTER (CMR) FORMAT

Odd word of Y-Pair as stored by Store Central Processor Register (scpr), TAG = 06.



An assemblage of flags and registers from the control unit. The mode register and cache mode register are both stored into the Y-pair by the Store Central Processor Register (scpr), TAG = 06, instruction. The cache mode register is loaded with the Load Central Processor Register (lcpr), TAG = 02, instruction.

The data stored from the cache mode register is address dependent. The algorithm used to map main memory into the cache memory is effective for the Store Central Processor Register (scpr) instruction. In general, the user may read out data from the directory entry for any cache memory block by proper selection of certain subfields in the 24-bit absolute main memory address. In particular, the user may read out the directory entry for the cache memory block involved in a suspected cache memory error by assuring that the required 24-bit absolute main memory address subfields are the same as those for the access which produced the suspected error.

The fault handling procedure(s) should be unencacheable (SDW.C = 0) and the history registers and cache memory should be disabled as quickly as possible in order that vital information concerning the suspected error not be lost.

The cache mode register provides configuration information and software control over the operation of the cache memory. Those items following with an "x" in the column headed L are not loaded by the Load Central Processor Register (lcpr), TAG = 02, instruction.

The functions of the constituent flags and registers are:

Bit	L	Field	Function
36-48	x	CACHE DIR ADDRESS	13 high-order bits of the cache memory block address from the cache directory (on GCOS III, E_0-E_5, A_0-A_6)
49-50			Unassigned
51	x...	PAR BIT	Cache memory directory parity bit
52	x	LEV FUL	The selected column and level is full/empty
53			Unassigned
54		CSH1 ON	Enable the upper 4096 words of the cache memory.
55		CAH2 ON	Enable the lower 4096 words of the cache memory.
56		OPND ON	Enable the cache memory for operands.
57-58	x		Unassigned
59		CASH REG	Enable cache-to-register (dump) mode. When this bit is set ON, double-precision operations unit read operands (e.g.,) Load AQ (ldaq operands) are read from cache memory according to the mapping algorithm and without regard to matching of the full 24-bit absolute main memory address.

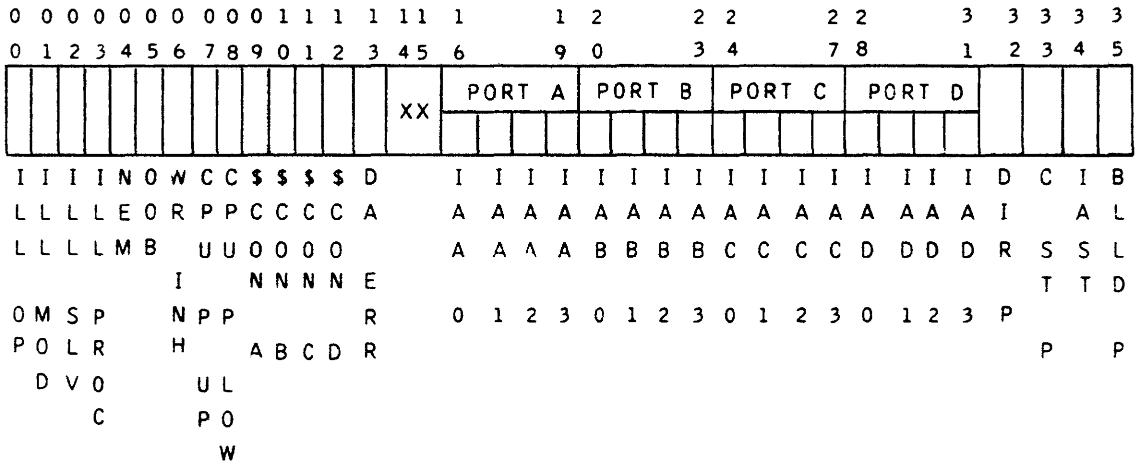
			All other operands address main memory as though the cache memory were disabled. This bit is reset automatically by the hardware for any fault or interrupt.
60			Unassigned
61	x	LEVEL 1/2	Level - LRU - Level 1/2
62	x	LEVEL 1/3	Level - LRU - Level 1/3
63	x	LEVEL 1/4	Level - LRU - Level 1/4
64	x	LEVEL 2/3	Level - LRU - Level 2/3
65	x	LEVEL 2/4	Level - LRU - Level 2/4
66	x	LEVEL 3/4	Level - LRU - Level 3/4
67-69	x		Unassigned.
70-71		....LUF MSB,LSB	Lockup timer setting. The lockup timer may be set to four different values according to the value of this field.

....LUF Value	Lockup Time
0	2 ms
1	4 ms
2	8 ms
3	16 ms

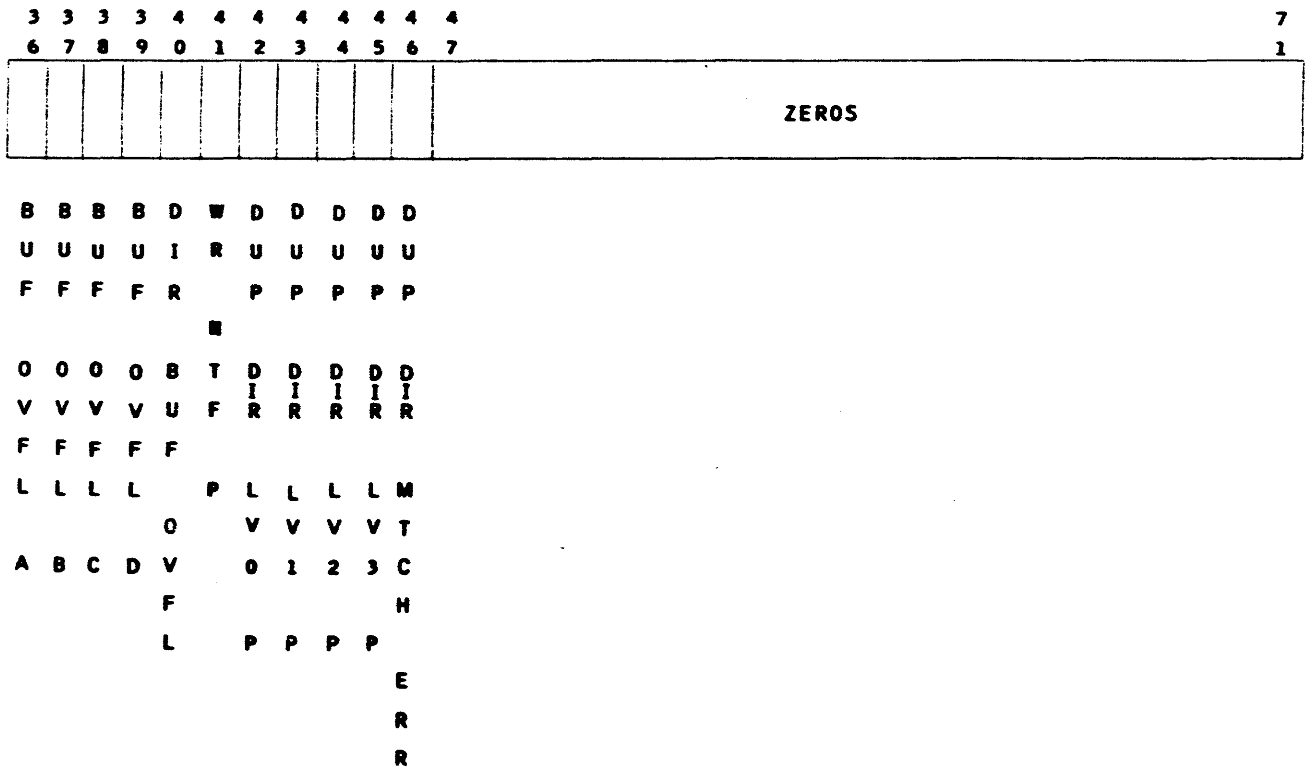
The lockup timer is set to 16 ms when the processor is initialized.

3.8 FAULT REGISTER FORMAT

Even word as stored by Store Central Processor Register (scpr), TAG = 01



Odd word as stored by Store Central Processor Register (scpr), TAG = 01



A combination of flags and registers located in the control unit. The registers are stored and cleared by the scpr, TAG = 01 instruction. The fault register cannot be loaded.

The fault register stores in the processor the conditions for several hardware faults. Data is strobed into the fault register during a fault sequence. Once a bit is set in the fault register, it remains set until the register is stored and cleared. The various cache errors, parity or buffer overflow have occurred some time since the last scpr instruction was executed. They are not defined in time, i.e., when and how many:

Bit	Field	Function
0	ILL OP	An illegal operation code has been detected.
1	ILL MOD	An illegal address modifier has been detected.
2	ILL SLV	An illegal BAR mode procedure has been encountered.
3	ILL PROC	An illegal procedure other than one of three above has been encountered.
4	NEM	A nonexistent main memory address has been requested.

5	OOB	Out of Bounds BAR mode boundary violation has occurred.
6	WR INH	An illegal decimal digit or sign has been detected by the decimal unit.
7	CPU P UP	A parity error has been detected in the upper 36 bits of data.
8	CPU P LOW	A parity error has been detected in the lower 36 bits of data.
9	$CON A	A connect signal has been received through port A.
10	$CON B	A connect signal has been received through port B.
11	$CON C	A connect signal has been received through port C.
12	$CON D	A connect signal has been received through port D.
13	DA ERR	Operations Not Complete. Processor/System controller interface sequence error has been detected.
14-15		Unused.
16-19	IAA	Coded illegal action lines port A.
20-23	IAB	Coded illegal action lines port B.
24-27	IAC	Coded illegal action lines port C.
28-31	IAD	Coded illegal action lines port D.

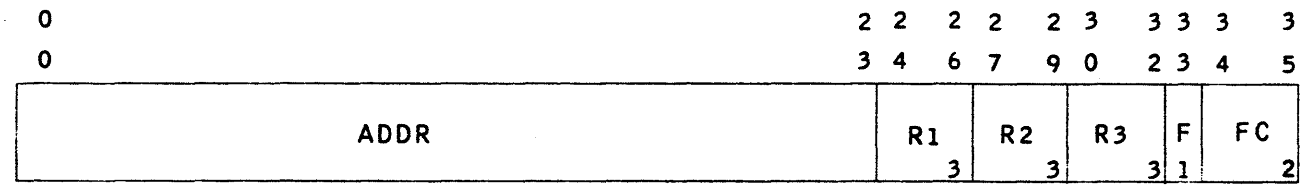
IAA 0123	Priority	Fault	Reason
0000	--		No illegal action
0001	--	Command	Unassigned
0010	05	Store	Nonexistent address
0011	01	Command	Stop on condition
0100	--	Command	Unassigned
0101	12	Parity	Data parity, store unit to system controller
0110	11	Parity	Data parity in store unit
0111	10	Parity	Data parity in store unit and store unit to system controller
1000	04	Command	Not control
1001	13	Command	Port not enabled
1010	03	Command	Illegal command
1011	07	Store	Store unit not ready
1100	02	Parity	Zone-address-command parity, processor to system controller
1101	06	Parity	Data parity, processor to system controller
1110	08	Parity	Zone-address-command parity, system controller to store unit
1111	09	Parity	Data parity, system controller to store unit

Bit	Field	Function
32	DIR P	Directory Parity Error
33	C ST P	Cache Store Parity Error
34	IA ST	Illegal Action on Store to SCU
35	BL LD S	Parity error on a Block Load
36	BUF OVFL A	Duplicate directory port A buffer overflow
37	BUF OVFL B	Duplicate directory port B buffer overflow
38	BUF OVFL C	Duplicate directory port C buffer overflow
39	BUF OVFL D	Duplicate directory port D buffer overflow
40	DIR BUF OVFL	Primary directory buffer overflow
41	WR NTF P	Write notify parity error on any port
42	DUP DIR LV0 P	Parity error on level 0 of duplicate directory
43	DUP DIR LV1 P	Parity error on level 1 of duplicate directory
44	DUP DIR LV2 P	Parity error on level 2 of duplicate directory
45	DUP DIR LV3 P	Parity error on level 3 of duplicate directory
46	DUP DIR MTCH ERR	Duplicate directory multiple match error
47-71		Unused

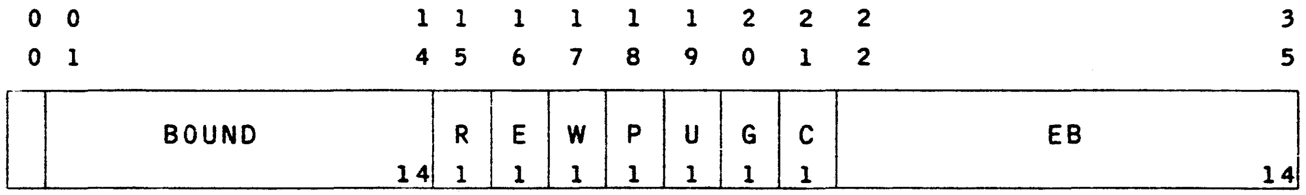
3.9 SEGMENT DESCRIPTOR WORD (SDW) FORMAT

The Segment Descriptor Word (SDW) pair contains information that controls the access to a segment. The SDW for segment _n_ is located at offset 2_n_ in the descriptor segment whose description is currently loaded into the Descriptor Segment Base Register (DSBR).

EVEN WORD



ODD WORD



Segment Descriptor Word (SDW) Format

Bits	Field	Function
0-23	ADDR	24-bit absolute main memory address of unpaged segment (U=1) or segment page table (U=0)
24-26	R1	Ring brackets
27-29	R2	
30-32	R3	
33	F	Directed fault flag 1 = the unpaged segment or segment page table is in main memory 0 = execute the directed fault specified in FC
34-35	FC	The number of the directed faults (DF0-DF3) to be executed if F=0
01-14	BOUND	14 high-order bits of the largest 18-bit modulo 26 offset that may be accessed without causing a descriptor violation, out of segment bounds, fault

15	R	Read permission bit
16	E	Execute permission bit (XEC and XED instructions excluded)
17	W	Write permission bit
18	P	Privileged mode bit 0 = privileged instructions cannot be executed 1 = privileged instructions may be executed if in ring 0
19	U	Paged/unpaged control bit 0 = segment is paged; ADDR is the 24-bit main memory address of the page table 1 = segment is unpaged; ADDR is the 24-bit main memory address of the origin of the segment
20	G	Gate indicator bit 0 = any call into the segment must be to an offset less than the value of EB 1 = any legal segment offset may be called
21	C	Cache control bit 0 = words (operands or instructions) from this segment may not be place in the cache memory 1 = words from this segment may be placed in the cache memory
22-35	EB	Entry bound Any call into this segment must be to an offset less than EB if G=0

3.10 PAGE TABLE WORD (PTW) FORMAT

The Page Table Word (PTW) contains main memory address and status information for a page of a paged segment.

00 17	18 23	24	25	26	27	28	29	30 32	33	34 35
ADDR	X X X X X X	0	0	U	0	0	M	X X X	F	FC
18	6		2	1		2	1	3	1	2

Bits pictured as "x" are ignored by the hardware and may be used by the operating system software.

Bit	Field	Function
00-17	ADDR	18-bit modulo 64 absolute main memory address of page The hardware ignores low order bits of the main memory page address according to page size based on the following:

Page Size in Words	ADDR Bits Ignored
64	none
128	17
256	16-17
512	15-17
1024	14-17
2048	13-17
4096	12-17

Bit	Field	Function
26	U	1 = page has been used (referenced)
29	M	1 = page has been modified

33	F	Directed fault flag 1 = page is in main memory 0 = page not in main memory; execute directed fault FC
34-35	FC	Directed fault number for page fault.

3.11 SEGMENT DESCRIPTOR WORD ASSOCIATIVE MEMORY (SDWAM) FORMAT

Even word of Y-pairs as stored by Store Segment Descriptor Registers (ssdr)

00 – 23	24 – 26	27 – 29	30 – 32	33 – 35
ADDR	R1	R2	R3	0 0 0
24	3	3	3	3

Odd word of Y-pairs as stored by Store Segment Descriptor Registers (ssdr)

36	37 – 50	51	52	53	54	55	56	57	58 – 75
	BOUND	R	E	W	P	U	G	C	CL
1	14	1	1	1	1	1	1	1	14

Data as stored Segment Descriptor Pointers (ssdp)

00	01 14	15 26	27	28 29	30 35
	POINTER 15	0 0 0 0 0 0 0 0 0 0 0 0	F 1	0 0	USE 6

Sixty-four combinations of registers and flags from the appending unit comprising the Segment Descriptor Word Associative Memory (SDWAM). The registers are numbered from 0 through 63 but are not explicitly addressable by number.

Hardware segmentation in the MULTICS processor is implemented by the appending unit. In order to permit addressing by segment number and offset as prepared in the temporary pointer register, a table containing the location and status of each accessible segment must be kept. This table is the descriptor segment. The descriptor segment is located by information held in the Descriptor Segment Base Register (DSBR).

Every time an effective segment number (TPR.TSR) is prepared, it is used as an index into the descriptor segment to retrieve the Segment Descriptor Word (SDW) for the target segment. To reduce the number of main memory references required for segment addressing, the SDWAM provides a content addressable memory to hold the 64 most recently referenced SDWs.

Whenever a reference to the SDW for a segment is required, bits ZESN 11-14 of the effective segment number (TPR.TSR) address a SDWAM column and bits ZESN 00-10 are matched against the four levels of the selected column to check if the referenced SDW is in the M. If the SDW is present the SDWAM directory indicates a bit. The SDW Associative Memory Full/Empty array identifies which levels have a SDW in them. The LRU array is used to determine which SDW is replaced when there is a miss. (LRU: Least Recently Used)

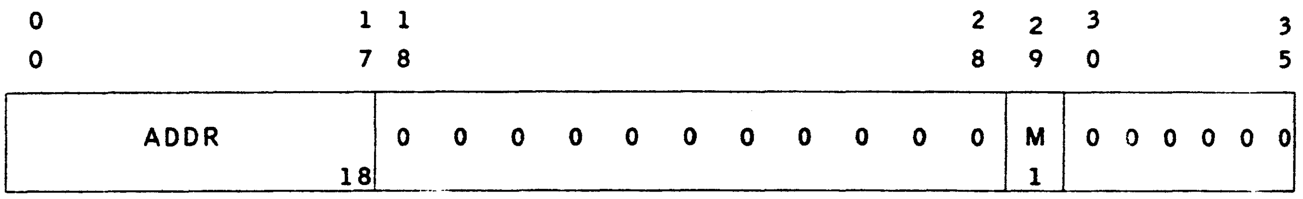
The functions of the constituent registers and flags of each SDWAM register are:

Bit	Register	Function
00-23	SDWAM.ADDR	The 24-bit absolute main memory address of the page table for the target segment if SDWAM.U = 0; otherwise, the 24-bit absolute main memory address of the origin of the target segment.
24-26	SDWAM.R1	Upper limit of read/write ring bracket.
27-29	SDWAM.R2	Upper limit of read/execute ring bracket.
30-32	SDWAM.R3	Upper limit of call ring bracket.
37-50	SDWAM.BOUND	The 14 high-order bits of the last Y-block 16 address within the segment that can be referenced without an access violation, out of segment bound, fault.
51	SDWAM.R	Read permission bit. If this bit is set ON, read access requests are allowed.
52	SDWAM.E	Execute permission bit. If this bit is set ON, the SDW may be loaded into the Procedure Pointer Register (PPR) and instructions fetched from the segment for execution.
53	SDWAM.W	Write permission bit. If this bit is set ON, write access requests are allowed.
54	SDWAM.P	Privileged flag bit. If this bit is set ON, privileged instructions from the segment may be executed if PPR.PRR is 0.

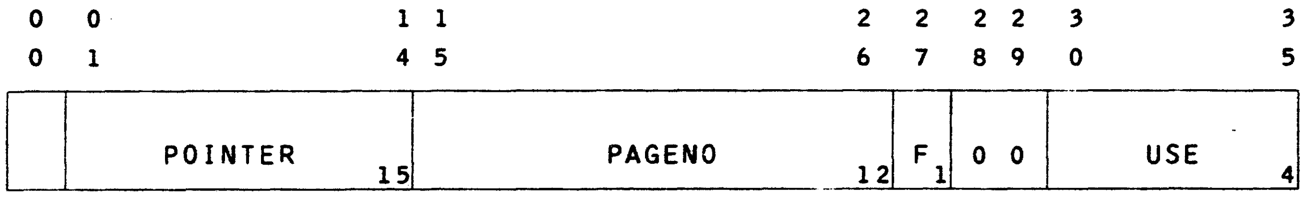
Bit	Register	Function
55	SDWAM.U	Unpaged flag bit. If this bit is set ON, the segment is unpaged and SDWAM.ADDR is the 24-bit absolute main memory address of the page table for the segment. If this bit is set OFF, the segment is paged and SDWAM.ADDR is the 24-bit absolute main memory address of the page table for the segment.
56	SDWAM.G	Gate control bit. If this bit is set OFF, calls and transfers into the segment must be to an offset no greater than the value of SDWAM.CL as described below.
57	SDWAM.C	Cache control bit. If this bit is set ON, data and/or instructions from the segment may be placed in the cache memory.
58-71	SDWAM.CL	Call limiter (entry bound) value. If SDWAM.G is set OFF, transfers of control into the segment must be to segment addresses no greater than this value.
01-14	SDWAM.POINTER	The effective segment number used to fetch this SDW from main memory.
27	SDWAM.F	Full/empty bit. If this bit is set ON, the SDW in the register is valid. If this bit is set OFF, a hit is not possible. All SDWAM.F bits are set OFF by the instructions that clear the SDWAM.
30-35	SDWAM.USE	This field is now six bits long and reflects the output of the 6-bit LRU array. Therefore, the field reflects the use value for the four levels of a column and no longer reflects a FIFO queue among all the PTWs.

3.12 PAGE TABLE WORD ASSOCIATIVE MEMORY (PTWAM) FORMAT

Data as stored by Store Page Table Register (sptr)



Data as stored by Store Page Table Pointers (sptp)



Sixty-four combinations of registers and flags from the appending unit comprising the Page Table Word Associative Memory (PTWAM). The registers are numbered from 0 through 63 but are not explicitly addressable by number.

Hardware paging in the MULTICS processor is implemented by the appending unit. In order to permit segment addressing by page number and page offset as derived from the computed address prepared in the temporary pointer register (TPR.CA), a table containing the location and status of each page of an accessible segment must be kept. This table is the page table for the segment. The page table for an accessible paged segment is located by information held in the Segment Descriptor Word (SDW) for the segment.

Every time a computed address (TPR.CA) for a paged segment is prepared, it is separated into a page number and a page offset. The page number is used as an index into the page table to retrieve the Page Table Word (PTW) for the target page. To reduce the number of page main memory references required for paging, the PTWAM provides a content addressable memory to hold the 64 most recently referenced PTWs.

Whenever a reference to the PTW for a page of a paged segment is required, the page number is matched associatively against all 64 PTWAM.PAGENO registers. If the PTWAM logic indicates a hit, the PTWAM Full/Empty array identifies which levels have valid page numbers. The LRU array is used to determine which PTW is replaced when there is a miss.

Bit	Register	Function
00-17	PTWAM.ADDR	The 18 high-order bits of the 24-bit absolute main memory address of the page. The hardware ignores low-order bits of this page address according to page size based on the following:
29	PTWAM.M	Page modified flag bit. This bit is set ON whenever the PTW is used for a store type instruction. When the bit changes value from 0 to 1, a special extra cycle is generated to write it back into the PTW in the page table in main memory.
01-14	PTWAM.POINTER	The effective segment number used to fetch this PTW from main memory.

Page size in words	ADDR bits ignored
64	17
128	16-17
256	15-17
512	15-17
1024	14-17
2048	13-17
4096	12-17

15-26	PTWAM.PAGENO	The 12 high-order bits of the 18-bit computed address (TPR.CA) used to fetch this PTW from main memory. Low-order bits are forced to zero by the hardware and not used as part of the page table index according to page size based on the following:

Page size in Words Forced	PAGENO Bits
64	11
128	10-11
256	09-11
512	09-11
1024	08-11
2048	07-11
4096	06-11

27	PTWAM.F	Full/Empty bit. If this bit is set ON, the PTW in the register is valid. If this bit is set OFF, a hit is not possible. All PTWAM.F bits are set OFF by the instructions that clear the PTWAM.
30-35	PTWAM.USE	The PTWAM.USE field reflects the six LRU bits from the Page Table Word Associative Memory. Decode of these bits determine which level of a particular column was Least Recently Used (LRU).

3.13 ID PROM

A 1K by 8-bit PROM (Programmable Read Only Memory) is included in the DPS 8 processor design. Data fields in the PROM identify the processor Model Number, Serial Number, Date Shipped, etc. Character information in the ID PROM shall be encoded in UASCI.

Byte locations in the PROM will be read into the lower 8-bit locations of the A-register by the execution of the TSW instruction with a DL modifier. RSW 123, DL will result in the contents of PROM byte 123 being read into locations 28-35 of the accumulator register.

Byte Location (Octal)	Contents
0-13	CPU Model Number (Byte 0 = Most Significant Byte)
13-25	CPU Serial Number (Byte 13 = Most Significant Byte)

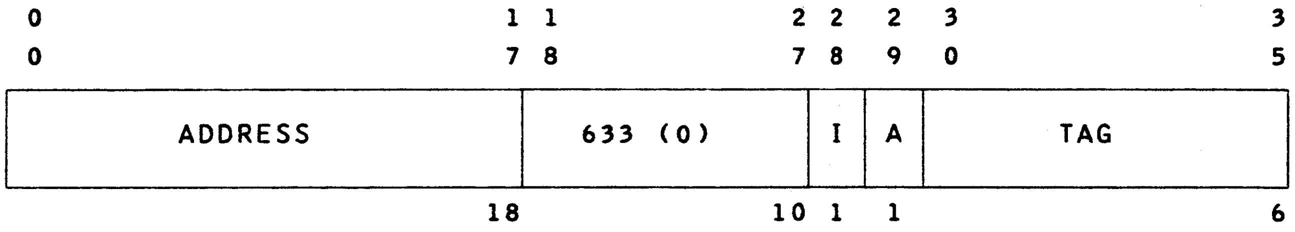
26-33		Date-Ship Code (YYMMDD)
34-40		CPU ID Field (reference RSW 2)
	Byte 40	Bits 0-3 (Bits 32-35 of RSW 2 Field) Bit 4=1 Hex Option included Bit 5=1 RSCR (Clock) is Slave Mode included Bits 6-7 Reserved for later use
50		Operating System Use
51-1777$_8$		To be defined

NOTE: There is the possibility of disagreement between the ID bits of RSW 2 and the ID bits of PROM locations 34-40. This condition could result when alterable configuration information is contained in the PROM. The user is advised to ignore the PROM fields which contain the processor fault vector base (GCOS III) and the processor number and rely on the RSW 2 bits for this purpose. Bits 14-16 of the RSW 2 should be ignored and the bits representing this information in the PROM should be treated as valid.

3.14 INSTRUCTIONS AFFECTED BY CHANGES

3.14.1 Instruction Affected By Processor Numbers Processor to SCU Ports

3.14.1.1 rccl Read Calendar Clock



SUMMARY: 00___0->C(AQ)

C(calendar clock)-> C(AQ)

MODIFICATIONS: All except du, dl, ci, sc, scr

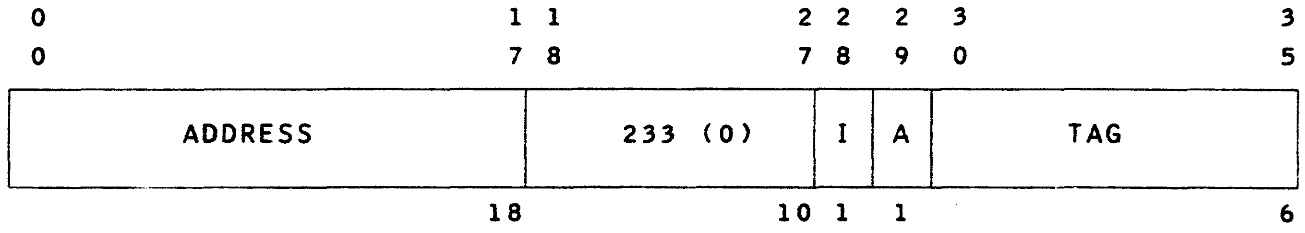
INDICATORS: None affected

NOTES: C(TPR.CA)$_{1,2}$ specify which processor port (i.e., which system controller) is to be used. The contents of the clock in the designated system controller replace the contents of the AQ-register.

Attempted execution in BAR mode causes an illegal procedure fault.

Attempted repetition with the rpt, rpd, or rpl instructions causes an illegal procedure fault.

3.1.4.1.2 rmcm Read Memory Controller Mask Register



SUMMARY: For the selected system controller:

If the processor has a mask register assigned, then

C(Interrupt Mask Register) -> C(AQ)$_{0,15}$ and $_{36,51}$

C(Port Enable Register) -> C(AQ)$_{32,35}$ and $_{68,71}$

00-0-> C(AQ)$_{16,31}$ and $_{52,67}$
otherwise, 00...0 ---> C(AQ)$_{0,31}$ and $_{36,67}$

C(Port Enable Register) -> C(AQ)$_{32,35}$ and $_{68,71}$

MODIFICATIONS: All except du, di, ci, sc, scr

INDICATORS: (Indicators not listed are not affected)

Zero	If C(AQ) = 0, then ON; otherwise OFF
Negative	If C(AQ)0 = 1, then ON; otherwise OFF

NOTES: The contents of the mask register remain unchanged.

$C(TPR.CA)_{1,2}$ specify which processor port (i.e., which system controller) is used.

Attempted execution in normal or BAR modes causes an illegal procedure fault.

3.14.1.3 smcm Set Memory Controller Mask Register

0–17	18–27	28	29	30–35
ADDRESS	553 (0)	I	A	TAG
18	10	1	1	6

SUMMARY; For the selected system controller:

If the processor has a mask register assigned,

$C(AQ)_{0,15}$ and $36,51$ ->C(Interrupt Mask Register)

MODIFICATIONS; All except du, di, ci, sc, scr

INDICATORS: None affected

NOTES: $C(TPR.CA)_{1,2}$ specify which processor port (i.e., which system controller) is used.

Attempted execution in normal or BAR modes causes an illegal procedure fault.

3.14.1.4 smic Set Memory Controller Interrupt Cells

00	17	18	27	28	29	30	35
ADDRESS		451 (0)		I	A	TAG	
	18		10	1	1		6

SUMMARY: For i = 0, 1, ..., 15 and $C(A)_{35}$ = 0: if $C(A)_i$ = 1, then set interrupt cell i ON

For i = 0, 1, ..., 15 and $C(A)_{35}$ = 1: if $C(A)_i$ = 1, then set interrupt cell 16+i ON

MODIFICATIONS: All except du, dl, ci, sc, scr

INDICATORS: None affected

NOTES: $C(TPR.CA)_{1,2}$ specify which processor port (i.e., which system controller) is used. If the processor has no assigned mask register in the selected system controller, a NOP occurs.

Attempted execution in normal or BAR modes causes an illegal procedure fault.

3.14.1.5 rsw Read Switches

00 – 17	18 – 27	28	29	30 – 35
ADDRESS	231 (0)	I	A	TAG
18	10	1	1	6

SUMMARY: Bit 16 and 17 of the final computed address, C(TPR.CA), are used to select certain processor switches whose settings are read into the A-register.

The switches selected are as follows:

Bit 16	Bit 17	Function
0	0	C(data switches)-> C(A)
0	1	C(configuration switches for ports A,B,C,D) -> C(A)
1	0	See configuration switch data Section 3.14.1.5.1 C(switches)-> C(A)

MODIFICATIONS: All, but none affect instruction execution

INDICATORS: (Indicators not listed are not affected)

Zero	If C(A) = 0, then ON; otherwise OFF
Negative	If $C(A)_0$ = 1, then ON; otherwise OFF

NOTES:

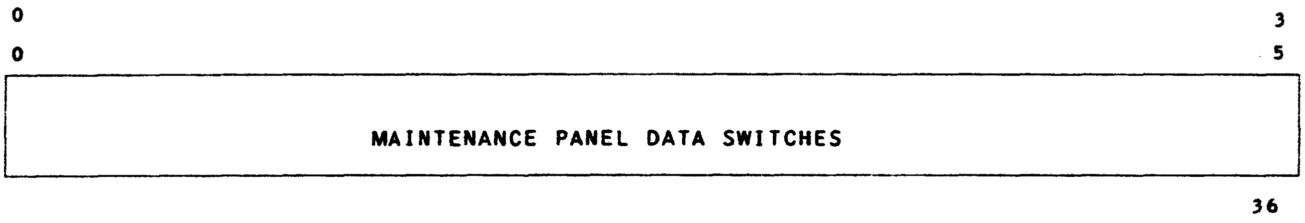
Attempted execution in normal or BAR modes causes an illegal procedure fault.

Attempted repetition with the rpt, rpd, or rpl instructions causes an illegal procedure fault.

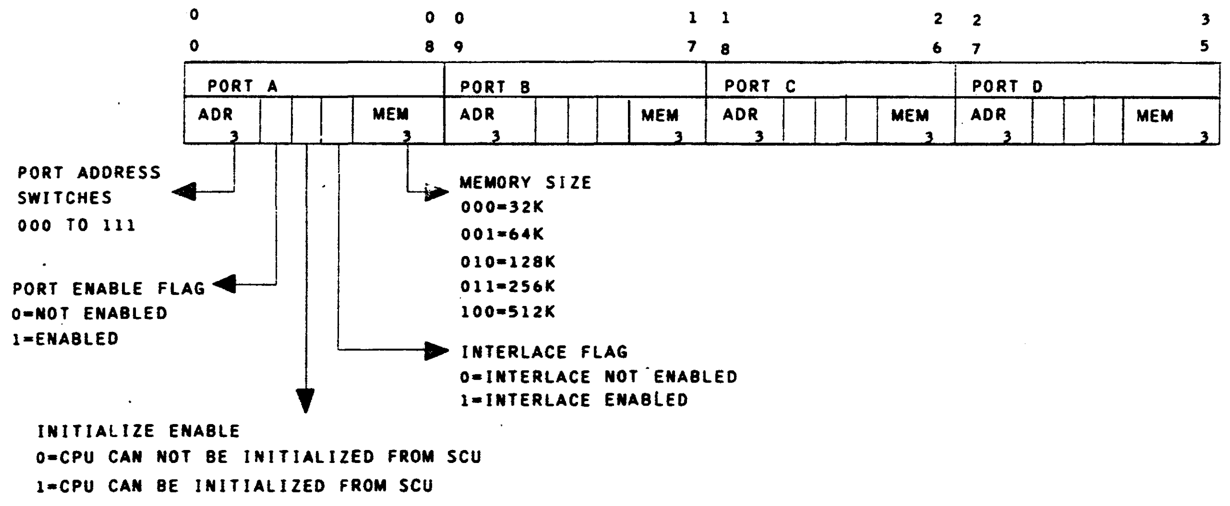
If bits 16 and 17 are 11, the function returned will be the same as when bits 16 and 17 are 00. Software should observe only the three definitions above, and not use 11.

3.14.1.5.1 Configuration Switch Data

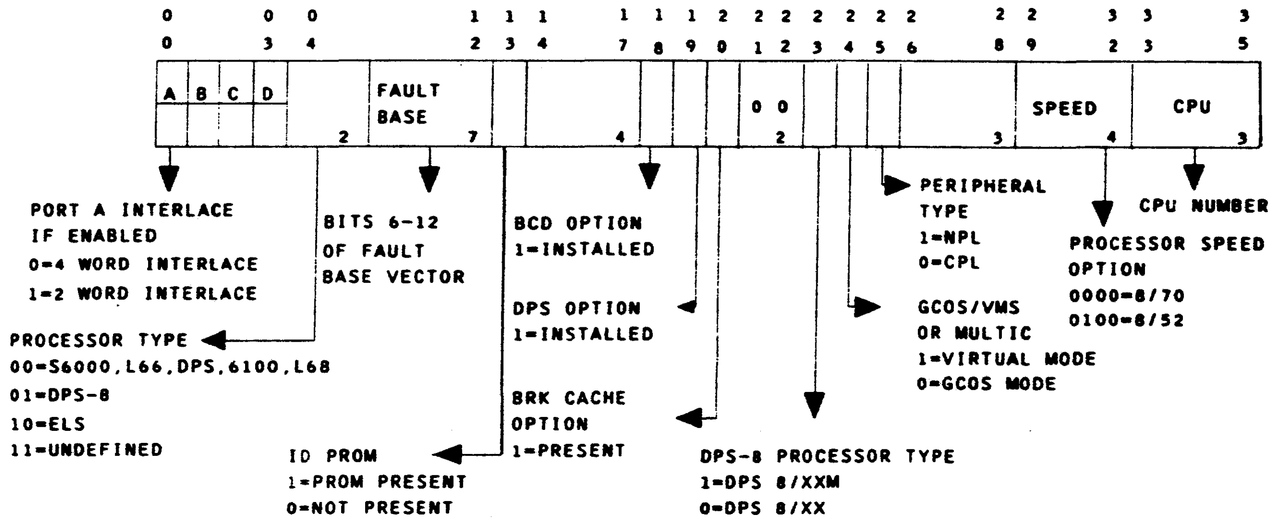
Data read by Read Switches (RSW)$Y_{16,17}$ = 00 or 11



Data read by Read Switches (RSW)$Y_{16,17}$ = 01 (port A-B)


00 08 09 17 18 26 27 35
PORT A
PORT B
PORT C
PORT D
ADR 3
MEM 3
PORT ADDRESS SWITCHES 000 TO 111
PORT ENABLE FLAG 0=NOT ENABLED 1=ENABLED
INITIALIZE ENABLE 0=CPU CAN NOT BE INITIALIZED FROM SCU 1=CPU CAN BE INITIALIZED FROM SCU
MEMORY SIZE 000=32K 001=64K 010=128K 011=256K 100=512K
INTERLACE FLAG 0=INTERLACE NOT ENABLED 1=INTERLACE ENABLED

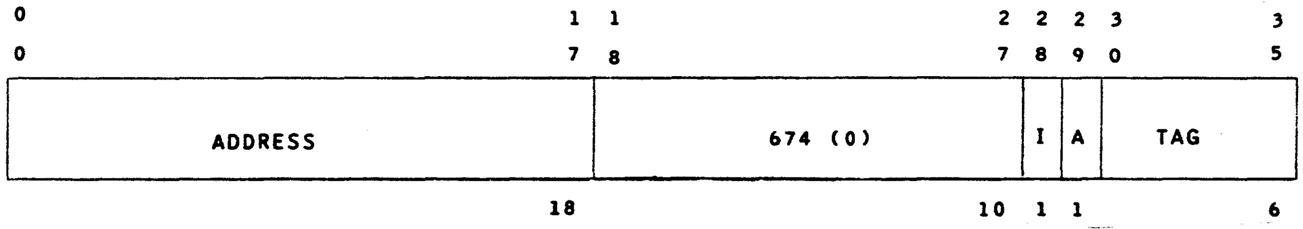
Data read by Read Switches (RSW)$Y_{16,17}$ = 10


00 03 04 12 13 14 17 18 19 20 21 22 23 24 25 26 28 29 32 33 35
A B C D
2
FAULT BASE 7
4
0 0 2
3
SPEED 4
CPU 3
PORT A INTERLACE IF ENABLED 0=4 WORD INTERLACE 1=2 WORD INTERLACE
PROCESSOR TYPE 00=S6000,L66,DPS,6100,L68 01=DPS-8 10=ELS 11=UNDEFINED
BITS 6-12 OF FAULT BASE VECTOR
ID PROM 1=PROM PRESENT 0=NOT PRESENT
BCD OPTION 1=INSTALLED
DPS OPTION 1=INSTALLED
BRK CACHE OPTION 1=PRESENT
DPS-8 PROCESSOR TYPE 1=DPS 8/XXM 0=DPS 8/XX
PERIPHERAL TYPE 1=NPL 0=CPL
GCOS/VMS OR MULTIC 1=VIRTUAL MODE 0=GCOS MODE
PROCESSOR SPEED OPTION 0000=8/70 0100=8/52
CPU NUMBER

The data above, read by RSW,Y_{16-27}= 10 is also called the processor identification field. The field serves as a diagnostic aid to identify the processor type and installed options.

3.14.2 Instructions Affected By History Registers

3.14.2.1 lcpr Load Central Processor Register



SUMMARY: Load selected register as noted.

MODIFICATIONS: None. The instruction word TAG field is used for the register selection word as follows:

C(TAG)	Data and Register(s)
02	C(Y)->C(Cache mode register)
04	C(Y)->(Mode register)
03	00...0 -> C(CU, OU/DU, and APU history register)$_{0,71}$
07	11...1 -> C(CU, OU/DU, and APU history register)$_{0,71}$

INDICATORS: None affected

NOTES: For TAG values 03 and 07, the history register loaded is selected by the current value of a cyclic counter for each unit. The individual cyclic counters are advanced by one count for each execution of the instruction.

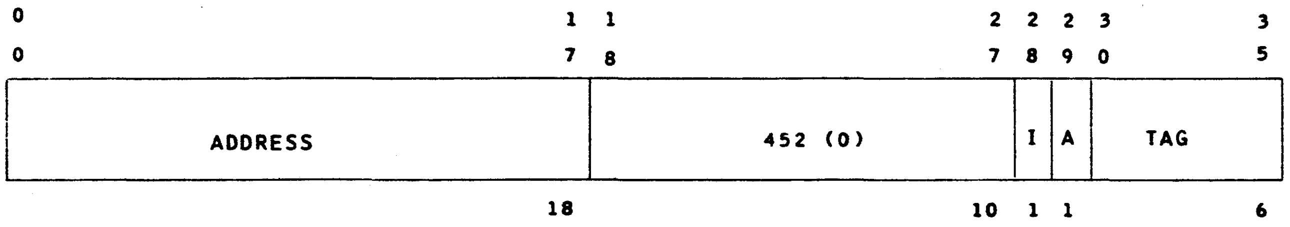
Use of TAG values other than those defined above causes an illegal procedure fault.

Attempted execution in normal or BAR modes causes an illegal procedure fault.

Attempted repetition with the rpt, rpd, or rpl instructions causes an illegal procedure fault.

HONEYWELL CONFIDENTIAL & PROPRIETARY

3.14.2.2 scpr Store Central Processor Registers



SUMMARY: Store selected register as noted.

MODIFICATIONS: None, the instruction word TAG field is used for register selection word as follows:

C(TAG)	MEANING
00	C(APU history register) ->C(Y-pair)
01	C(fault register) ->C(Y-pair)$_{0,46}$ 00...0->C(Y-pair)$_{47,71}$
06	C(mode register) ->C(Y-pair)$_{0,35}$ C(Cache mode register) ->C(Y-pair)$_{36,71}$
20	C(CU history register) ->C(Y-pair)
40	C(OU/DU history register) ->C(Y-pair)

INDICATORS: None affected

NOTES: For TAG field values shown are octal.

For TAG values 00, 20 and 40, the history register stored is selected by current value of a cyclic counter for each unit. The individual cyclic counters are advanced by one count for each execution of the instruction.

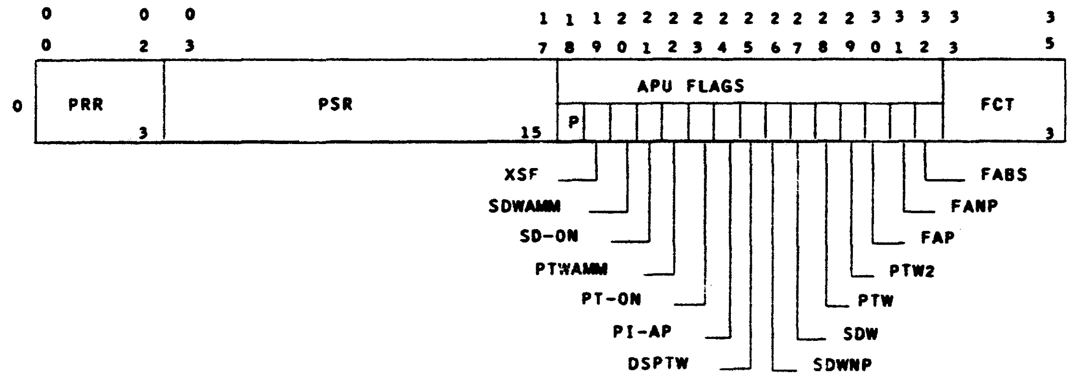
The use of TAG values other than those defined above causes an illegal procedure fault.

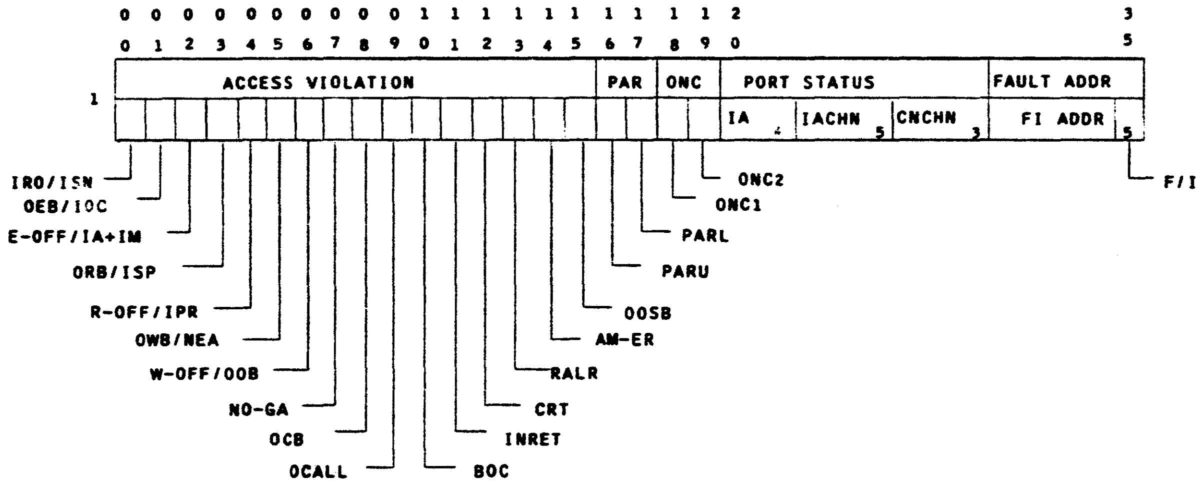
Attempted execution in normal or BAR modes causes an illegal procedure fault.

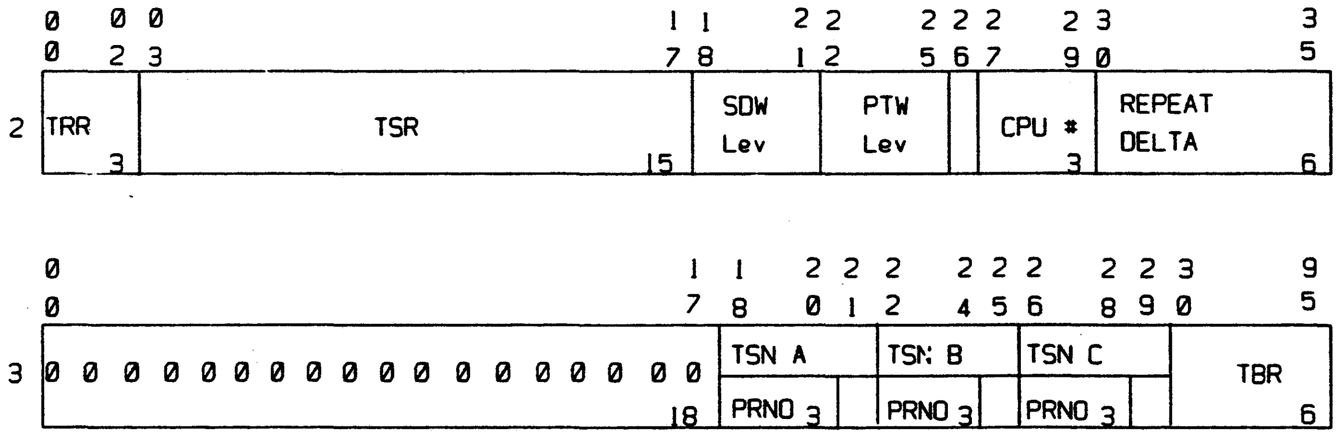
Attempted repetition with the rpt, rpd, or rpl instructions causes an illegal procedure fault.

3.14.3 scu Store Control Unit

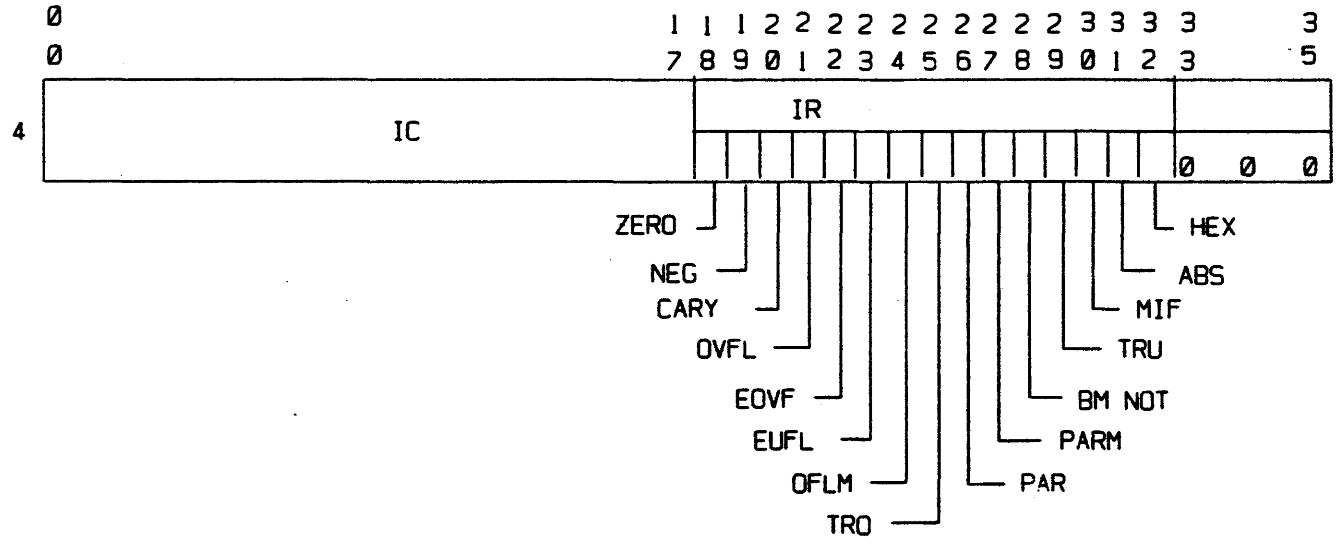
Control unit date stored by Store Control Unit (scu) instruction is eight machine words. Format follows:



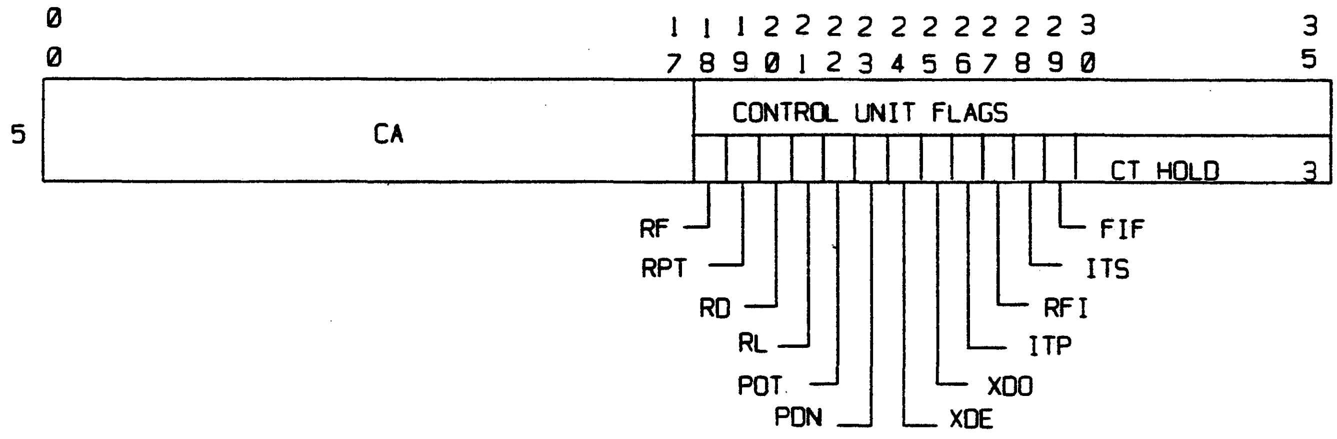




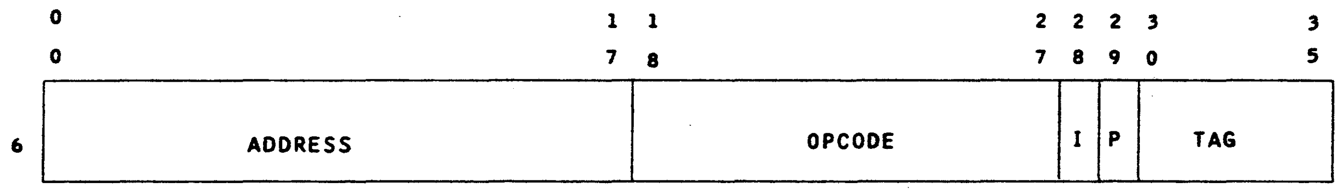
00 02 03 17 18 21 22 25 26 27 29 30 35
2
TRR 3
TSR 15
SDW Lev
PTW Lev
CPU # 3
REPEAT DELTA 6
00 17 18 20 21 22 24 25 26 28 29 30 35
3
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 18
TSN A
PRNO 3
TSN B
PRNO 3
TSN C
PRNO 3
TBR 6



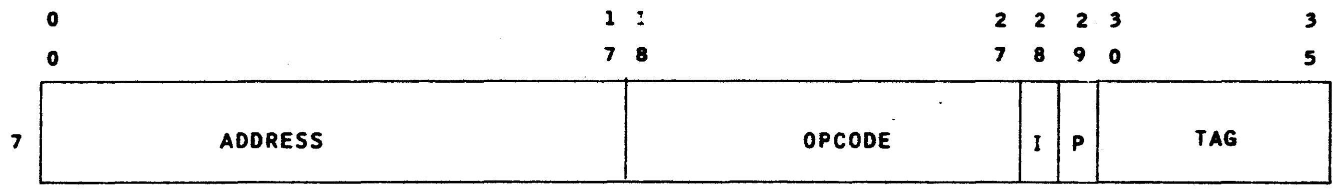
00 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 35
4
IC
IR
0 0 0
ZERO
NEG
CARY
OVFL
EOVF
EUFL
OFLM
TRO
PAR
PARM
BM NOT
TRU
MIF
ABS
HEX



00 17 18 19 20 21 22 23 24 25 26 27 28 29 30 35
5
CA
CONTROL UNIT FLAGS
CT HOLD 3
RF
RPT
RD
RL
POT
PDN
XDE
XDO
ITP
RFI
ITS
FIF



Contents of the working instruction register, reflect conditions at the exact point of address preparation when a fault or interrupt occur. ADDRESS and TAG fields are replaced with data from pointer registers during indirect cycle. Each instruction of the current pair is moved to this register before actual address preparation begins.



Contents of the instruction holding register. Contains the odd word of the last instruction pair fetched from main memory. Note, primarily because of overlap this instruction is not necessarily paired with the instruction in word 6.

WORD	BIT	FIELD	FUNCTION
0	0-2	PRR	Procedure Ring Register (PPR.PRR)
	3-17	PSR	Procedure Segment Register (PPR.PSR)
	18	P	Privileged bit (PPR.P)
	19	XSF	External Segment Flag
	20	SDWAMM	Match on SDWAM
	21	SD-ON	SDWAM enabled
	22	PTWAMM	Match on PTWAM
	23	PT-ON	PTWAM enabled
	24	PI-AP	Instruction fetch append cycle
	25	DSPTW	Fetch descriptor segment PTW
	26	SDWNP	Fetch SDW - Nonpaged
	27	SDWP	Fetch SDW - paged
	28	PTW	Fetch PTW
	29	PTW2	Fetch prepared PTW
	30	FAP	Fetch final address-paged
	31	FANP	Fetch final address-nonpaged
	32	FABS	Fetch final address-absolute
	33-35	FCT	Fault counter - counts retries
1	0	IRO	Access violation fault - Illegal Ring Order
		ISN	Store Fault - Illegal Segment Number
	1	OEB	Access violation fault - Out of Execute Fault
		IOC	Illegal procedure fault - Illegal Op Code
	2	E-OFF	Access violation fault - Execute bit OFF
		IA+IM	Illegal procedure fault - Illegal Address or Modifier
	3	ORB	Access violation fault - Out of Read Bracket
		ISP	Illegal procedure fault - Illegal Slave Procedure
	4	R-OFF	Access violation fault - Read bit is OFF
		IPR	Illegal procedure fault - Illegal EIS digit.
	5	OWB	Access violation fault - Out of Write Bracket
		NEA	Store fault - Nonexistent Address
	6	W-OFF	Access violation fault - Write bit is OFF
		OOB	Store fault - Out Of Bounds (BAR mode)
	7	NO-GA	Access violation fault - Not a Gate
	8	OCB	Access violation fault - Out of Call Bracket
	9	OCALL	Access violation fault - Outward Call
	10	BOC	Access violation ffault - Bad Outward Call
	11	INRET	Access violation fault - Inward Return
	12	CRT	Access violation fault - Cross Ring Transfer

WORD	BIT	FIELD	FUNCTION
	13	RALR	Access violation fault - Ring Alarm
	14	AM-ER	Access violation fault - Associative Memory Error
	15	OOSB	Access violation fault - Out Of Segment Bounds
	16	PARU	Parity fault - Processor Parity Upper
	17	PARL	Parity fault - Processor Parity Lower
	18	ONC1	Operation not complete fault - Processor/system controller sequence error #1
	19	ONC2	Operation not complete fault - Processor/system controller sequence error #2
	20-23	IA	System controller illegal action lines
	24-28	IACHN	Illegal action processor port
	29-31	CNCHN	Connect fault - connect processor port
	32-34	F/I ADDR	Module 2 fault/interrupt vector address
	35	F/I	Fault/Interrupt flag 0 = interrupt 1 = fault
2	0-2	TRR	Temporary Ring Register (TPR.TRR)
	3-17	TSR	Temporary Segment Register (TPR.TSR)
	18-21	SDW-Lev	SDW AM level enabled
	22-25	PTW-Lev	PTW AM level enabled
	27-29	CPU #	CPU number
	30-35	DELTA	Address increment for repeats
	0-17		Must be zeros - unassigned
3	18-21	TNS A	Pointer number register for non-EIS operands or for EIS operand #1 substructured are:
	18-20	PRNO	Pointer Register number
	21	---	1 = PRNO valid
	22-25	TSN B	Pointer register number for EIS operand #2 substructured as TSN A
	26-29	TSN C	Pointer register number for EIS operand #3 substructured as TSN A
	30-35	TBR	Current bit offset

WORD	BIT	FIELD	FUNCTION
4	0-17	IC	Instruction counter (PPR.IC)
	18	ZERO	Zero indicator
	19	NEG	Negative indicator
	20	CARY	Carry indicator
	21	OVFL	Overflow indicator
	22	EOVF	Exponent Overflow indicator
	23	EUFL	Exponent Underflow indicator
	24	OFLM	Overflow mark indicator
	25	TRU	Tally Runout indicator
	26	PAR	Parity Error indicator
	27	PARM	Parity Mark indicator
	28	BM NOT	Not BAR mode indicator
	29	TRU	EIS truncation indicator
	30	MIF	Midinstruction interrupt indicator
	31	ABS	Absolute Mode indicator
	32	HEX	Hex Mode indicator
5	0-17	CA	Current Computer Address (TPR.CA)
	18	RF	First cycle all repeat instructions
	19	RPT	Execute a Repeat (rtp) instruction
	20	RD	Execute a Repeat Double (rpd) instruction
	21	RL	Execute a Repeat Link (rpl) instruction
	22	POT	Prepare operand tally. This flag is up until the indirect word of an indirect, then tally address modifier is successfully fetched
	23	PON	Prepare operand no tally. This flag is up until the indirect word of a return type transfer instruction is successfully fetched. It indicates that there is no indirect chain even though an indirect fetch is being done
	24	XDE	Execute Double (Xed) instruction is at an even location
	25	XDO	Execute Double (Xed) instruction is at an odd location
	26	ITP	Execute ITP indirect cycle
	27	RFI	Faulted instruction pointed to by Xed was at even location, or fault occurred on a decimal instruction
	28	ITS	Execute its indirect cycle
	29	FIF	Fault occurred during instruction fetch
	30-35	CT HOLD	Contents of the modified holding register

3.14.4 Instructions Affected by 8K Cache

3.14.4.1 stacq Store A Conditional on Q

00 17	18 27	28	29	30 35
ADDRESS	654 (0)	I	A	TAG
18	10	1	1	6

SUMMARY: This instruction causes the processor to delay until all its outstanding writes have received completion responses from the SCUs.

If C(Y) = C(Q), then C(A) -> C(Y)

MODIFICATIONS: All except du, dl, ci, sc, scr

INDICATORS: (Indicators not listed are not affected)

Zero If initial (C(Y) = C(Q); then ON; otherwise OFF

NOTES: If the initial C(Y) = C(Q); then C(Y) is not changed by the stacq instruction.

Attempted repetition with the RPL instruction causes an illegal procedure fault.

The stacq instruction uses a special main memory reference that prohibits such references by other processors between the test and the data transfer. Thus, it may be used for data locking.

This instruction delay provides a synchronizing instruction which is required at the end of a block of gated code immediately preceding the operation which opens the gate. Otherwise, the hardware will not guarantee that all stores preceding the gate opening have been completed. This instruction combines the synchronizing function with the open gate operation.

3.14.4.2 camp Clear Associative Memory Page

0–17	18–27	28	29	30–35
ADDRESS	532 (1)	I	A	TAG
18	10	1	1	6

SUMMARY: For i=0,1,---,15

0-> C(PTWAM(i)-F)

C(PTWAM(i)-LRU) is initialized

MODIFICATIONS: All except du,dl,ci,scr

INDICATORS: None affected

NOTES: The full/empty bit of each PTWAM register is set to 0, and the LRU counters (PTWAM.LRU) are initialized. The remainder of C(PTWAM(i)) is unchanged.

A level of the associative memory is disabled if $C(TPR.CA)_{16,17}=01$.

A level of the associative memory is enabled if, it is disabled, when $C(TPR.CA)_{16,17}=10$.

Level j is selected to be disabled/enabled if

$C(TPR.CA)_{10+j}=1$; j=1,2,3,4

All levels are selected to be disabled/enabled if

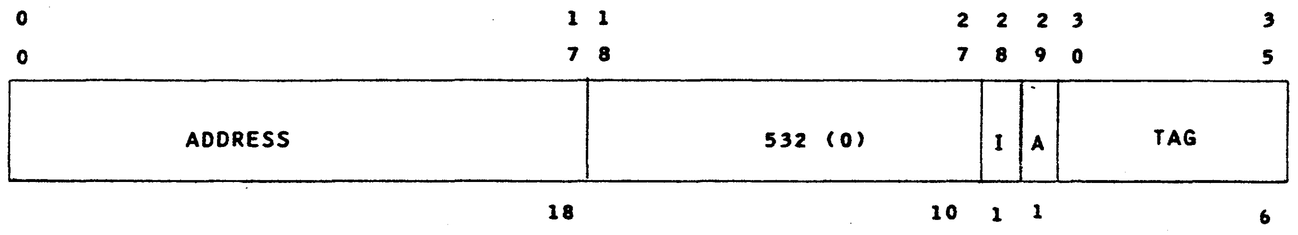
$C(TPR.CA)_{11,14}=0$.

$C(TPR.CA)_{15}$ is disregarded (old cached clear bit).

Attempted execution in normal or BAR modes caused an illegal procedure fault.

Attempted repetition with the rpt, rpd, or rpl instructions causes an illegal procedure fault.

3.14.4.3 cams Clear Associative Memory Segments



SUMMARY: For i=0,1,---,15

0-> C(SDWAM(i).FO)

C(PTWAM(i).LRU) is initialized

MODIFICATIONS: All except du,dl,ci,sc,scr

INDICATORS: None affected

NOTES: The full/empty bit of each SDWAM register is set to zero, and the LRU counters (SDWAM.LRU) are initialized. The remainder of C(SDWAM(i)) are unchanged.

A level of the associative memory is disabled if $C(TPR.CA)_{16,17}$=01.

A level of the associative memory is enabled if, it is disabled, when $C(TPR.CA)_{16,17}$=10.

Level j is selected to be disabled /enabled if $C(TPR.CA)_{10+j}$=1; j-1,2,3,4

All levels are selected to be disabled/enabled if

$C(TPR.CA)_{11,14}$=0.

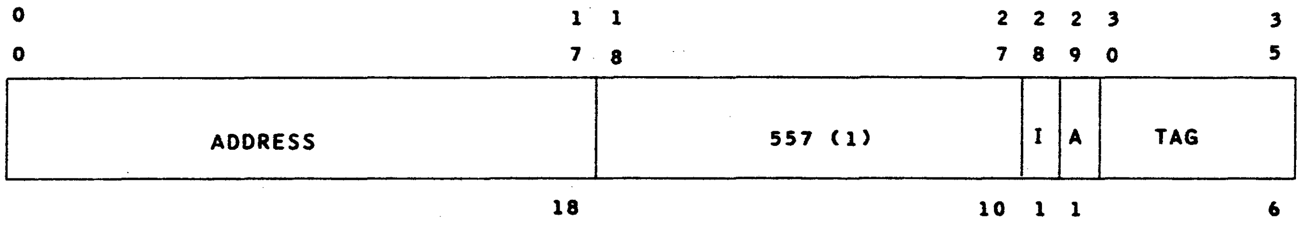
$C(TPR.CA)_{15}$ is disregarded (old cache clear bit).

Attempted execution in normal or BAR modes causes an illegal procedure fault.

Attempted repetition with the rpt,rpd, or rpl instructions causes an illegal procedure fault.

3.14.5 Instructions Affected by Associative Memory Changes

3.14.5.1 sptp STore Page Table Pointer



SUMMARY: This instruction stores 16 words from the selected level (j) of the directory of the Page Table Word associative memory. There are four levels.

Level j is selected by $C(TPR.CA)_{12,13}$

For i=0,1,---15

$C(PTWAM(i,j).POINTER) \rightarrow$

$C(Y\text{-}BLOCK16=i)_{0,14}$

$C(PTWAM(i,j).PAGENO) \rightarrow$

$C(Y\text{-}block\ 16+i)_{15,22}$

$0000 \rightarrow C(Y\text{-}block16+i)_{23,27}$

$C(PTWAM(i,j).F \rightarrow C(Y\text{-}block16+i)_{27}$

$00 \rightarrow C(Y\text{-}block16+i)_{28,29}$

$C(PTWAM(i,j.).LRU) \rightarrow$
$C(Y\text{-}block16+i)_{30,35}$

MODIFICATIONS: All except du,dl,ci,sc,scr

INDICATORS: None affected

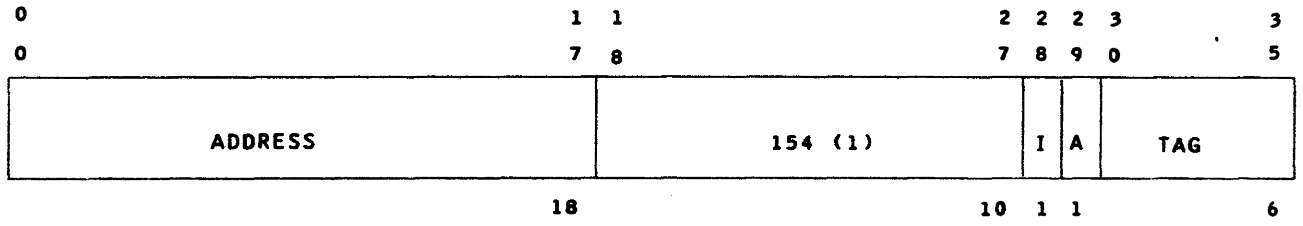
NOTES: The contents of the associative memory remains unchanged.

The associative memory is ignored during virtual-to-real address preparation (forced to "no match").

Attempted execution in normal or BAR modes causes an illegal procedure fault.

Attempted repetition with rpt,rpd, or rpl instruction causes an illegal procedure fault.

3.14.5.2 sptr Store Page Table Registers



SUMMARY This instruction stores 16 words from the selected level (j) of the contents of the Page Table Word associative memory. There are four levels.

Level j is selected by C(TPR.CA)$_{12,13}$

For i=0, lm --- 15

C(PTWAM(i,j).PAGE ADDR)->

C(-BLOCK16+i)$_{0,13}$

00...0-> C(Y-BLOCK16+i)$_{14,28}$

C(PTWAM(i,j53M)-> C(Y-BLOCK16+i)$_{29}$

000000-> C(Y-BLOCK16+i)$_{30,35}$

MODIFICATIONS: All except du,dl,ci,sc,scr

INDICATORS: None affected

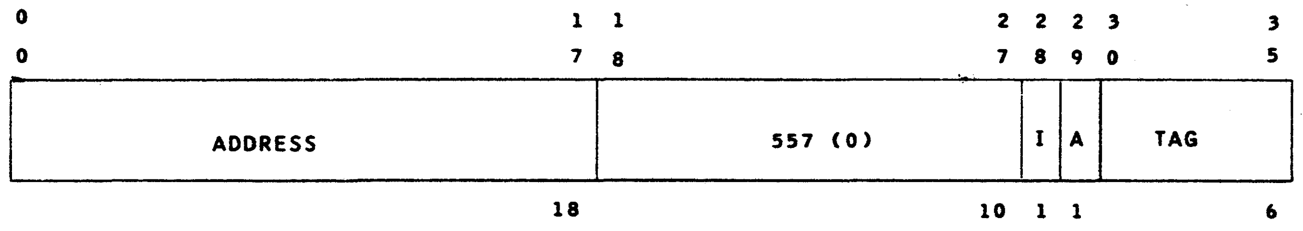
NOTES: The contents of the associative memory remains unchanged.

The associative memory is ignored during virtual-to-real address preparation (forced to "no match").

Attempted execution in normal or BAR modes causes an illegal procedure fault.

Attempted repetition with rpt,rpd, or rpl instructions causes an illegal procedure fault.

3.14.5.3 ssdp Store Segment Descriptor Pointer



SUMMARY: This instruction stores 16 words from the selected level (j) of the directory of Segment Descriptor Word associative memory. There are four levels.

Level j is selected by $C(TPR.CA)_{12,13}$

For i=0,1,...15

$C(SDWAM(i,j).POINTER)$->
$C(Y\text{-}BLOCK16+i)_{15,26}$

$C(SDWAM(i,j).F)$->
$C(Y\text{-}BLOCK16+i)_{28,29}$

$C(SDWAM(i,j).LRU)$->
$C(Y\text{-}block16+i)_{30,35}$

MODIFICATIONS: ALL except du, dl, ci, scr

INDICATORS: None affected

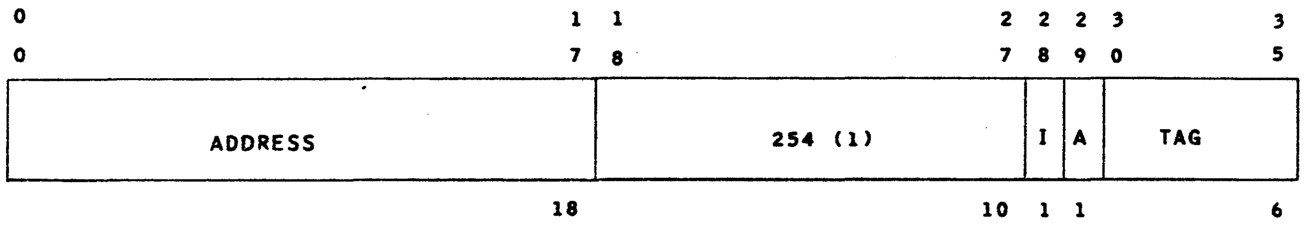
NOTES: The contents of the associative memory remains unchanged.

The associative memory is ignored during virtual-to-real address preparation (forced to "no match").

Attempted execution in normal or BAR modes causes an illegal procedure fault.

Attempted repetition with rpt,rpd, or rpl instructions causes an illegal procedure fault.

3.14.5.4 sstr Store Segment Descriptor Register



SUMMARY: This instruction stores 16 double words from the selected level (j) of the directory of the Segment Descriptor Word associative memory. There are four levels.

Level j is selected by C(TPR.CA)$_{11,12}$

For i=0,1,...15

C(SDWAM(i,j).ADDR)-> C(Y-block32+i)$_{0,23}$

C(SDWAM(i,j).R1,R2,R3)-> C(Y-block32+i)$_{24,32}$

000-> C(Y-block32+i)$_{33,35}$

0-> C(Y-block32+i)$_{36}$

	C(SDWAM(i,j).BOUND)-> C(Y-block32+i)$_{37,50}$
	C(SDWAM(I,J).R,E,W,P,U,G,C)-> C(Y-block32+i)$_{51,57}$
	C(SDWAM(I,J.).CALL LIMIT)-> C(Y-block32+i)$_{58,71}$
MODIFICATIONS:	All except du,dl,ci,sc,scr
INDICATORS:	None affected
NOTES:	The contents of the associative memory remains unchanged.
	The associative memory is ignored during virtual-to-real address preparation (forced to "no match").
	Attempted execution in normal or BAR modes causes an illegal procedure fault.
	Attempted repetition with rpt,rpd, or rpl instructions causes an illegal procedure fault.

4.0 VIRTUAL ADDRESS FORMATION

4.1 DEFINITION OF VIRTUAL ADDRESS

The virtual address in the MULTICS processor is the user's specification of the location of a data item in the MULTICS virtual memory. Each reference to the virtual memory for operands, indirect words, indirect pointers, operand descriptors, or instructions must provide a virtual address. The hardware and the operating system translate the virtual address into the true location of the data item and assure that the data item is in main memory for the reference.

The virtual address consists of two parts, an effective segment number and an offset or computed address. The value of each part is the result of the evaluation of a hardware algorithm of one or more terms. The selection of the algorithm is made by the use of control bits in the instruction word; for example, bit 29 for modification by pointer register and bits 30-35 (the TAG field) for modification by index register of indirect word. For certain modifications by indirect word, the TAG field of the indirect word is also treated as an address modifier, thus establishing a continuing "indirect chain". Bit 29 of an indirect word has no meaning in the context of virtual address formation.

The results of evaluation of the virtual address formation algorithms are stored in temporary registers used as working registers by the processor. The effective segment number is stored in the temporary segment register, TPR.TSR. The offset is stored in the computed address register, TPR.CA. When each virtual address computation has been completed, C(TPR.TSR) and C(TPR.CA) are presented to the appending unit for translation to a 24-bit absolute main memory address.

4.2 TYPES OF VIRTUAL ADDRESS FORMATION

There are two types of virtual address formation. The first type does not make explicit use of segment numbers. The algorithms produce values for the computed address, C(TPR.CA), only. The effective segment number in C(TPR.TSR) does not change from the value used to fetch the current instruction. In this case, all references are said to be "local" to the procedure segment pointed to by the procedure pointer register (PPR).

The second type makes use of a segment number in an indirect word-pair in main memory or in a pointer register (PR_n_). The algorithms produce values for both the effective segment number, C(TPR.TSR), and the computed address, C(TPR.CA). The effective segment number in C(PTR.TSR) may change and, if it changes, references are said to be "external" to the procedure segment.

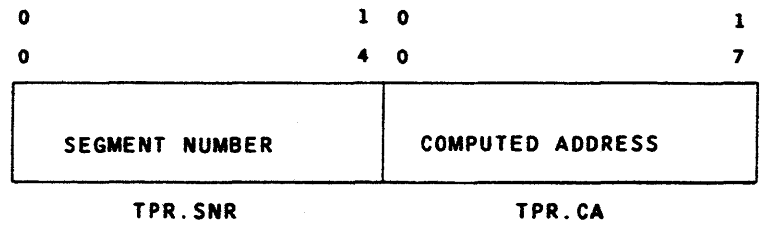
Both types of virtual address formation for the operand of a basic or EIS single-word instruction begins with a preliminary step of loading TPR.CA with the ADDRESS field of the instruction word. This preliminary step takes place during instruction decode.

The two types of virtual address formation can be intermixed. In cases where virtual address calculations are chained together through pointer registers or indirect words, each virtual address is translated to a 24-bit absolute main memory address to fetch the next item in the chain.

HONEYWELL CONFIDENTIAL & PROPRIETARY

5.0 ADDRESS APPENDING

At the completion of the formation of the virtual memory address an effective segment number (segno) is in the segment number register of the temporary pointer register (TPR.SNR) and a computer address (offset) is in the computed address register of the temporary pointer register (TPR.CA).



There follow a number of different and distinct append unit cycles depending on whether the segment is paged or unpaged and if the required SDW's or PTW's are in the associative memory or must be fetched from storage.

The operation of the appending unit is shown in the following table and flowchart. Flowchart assumes that directed faults, store faults, and parity faults do not occur.

A segment boundary check is made in every cycle except FSDWP. If a boundary violation is detected, an access violation, out of segment bounds fault is generated and the execution of the instruction interrupted. The occurrence of any fault interrupts the sequence at the point of occurrence. The operating system software should store the control unit data for possible later continuation and attempt to resolve the fault condition.

The value of the associative memories may be seen in the flowchart by observing the number of appending unit cycles and resulting main memory cycles bypassed if an SDW or PTW is found in the associative memories.

There are nine different appending unit cycles that involve accesses to main memory. Two of these (FANP, FAP) generate the 24-bit absolute main memory address and initiate a main memory access for the operand, indirect word, or instruction pair; five (FSDWNP, FSDWP, PTW, PTW2, and DSPTW) generate a main memory access to fetch an SDW or PTW; and two (MDSPTW and MPTW) generate a main memory access to update page status bits (PTW.U and PTW.M) in a PTW. The cycles are defined in Table 5-1, Appending Unit Cycle Definitions and Figure 5-1, Appending Unit Operation Flowchart.

TABLE 5-1

APPENDING UNIT CYCLE DEFINITIONS

Cycle Name	Function
FANP	Final address nonpaged Generates the 24-bit absolute main memory address and initiates a main memory access to an unpaged segment for operands, indirect words, or instructions.
FAP	Final address paged Generates the 24-bit absolute main memory address and initiates a main memory access to a paged segment for operands, indirect words, or instructions.
FSDWNP	Fetch SDW nonpaged Fetches an SDW from an unpaged descriptor segment.
FSDWP	Fetch SDW paged Fetches an SDW from a paged descriptor segment.
PTW	PTW fetch Fetches a PTW from a page table other than a descriptor segment page table and sets the page accessed bit (PTW.U).
PTW2	Prepage PTW fetch Fetches the next PTW from a page table other than a descriptor segment page table during hardware prepaging for certain uninterruptible EIS instructions. This cycle does <u>not</u> load the next PTW into the appending unit. It merely assures that the PTW is not faulted (PTW.F = 1) and that the target page will be in main memory when and if needed by the instruction.
DSPTW	Descriptor segment PTW fetch Fetches a PTW from a descriptor segment page table.

TABLE 5-1 (continued)

APPENDING UNIT CYCLE DEFINITIONS

Cycle Name	Function
MDSPTW	Modify DSPTW Sets the page accessed bit (PTW.U) in the PTW for a page in a descriptor segment page table. This cycle always immediately follows a DSPTW cycle.
MPTW	Modify PTW Sets the page modified bit (PTW.M) in the PTW for a page in other than a descriptor segment page table.

HONEYWELL CONFIDENTIAL & PROPRIETARY

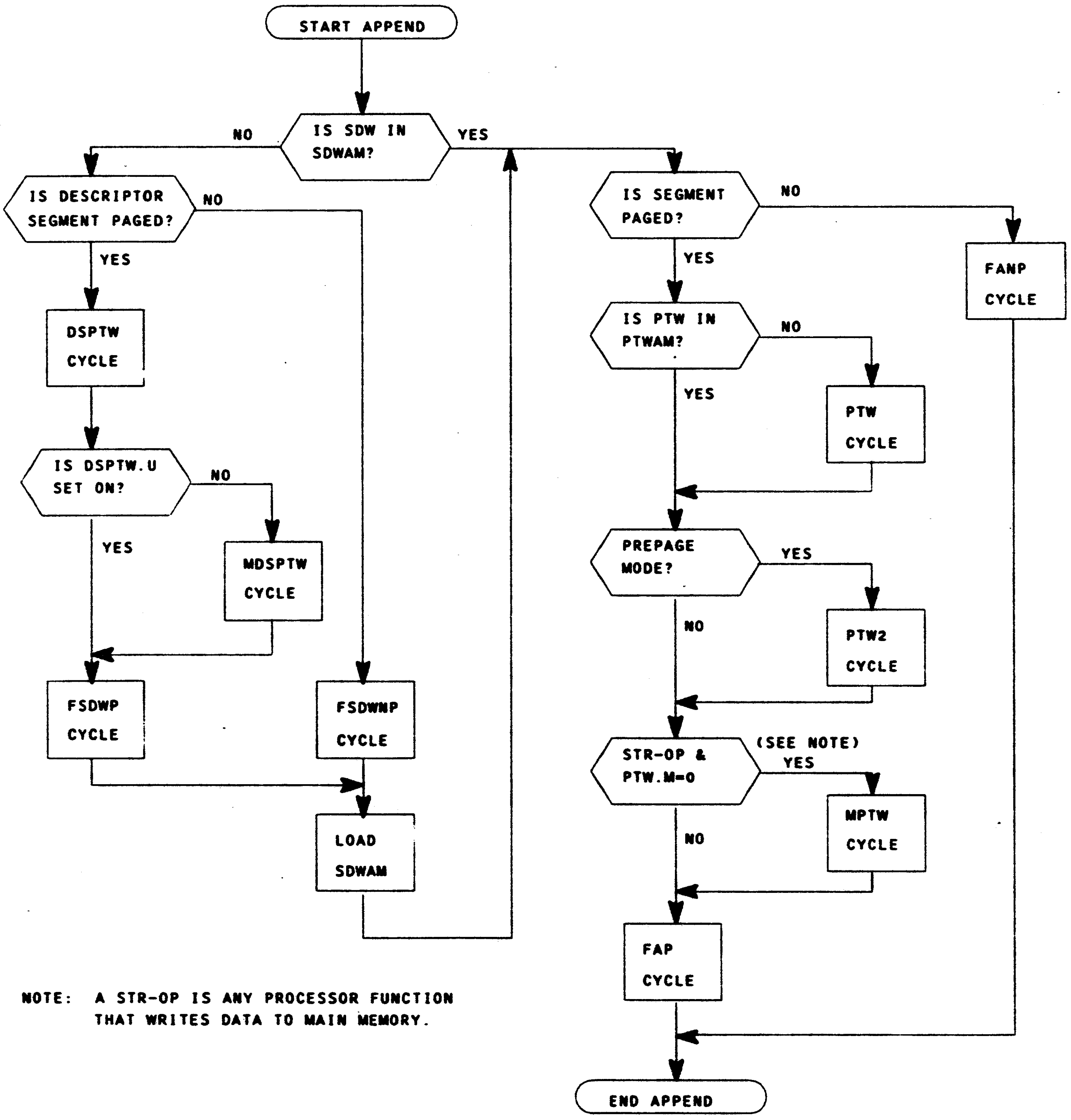


FIGURE 5-1. APPENDING UNIT OPERATION FLOWCHART

In the MULTICS mode the appending unit is utilized during address preparation for each main memory access. Use of the PTWAM and SDWAM depend on the unpaged list in the Descriptor Segment Base Register (DSBR). If either the SDW or PTW are in their respective associative memory it is not necessary to access main memory, the required information is retrieved from the associative memory.

5.1 UNPAGED DSBR.U=1

Figure 5-2 illustrates address preparation when the current descriptor segment is unpaged.

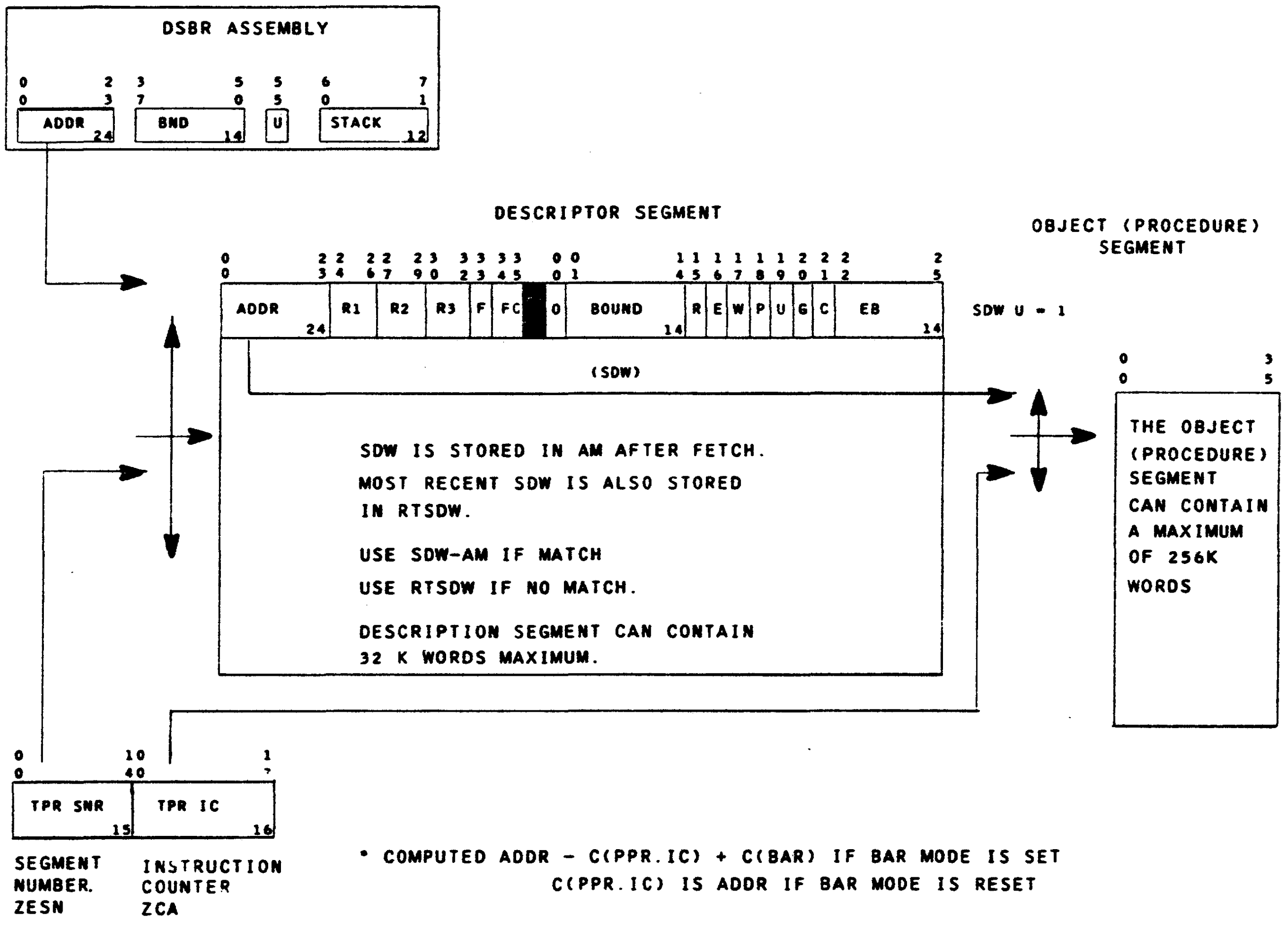

DSBR ASSEMBLY
ADDR 24
BND 14
U
STACK 12
DESCRIPTOR SEGMENT
ADDR 24
R1
R2
R3
F
FC
0
BOUND 14
R
E
W
P
U
G
C
EB 14
(SDW)
SDW U = 1
SDW IS STORED IN AM AFTER FETCH.
MOST RECENT SDW IS ALSO STORED IN RTSDW.
USE SDW-AM IF MATCH
USE RTSDW IF NO MATCH.
DESCRIPTION SEGMENT CAN CONTAIN 32 K WORDS MAXIMUM.
OBJECT (PROCEDURE) SEGMENT
THE OBJECT (PROCEDURE) SEGMENT CAN CONTAIN A MAXIMUM OF 256K WORDS
TPR SNR 15
TPR IC 16
SEGMENT NUMBER. ZESN
INSTRUCTION COUNTER ZCA
* COMPUTED ADDR - C(PPR.IC) + C(BAR) IF BAR MODE IS SET
C(PPR.IC) IS ADDR IF BAR MODE IS RESET

FIGURE 5-2. ADDRESS PREPARATION DESCRIPTOR SEGMENT - UNPAGED

When the unpaged bit = 1 the DSBR.ADDR field contains the absolute address of the origin (base) of the current descriptor segment. The segment number PPR.PSR is added to DSBR.ADDR to form the absolute address of the requested Segment Descriptor Word (SDW).

The descriptor segment described by TPR.SNR (ZESN00-14) is 32K words long. The SDW address is checked against the SDWAM directory. If the SDW is not in the AM (no match) it is fetched from main memory and placed in the AM. In this case the SDW will also be placed in the RTSDW and used from there. If the SDW is in the associative memory (a match) it will be used from the AM.

DSBR.BND is the 14 most-significant bits of the highest Y-Block 16 address of the descriptor segment that can be addressed without causing an access violation or out of segment bounds fault. (See Section 2.0 ,Definition of Notation and Symbols for Y-Block$\underline{n}$ description).

The SDW.ADDR field contains the absolute address of the origin (base) of the current Object (Procedure) Segment. The instruction counter PPR.IC is added to the SDW.ADDR to form the absolute address of the instruction or operand currently requested from main memory. PPR.IC is an 18 bit field this allows the Object (Procedure) Segment to be 262K words long.

SDW.BOUND contains the 14 high order bits of the last Y-block 16 address within the segment that can be referenced without an access violation, out of segment bound fault.

5.2 PAGED DSBR.U = 0

Figure 5-3 illustrates address preparation when the current descriptor segment is paged. When the virtual address has been formed, the portion of the virtual address in TPR.SNR and TPR.CA are interpreted differently. The Segment Number and Computed Address fields are both divided into subfields containing a Page Number and a Word Number.

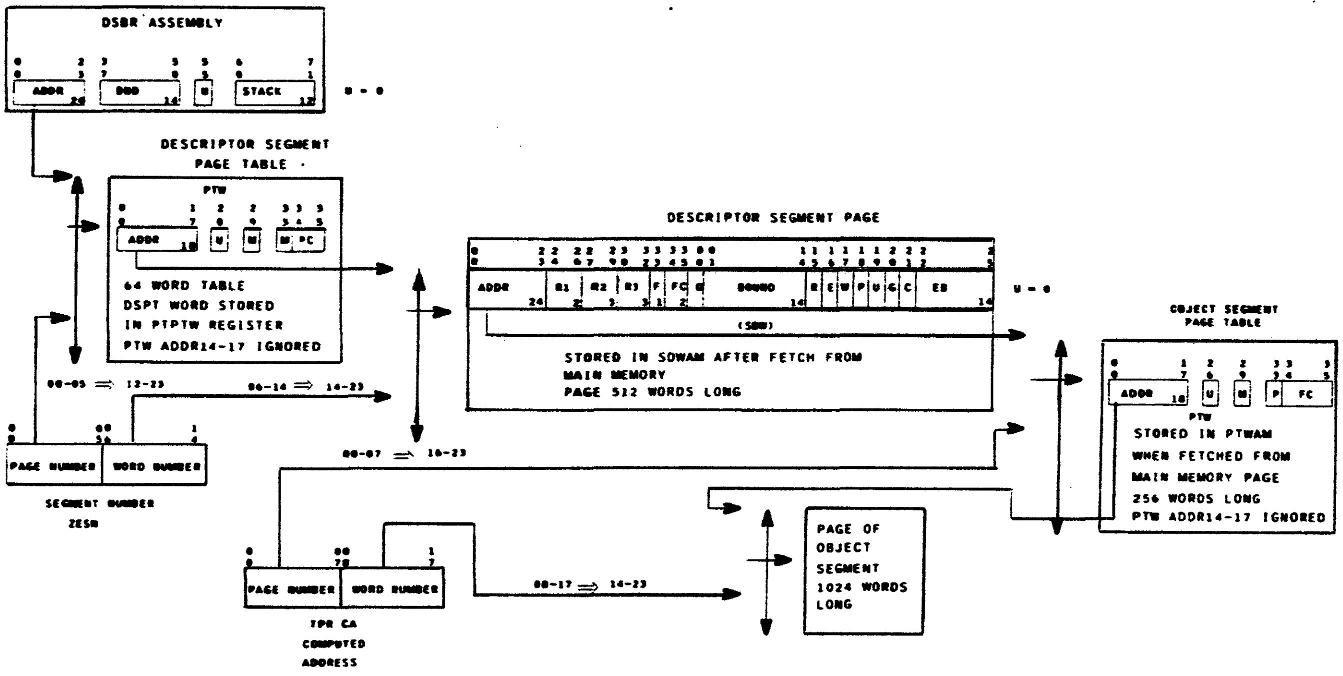


FIGURE 5-3. ADDRESS PREPARATION DESCRIPTOR SEGMENT - PAGED

When the unpaged bit of the DSBR.U = 0 the DSBR.ADDR field contains the absolute address of the origin (base) of the current Descriptor Segment Page Table. The TPR.SNR page number field is added to the DSBR.ADDR to form the absolute address of the requested DSPT word. The Descriptor Segment Page Table described by ZESN 00-05 is 64 words long.

The addressed Descriptor Segment Page Table Word is fetched from main memory and stored in the RTPTW register. The DSPTW is not stored in the associative memory. As long as the processor remains in the same segment the DSPTW remains the same. When a new segment is entered a new DSPTW is requested.

The DSPT.ADDR field bits 00-13 contain the absolute address of the base of the Descriptor Segment Page. The word number from TPR.SNR bits 06-14 are appended to DSPT.ADDR field bits 00-13 as bits 12-22 to form the absolute address of the SDW in the Descriptor Segment Page.

The absolute address of the Descriptor Segment Page Word (DSPW) is checked against the SDWAM directory. If the word is not in (no match) the AM it is fetched from main memory and placed in the SDWAM. The SDW is also placed in the RTSDW register for current use. If the DSPW is in the AM (a match) it will be used from the AM. TPR.SNR bits 06-14 define a descriptor segment page that is 10242 words long or 512 SDW's.

The SDW.ADDR field contains the absolute address of the base of the Object Segment Page Table. The page number from TPR.CA bits 00-07 are added to SDW.ADDR field as bits 16-23 to form the absolute address of the PTW in the Object Segment Page Table.

The absolute address of the Object Segment Page Table Word (OSPTW) is checked against the PTWAM. If the word is not in the AM (no match) it is fetched from main memory and placed in the PTWAM. The word is also placed in the PTPTW register for current use. If the OSPTW is in the PTWAM (a match) it will be used from the AM.

The PTW.ADDR field contains the absolute address of the base (origin) of the Object (Procedure) Segment Page. This page contains instructions or data for the object program. The word number from TPR.CA bits 08-17 is added to the PTW.ADDR to form the absolute address of the requested instruction or operand.

HONEYWELL CONFIDENTIAL & PROPRIETARY

6.0 DPS 8/70M PHYSICAL HARDWARE

The DPS 8/70M design goals were to improve performance from the L68, introduce remote maintenance capability and reduce cost from L68.

The various approaches used to achieve these goals are as follows:

- o High density hard copper printed wiring boards and MSI chips are being used where practical to reduce cost and improve performance.
- o Schottky chips, 8K hardware controlled cache and 64 word associative memory are used to increase performance.
- o A microprocessor and VIP replace the maintenance panel and provide improved control and display capability over the L68 maintenance panel.
- o A low profile cabinet was adopted for aesthetic reasons and as a cost improvement.

6.1 LOGIC BOARD/BACKPANEL COMPARISON

The L68 used a total of 100 boards when eight active ports were in use. There were 67 board types; 42 board types were common to the L66 and 25 board types were unique to the L68.

On the DPS 8/70M a total of 64 boards are used when four active ports are in use. There are 51 board types; 31 board types are common to the DPS 8/70 and 20 board types are unique to the DPS 8/70M. The hard copper boards used in the DPS 8/70M have voltage and ground planes. On each side of these planes are X and Y layers for IC interconnect. On the outside of the board there are random connect layers to handle the X-Y plane can't connect.

The L68 was a ten bucket, double bay, high profile cabinet. The DPS 8/70M is in a single bay, low profile cabinet that contains five buckets. Each bucket can contain a maximum of 19 multi-layer hard copper boards. The backpanels use 3 X and Y planes to interconnect a card cage and jumper blocks between cages.

6.2 DPS 8/70M BLOCK DESIGN

Figure 6-1 is a high level block diagram of the DPS 8/70M CPU.

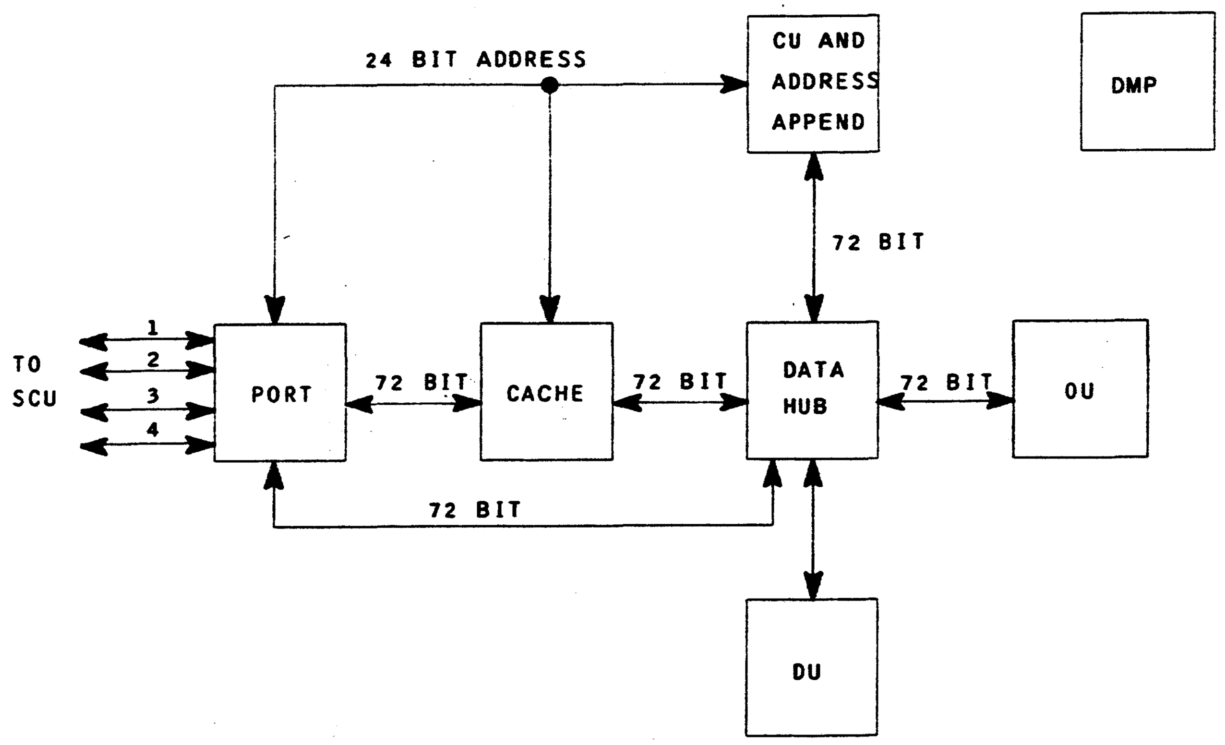


FIGURE 6-1. DPS 8/70M BLOCK DIAGRAM

The changes took place in the port and data hub areas, as well as in the CU and address append units. The OU and DU were unchanged The Diagnostic Maintenance Panel (DMP) board replace the L68 maintenance panel.

6.3 DIAGNOSTIC MAINTENANCE PANEL (DMP) FEATURES

The DMP and a VIP terminal are used to replace the lights and switches on the L68 maintenance panel. The actual use of the DMP instruction set and display formats, are explained in the Freestanding DPS 8 System Test and Repair Manual, 58009928.

The DMP is located on the ETCMP board and is composed of an INTEL 8085A microprocessor with 32K bytes of EPROM and 8K bytes of RAM. Software for the INTEL 8085A is contained in the EPROM. The microprocessor controls the interfaces to the DPS 8/70M and two USARTs that provide a local and remote communication interface to VIPs or the Diagnostic Maintenance Unit (DPU).

Figure 6-2, DPS 8/70 Maintenance Controller Functional Block Diagram, applies to both the DPS 8/70 and 8/70M processors.

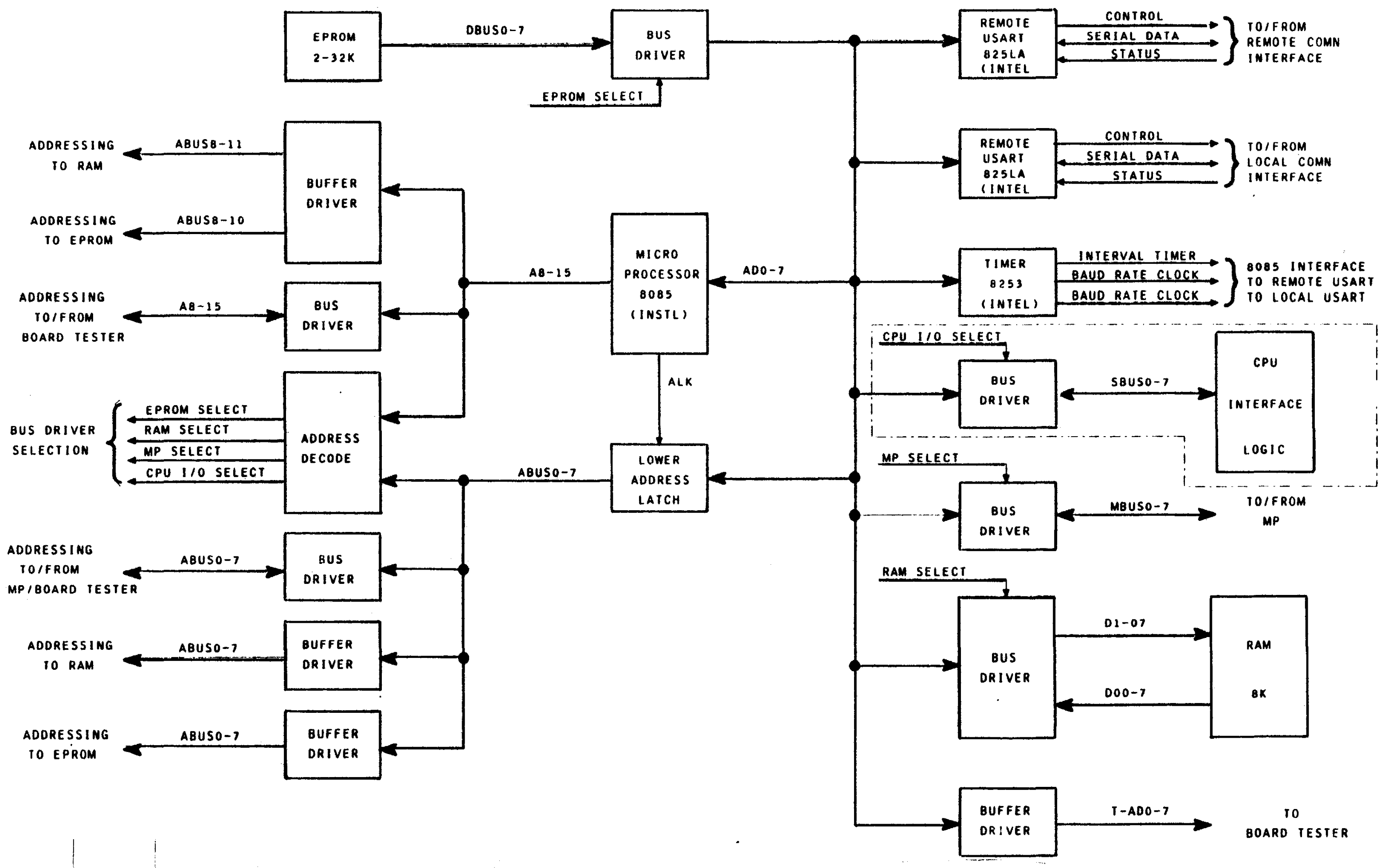


FIGURE 6-2. DPS 8/70 MAINTENANCE CONTROLLER FUNCTIONAL BLOCK DIAGRAM

The INTEL USARTs provide communication line-type interfaces to a local VIP or to a modem that then provides communications to a VIP located at a remote location. The EIA RS232C communications interface supplied by the USART requires a DC/DC converter to convert +5 volts to ±12 volts at 200 ma. This converter is mounted on the backpanel. The interfaces labeled "to board tester" are provided to aid testing of the ETCMP board on the board tester.

The logic enclosed by dotted lines on Figure 6-2, DPS 8/70 Maintenance Controller Functional Block Diagram is expanded on Figure 6-3, DPS 8/20/CPU Interface Logic. This includes the bus drivers, the S-BUS 0-7 and the CPU interface logic.

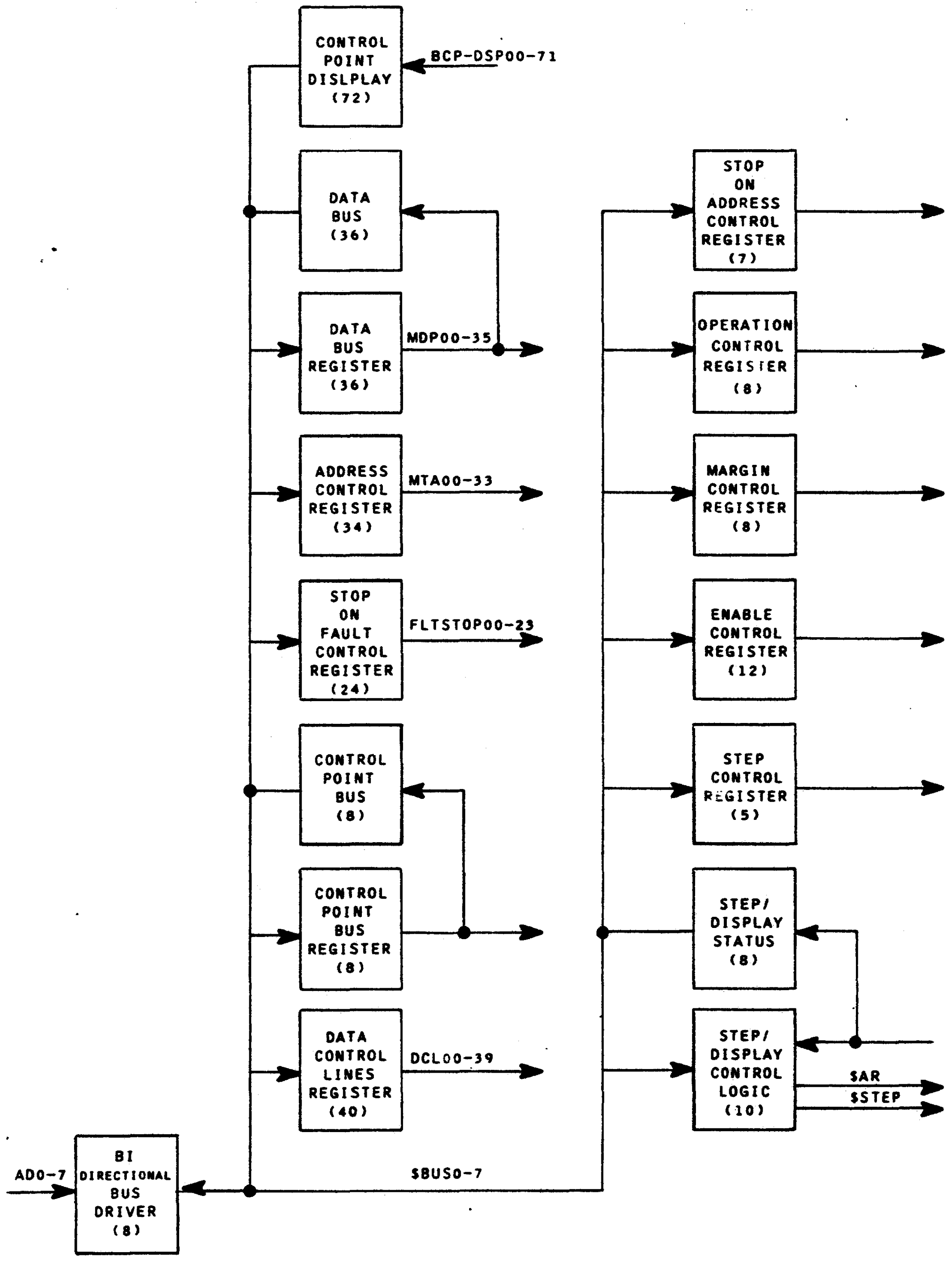

CONTROL POINT DISLPLAY (72)
BCP-DSP00-71
DATA BUS (36)
DATA BUS REGISTER (36)
MDP00-35
ADDRESS CONTROL REGISTER (34)
MTA00-33
STOP ON FAULT CONTROL REGISTER (24)
FLTSTOP00-23
CONTROL POINT BUS (8)
CONTROL POINT BUS REGISTER (8)
DATA CONTROL LINES REGISTER (40)
DCL00-39
BI DIRECTIONAL BUS DRIVER (8)
AD0-7
$BUS0-7
STOP ON ADDRESS CONTROL REGISTER (7)
OPERATION CONTROL REGISTER (8)
MARGIN CONTROL REGISTER (8)
ENABLE CONTROL REGISTER (12)
STEP CONTROL REGISTER (5)
STEP/ DISPLAY STATUS (8)
STEP/ DISPLAY CONTROL LOGIC (10)
$AR
$STEP

FIGURE 6-3. DPS 8/70 CPU INTERFACE LOGIC

The Control Point Display BCP-DSP00-71 is from the DU. The data bus and Data Bus Register provide a two-way interface to the data hub and a wraparound path for DMP self test.

The Address Control Register provides the control point to set up stop on address and provides the Maintenance Panel Address to the CC board. The Control Point Bus and Control Point Bus Register take the place of control point lights and switches that were on the L66 maintenance panel.

Data Control Lines Register. These 40 lines control what the processor displays. The large number of lines made less encoding and decoding necessary in allowing display of internal processor points.

The Operations Control Register is used for functions such as Repeat Execute, Execute Fault, etc. The Margin Control Register allows setting fast/slow timing margins on the DU,OU,CU and Append Units. The Enable Control Register is used to enable and control Cache and the Associative Memories.

The Step Control Register controls putting the OU,DU,CU,AU into and out of step mode.

Other than the configuration display, the displays should not be requested while the DPS 8/70M is running. The DMP will put the processor in step, then request the required display information, format it, send it to the VIP and when done, take the processor out of step back to run. In a GCOS or MULTICS operating environment the system does not always restart correctly. In a multiprocessor environment it is harder to restart successfully.

6.3.1 DMP TO VIP CONNECTION

The following block diagram Figure 6-4, illustrates the DMP connection to local and remote VIPs.

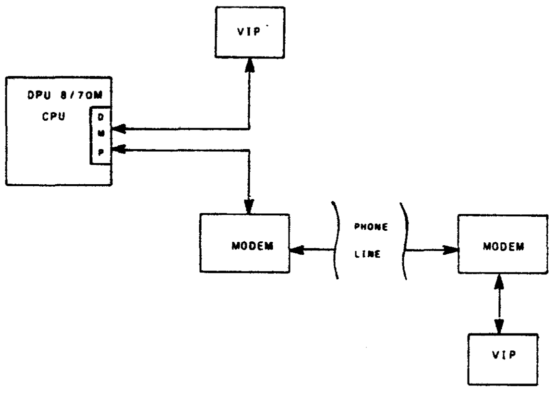


FIGURE 6-4. DMP TO VIP CONNECTION

The DMP remote communications interface is enabled and set up by the VIP and the local communications interface. When the remote interface is in control, the local VIP can copy messages sent between the DMP and the remote VIP as the remote VIP accesses the DMP.

6.3.2 DMP TO DPU CONNECTION

The DPU (Diagnostic Processor Unit) is a Level 6 which allows the VIP to be switched between system modules that contain DMPs and LCCs. The DPU also provides interconnection to phone lines. The block diagram in Figure 6-5 illustrates these connections.

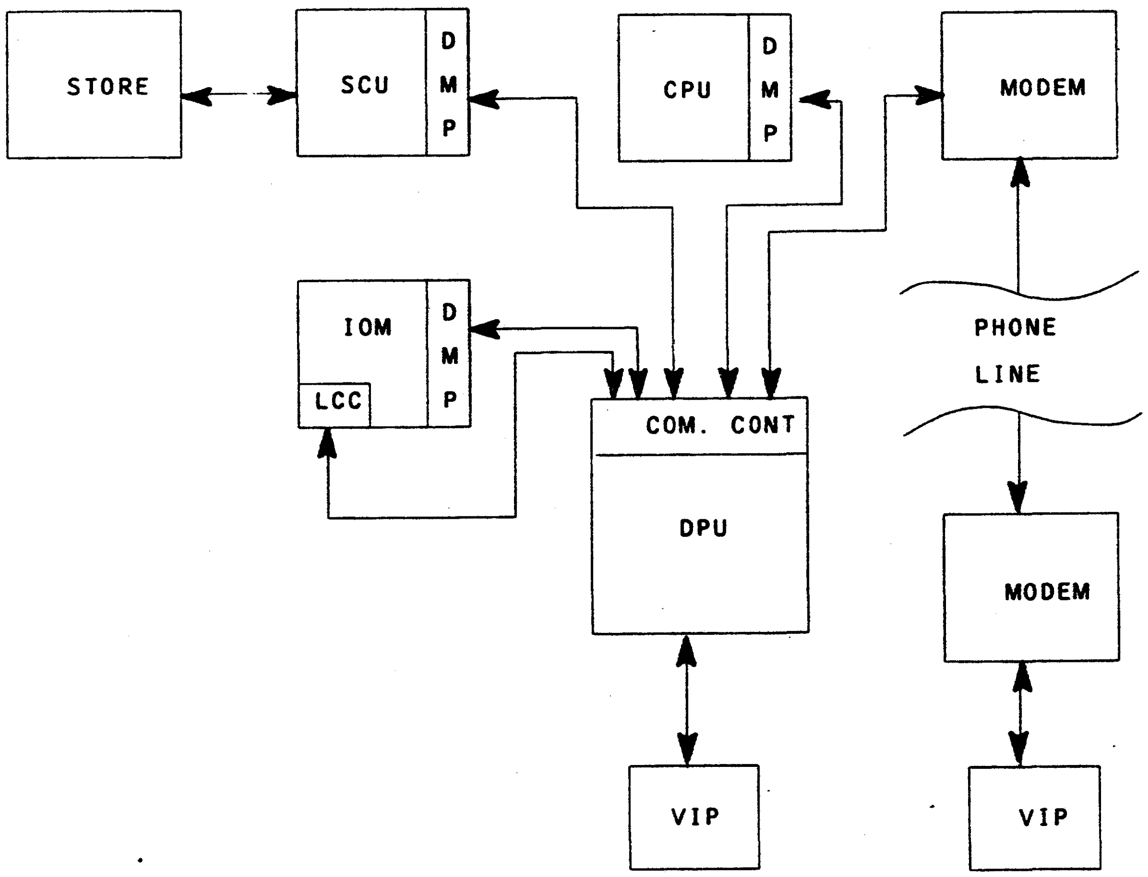


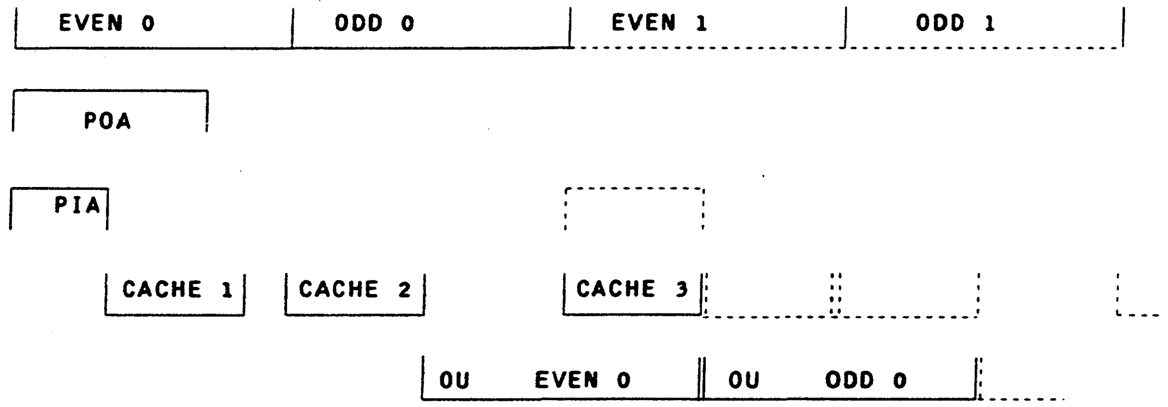
FIGURE 6-5. DMP TO DPU CONNECTION

The DPU can provide connection for up to 16 DMPs or LCCs including one remote connection to a modem. The DPU uses the local communications interface of the DMP.

7.0 BASIC OPERATION

7.1 INSTRUCTION STREAM SEQUENCE

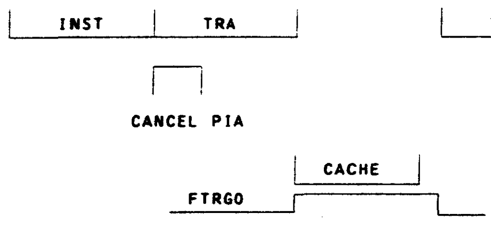
On the DPS 8/70M system, the processor attempts to maintain the instruction buffers full at all times. PIA (Prepare Instruction Address) if set will stay set except for a 710 instruction. A 710 instruction is the only one that will cancel a PIA. The time for an instruction is about 260 nanoseconds, cache can operate in about 100 nanoseconds See the following diagram of the Basic Instruction Sequence:



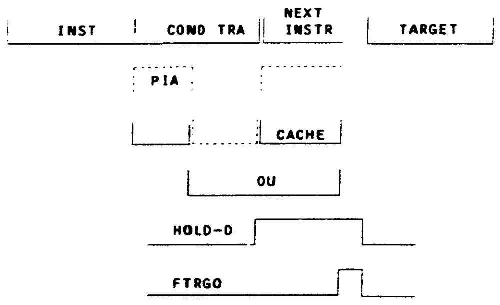
If the processor starts off with EVEN 0 instruction, it will start PIA and POA cycles. The Processor gets through the PIA which is very short in less than 100 nanoseconds. In another 120 nanoseconds the EVEN 1 and ODD 1 instruction pair for the PIA have been pulled from cache in the cache 1 cycle. Cache is again available. By this time the POA (Prepare Operand Address) cycle is done and $AR is ready, so the processor can start on the operand cache cycle, cache cycle 2. If it is assumed that the instructions are buffers full at this point, $AR will let the next POA cycle start to pull the next OU ODD 0 operand during cache 3 cycle. The intention was to have two cache cycles take place during an OU or DU cycle. The above is in GCOS mode. When the processor is in MULTICS mode, the appending unit will take a little bit longer and will expand the PIA and POA cycles; the rest will remain the same. If the appending unit has to go into a paging cycle, it will insert its cycles before $AR is reached, during the POA or PIA cycles. Everything would just stretch out. The difference between GCOS and MULTICS are the extra cycles involved.

7.1.1 TRANSFER SEQUENCES

One of the biggest changes made between the L68 and DPS 8/70M processors was in the way the transfer instructions are handled. See the following diagram of the fixed transfer sequence. The basic assumption is that the processor works out of cache. Changes were made such that if the processor were not operating out of cache, it would become very slow. This is due to the excessive fetches made from backing store (main memory) to keep the instruction buffers full. When using cache the time is hidden.



When the processor is executing a fixed transfer, PIA will be cancelled if it comes up. Cache is accessed at $AR time, FTRGO is set and when the cache cycle is finished, the next instruction is started. The transfer then is about one and one-half cycles long.

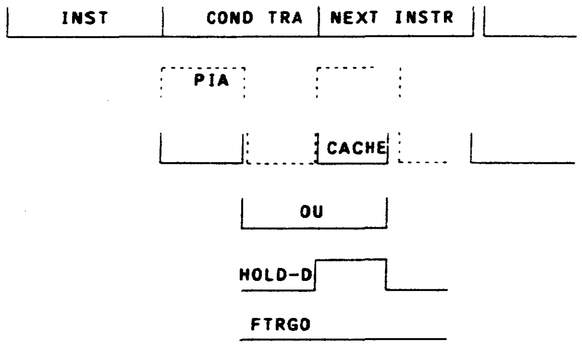


Conditional transfers are more of a problem. When the processor enters a conditional transfer and PIA comes up, the processor will complete that PIA. If the instructions for that PIA come from cache, it will not cost any additional time; if the instructions come from main memory, it will cost time then but it is likely the process will eventually use the instructions so it may as well get them.

In conditional transfers, at this point the processor goes to cache and brings out the target pair. See the following diagram of conditional transfer-go. At the same time the processor had stepped into the next instruction, the no go path. HOLD-D is set and will remain up until the processor finds out if the transfer is a go or no go. The processor already has the no go instruction so why not get into it. While the processor starts the no go instruction the cache cycle starts to fetch the target instruction. If the processor had been executing an OU instruction before the conditional transfer and the conditional transfer needed the OU indicators, the OU would be about done by this time. If the transfer is a go, FTRGO is set and the processor starts the target instruction. In this case the transfer is about one and one-half cycles long.

For a conditional transfer that goes out of bounds the store fault is saved and only used if the transfer path that goes out of bounds is taken.

The following illustrates a conditional transfer that is no go. Everything else is the same. At the end, when the decision is made that the transfer is no go, HOLD-D is reset and FTRGO is not set. In this case the transfer instruction is just a little over an instruction long.



7.2 8K CACHE

7.2.1 OPERATION DESCRIPTION

All DPS 8/XXM processors are fitted with a cache memory. The operation of this cache memory is described as follow:

Cache memory is a high speed buffer memory, in this case, distributed within the processor cabinet. Cache memory holds operands and instructions that are expected to be used by the process currently under execution by the processor. Contents of cache are continually being changed and updated.

Cache is made up of four major sections: Primary Cache Directory, Duplicate Cache Directory, Cache Data Storage and Cache Control. See Figure 7-1, Cache Subsystem Block Diagram for a basic block diagram of the cache subsystem.

The ETCCY board contains the Duplicate Cache Directory and duplicate full/empty array. The ETCCZ board contains the SCU interface for change notification address as well as the stack for changed addresses and control for duplicate directory. The ETCCC board contains the Primary Cache Directory, cache function control, primary full/empty array and LRU array. The ETCCD boards contain Cache Data Storage.

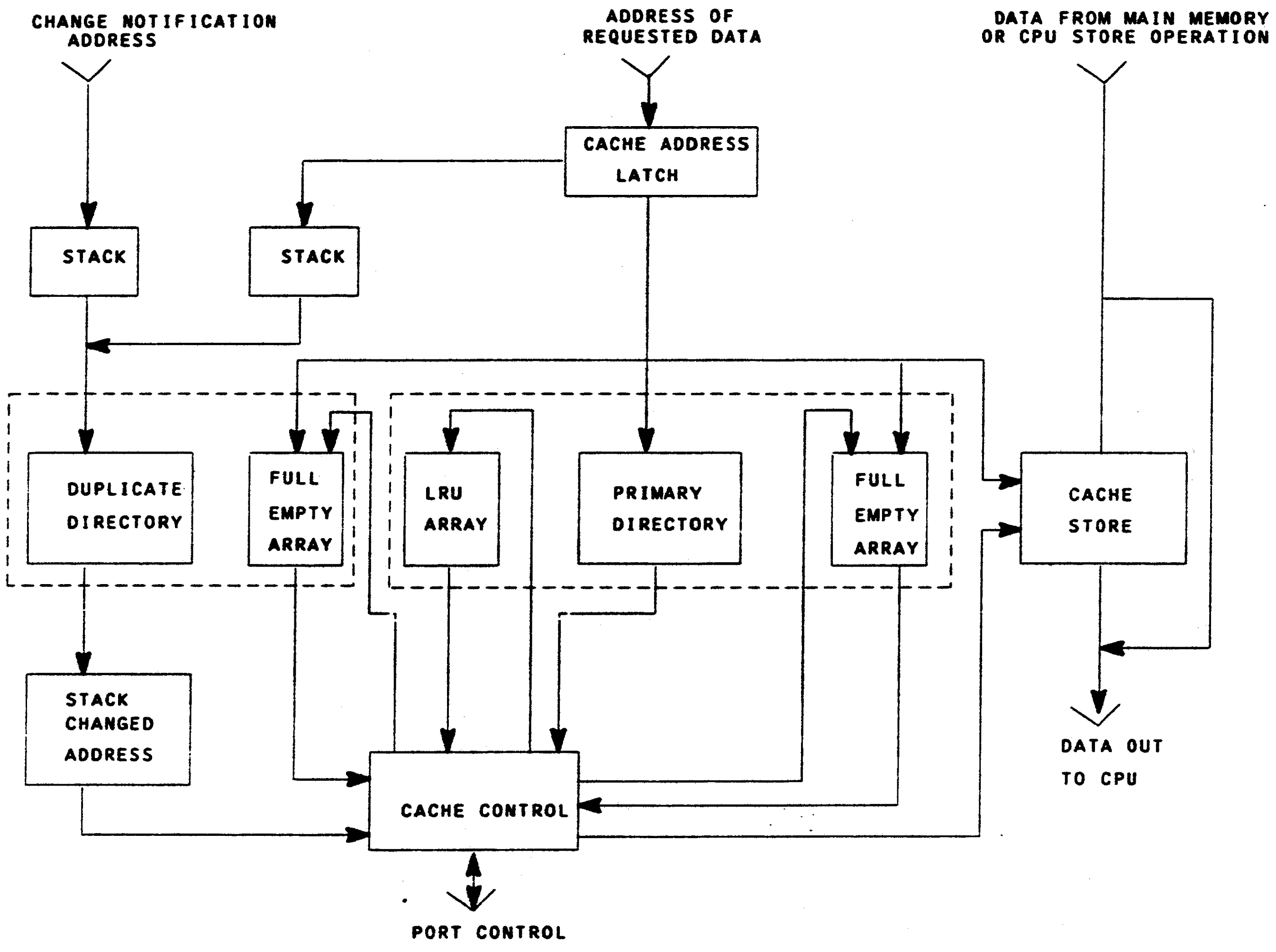


FIGURE 7-1. CACHE SUBSYSTEM BLOCK DIAGRAM

Cache contains 8192 36-bit words of high speed buffer for operands or instructions and the means of controlling this storage. Each section of cache, Primary Directory, Duplicate Directory and Data Storage are organized into four levels. Each level is further divided into 512 columns.

In Cache Data Storage, each column location contains space for a four word block of data so 2048 words are stored on each level.

In the directories, each column location contains a partial main memory address of the data words stored in Cache Data Storage.

Cache memory is also divided into two halves, levels one and two comprise the first half. Levels three and four comprise the second half. This allows the CPU to utilize either half independently so operation may continue if there is a failure in either half of the cache.

The purpose of the primary directory is to store a partial address that identifies the data words in main memory that are stored in cache data store.

The duplicate cache directory is part of a selective clearing system. The duplicate directory is organized the same as the primary directory in levels and columns and contains duplicate partial main memory addresses of all data stored in cache data storage. Use of the duplicate directory allows the cache subsystem to clear only those locations in cache that have been modified in main memory. Instructions or operands requested by the CU are supplied by cache data storage or main memory. On a cache hit the instruction or operand is supplied by cache. On a cache miss a four word block of data containing the requested word is read from main storage and stored in cache.

Store operations by the CPU always store data into main memory. A store operation involves a simultaneous store into both main memory and cache.

When a cache miss is encountered, the full/empty and LRU (Least Recently Used) arrays are used to determine which column level will store the four word block from main memory. First the full/empty array is examined to check for an empty level at the selected column. If there is no empty level, the LRU array is evaluated to find the least recently used level of the selected column. The four word data block replaces the data that was at the selected column and level.

Cache and main memory access cycles are started at the same time, after a complete address is prepared. These cycles are started in parallel at $AR time. For cache the directory look up cycle takes place first. If there is a MATCH, the main memory cycle is aborted and data is supplied from cache data storage. If there is not a match (MATCH), the main memory cycle continues and a four word block of data is stored in cache data memory.

7.2.2 DIRECTORY AND CACHE ADDRESSING

The main storage and cache access cycles are started at the same time, after the absolute address has been prepared and registered in the RADDR 00-23 address register. Figure 7-2 illustrates directory and cache memory addressing.

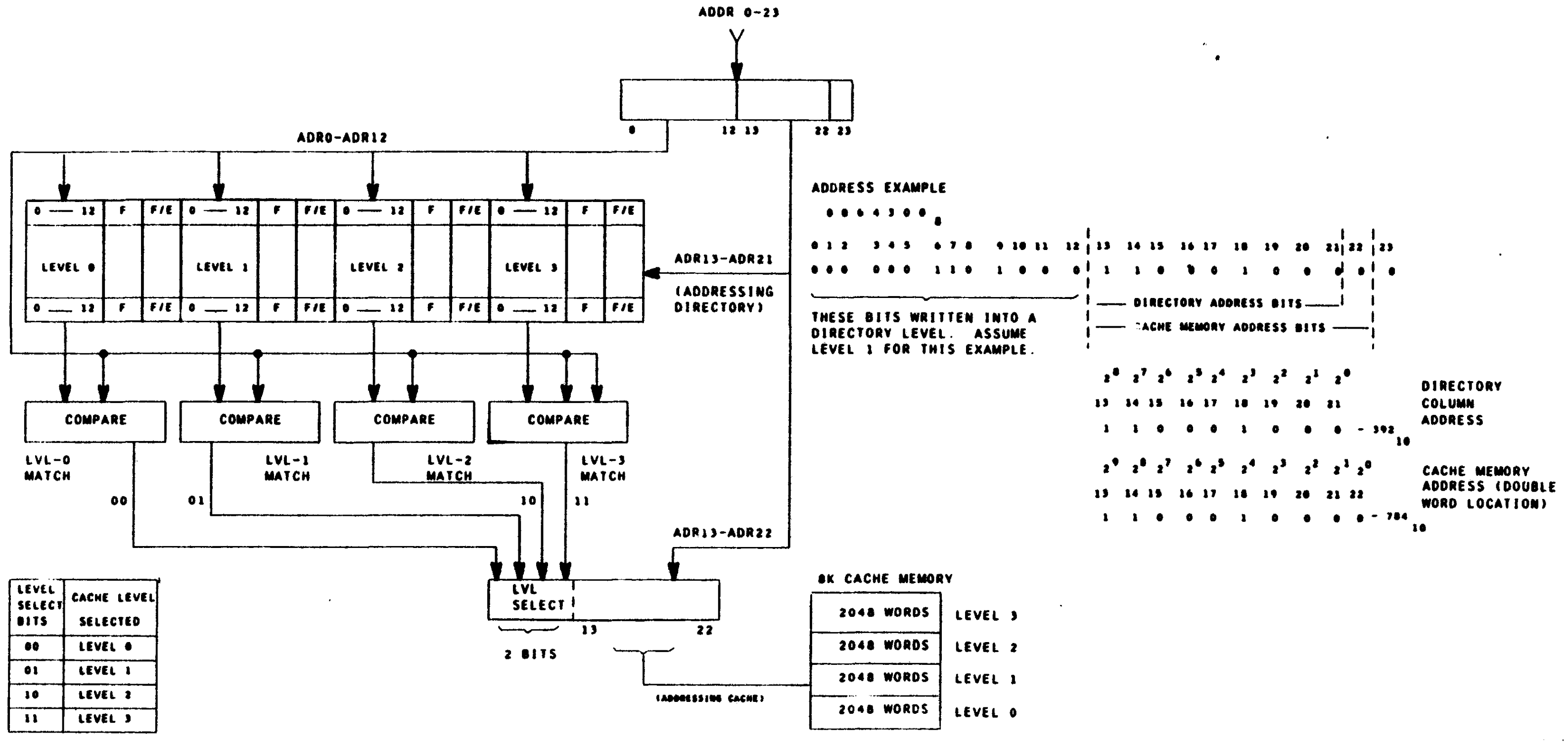
The absolute address is divided into two fields A00-12 and A13-22, bit A23 is not used.

On a write to cache operation, when a four word block of data from main memory is stored in cache memory, the low order bits A13-21 are used to address the primary directory column. The bits A22 and A23 are not needed to address the directory columns. The high order address bits A00-12 are written into the primary directory in any one level 0 through 3 depending on the full/empty bits and LRU settings for the addressed column.

The data words from main memory are stored in cache memory in two, two word storage cycles. Address bits A13-22 are used to identify the column and word pair being stored and two bits of level number identify which level the two, two word pairs will be stored in. The level corresponds to the level selected by the Full/Empty bits and LRU array.

On a read from cache operation address bits A13-21 are used to address the primary directory column. The address bits A00-12 previously stored at each of the four levels of the addressed directory column are compared with the current address bits A00-12. If there is a comparison (level match or bit) the requested data is in cache memory and the LRU code is set for the hit level. On hits the LRU update is one cycle behind the associated directory access. On a miss, the LRU update is in the same cycle as the directory access. Address bits A13-22 with the two level bits are used to access cache memory and select the correct word pair.

To recap, when a complete address for a storage access cycle is prepared, the main storage and cache access cycles are started at the same time. The absolute address bits A12-A21 are used as the address to locate the column of the primary directory that may contain the remaining bits of the absolute address. The absolute address bits A00-11 are used to compare against the bits stored in each of the four levels of the primary directory. If there is a match hit on one of the levels, the requested data is stored in cache data storage at the same column and level. The column address A12-A21 along with the word pair bit A22 and the level bits are used to access the word pair in cache data storage.



LEVEL SELECT BITS	CACHE LEVEL SELECTED
00	LEVEL 0
01	LEVEL 1
10	LEVEL 2
11	LEVEL 3

FIGURE 7-2. DIRECTORY AND CACHE ADDRESSING

7.2.3 LEAST RECENTLY USED HARDWARE

When there is a cache miss and main memory is accessed, the Least Recently Used hardware selects in which level of a column to store the four word block from main memory.

The procedure implemented in the LRU hardware is to first look for any empty levels and if an empty level is found store the block in it. If there is no empty level the Least Recently Used level is determined and the level is overwritten with the block from main memory.

7.2.4 F/E BIT

When an entry is written into the directory, a full/empty bit associated with each entry is set to indicate that a valid entry exists for the column and level selected. On a compare operation, only those entries that have their F/E bit set will be compared.

When the directory is empty and loading starts, the levels are loaded sequentially with level zero being first. If the levels of that directory column are randomly empty the levels are sequentially loaded, selecting the lower number level first.

7.2.5 LRU BITS

If the Directory Column has all four levels full, the blocks are placed in the levels by use of the LRU code. The LRU array stores six bits for each column and its associated four levels. The purpose of the LRU array is to store a six bit pattern that identifies which level was least recently used. Three of the six flags are conditioned on each cache access. Each bit represents the relative time of last usage of two of the four levels in a particular column of the primary directory. The following chart illustrates the bit comparisons.

TOP			BOTTOM		
1/2	1/3	1/4	2/3	2/4	3/4

If the bit = 0, the top level is Least Recently Used. If the bit = 1, the bottom level is Least Recently Used. For example:

	1/2	1/3	1/4	2/3	2/4	3/4
IF LEVEL 1 IS LRU THE BITS =	0	0	0	X	X	X
2	1	X	X	0	0	X
3	X	1	X	1	X	0
4	X	X	1	X	1	1
IF LEVEL 1 IS MRU THE BITS =	1	1	1	X	X	X
2	0	X	X	1	1	X
3	X	0	X	0	X	1
4	X	X	0	X	0	0

X = DON'T CARE

The LRU pattern is checked to see which level was Least Recently Used. When a level is Read or Written (Used) the Most Recently Used pattern is written into the LRU bits.

The ETCCC board contains the logic for the Primary directory, Full/Empty array, LRU array and LRU encode.

7.2.6 DUPLICATE CACHE DIRECTORY

When more than one active device (Processor or IOM) is attached to a memory subsystem, one of these active devices may change main memory data which resides concurrently in cache of another active unit. The purpose of the Duplicate Cache Directory is to keep track of and notify the Primary Cache Directory if a word duplicated in cache data storage has been modified in main storage. This keeps the processor from supplying itself with incorrect data from its cache, and will allow the processor to get the correct modified word from main storage.

In order to assure integrity of cache data, the directory look-up and clear capability must operate independently from the remainder of the processor, and must operate while the processor is in the idle state.

The Duplicate Cache Directory is organized the same as the Primary Cache Directory: four levels divided into 512 columns. The duplicate directory contains duplicate partial main memory addressed (bits A00-A12) for all data stored in cache data storage. There is also a Full/Empty array for the Duplicate Directory. The F/E bit defines if there is a data block stored in cache data storage at the corresponding location.

See Figure 7-3 for a block diagram of the duplicate directory. One path to update the duplicate directory is through the write stack. The address is loaded into the write stack of the duplicate directory at the same time it is loaded into the primary directory. The level signal from cache control directs address bits A00-A12 to the same level of the column in duplicate directory.

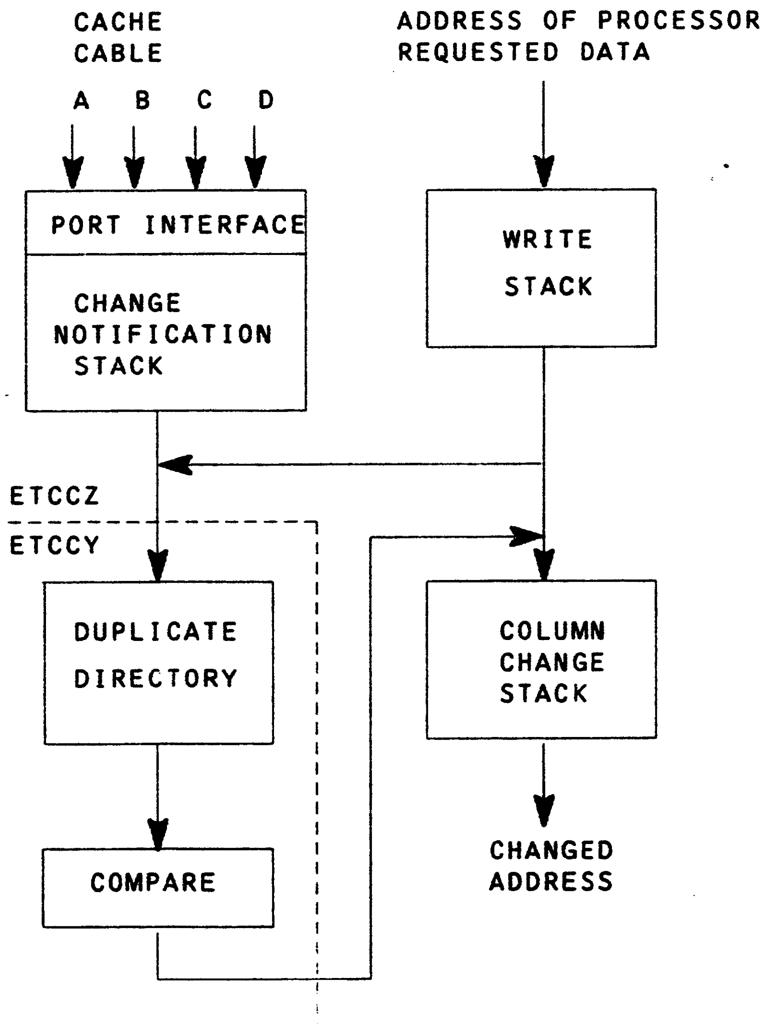


FIGURE 7-3. DUPLICATE DIRECTORY BLOCK DIAGRAM

The second path to update and check the duplicate directory is used each time an active module writes through a system controller to a main memory location. The system controller sends the absolute address to the cache duplicate directory through the cache cable interface. Each interface is buffered four deep in the change notification stack. The addresses in the change notification stack are processed asynchronously with processing and occurrences in other parts of cache.

Absolute address bits A13-A21 address a particular column and bits A00-A12 are compared to the four column level contents, if there is a match the Full/Empty bit is set empty and the address and level number is sent to the four deep column change stack.

The write notification interface from the SCU, change notification stack, write stack and column change stack are located on the ETCCZ board. The storage for the duplicate directory and comparator are located on the ETCCY board.

7.2.7 CACHE DATA STORE

Cache Data Store is made up of four levels, each level is divided into 512 columns that are individually addressable. Each column location contains space for a block of four-36 bit words and parity. Data is handled as two 72 bit words plus parity. The cache column address (A13-A22) along with level identification determine which column and level will supply the addressed word pair.

Cache memory is divided between the six ETCCD boards. Each CD board handles six bits of both the odd and even word pair.

Board Type	Logic Bay	Slot	Bits Even Word	Bits Odd Word
ETCCD	AB	N	00-05	36-41
ETCCD	AB	P	06-11	42-47
ETCCD	AB	Q	12-17	48-53
ETCCD	AB	R	18-23	54-59
ETCCD	AB	S	24-29	60-65
ETCCD	AB	T	30-35	66-71

The above chart identifies the bit slice of the even and odd words handled by each board.

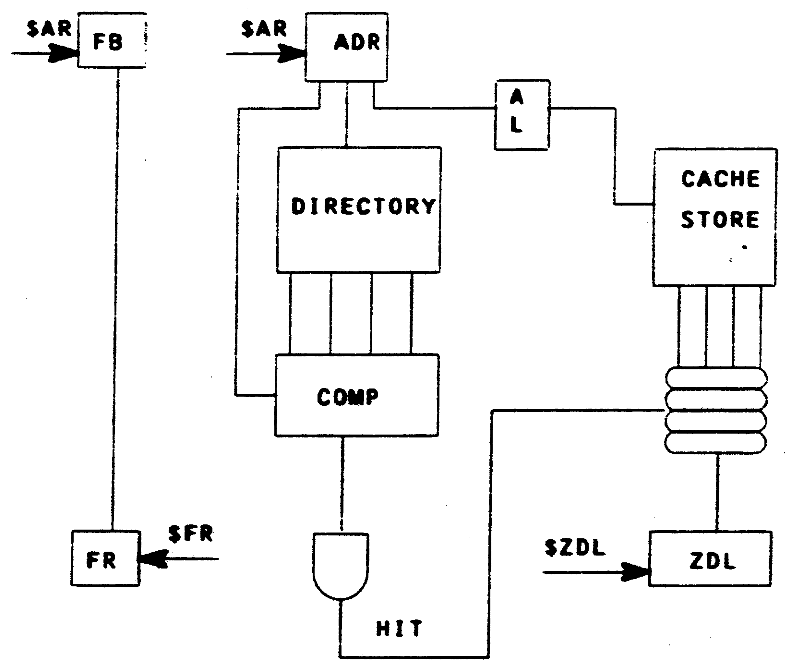
Cache can perform four types of operation:

- o Cache Read
- o Block Load
- o Store Operations to Cache
- o Store Operations not to Cache

When the DPS 8/70M Processor is first powered up, initialized or after certain fault conditions, cache is "flushed". The cache flush cycle will set all levels at each column position to empty.

7.2.8 CACHE HIT TIMING

Figure 7-4 illustrates the cache access path and timing cycle for a cache hit cycle.



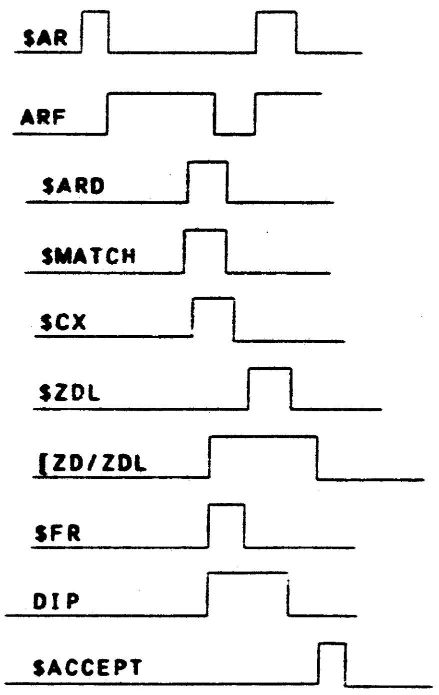


FIGURE 7-4. CACHE ACCESS PATH AND CACHE HIT TIMING

$AR (Address Ready) starts the cache. $AR occurs at the end of the address preparation cycle and allows a new address preparation cycle to begin.

$AR will put the 24 bit address into the ADR (Address Register) and sets ARF flag (Address Register Full). $ARD (Address Ready Delayed) is $AR through a delay line. The delay line allows the directory to be read out so that we can determine what is going to happen next. We determine by $ARD time if it is a HIT (match) or a MISS. If it is a hit, we send $CX back to the ETCCX board allowing cache to go ahead with the next cycle. At that time we will strobe data from cache and store it into the ZD latch with $ZDL. When there is a directory hit, the level bits determine which level of cache store is gated to ZDL.

Starting with $AR, we load the FB (Function Buffer). The function buffer keeps track of outstanding cache and port cycles. The port and cache control can have up to three cycles outstanding at any one time. The function buffer keeps track of which port or cache has outstanding cycles, cycle type (read or write) and what processor unit (OU,CU,DU,AU) requested the cycle. The function buffer and function register assures that requests are honored in order even though cache and port speeds may vary widely.

There are several things that will reset ARF. With a cache hit it is reset with HIT. If the data is not in cache and the processor goes after a port cycle, it is reset with $INT.

7.2.9 CACHE MISS TIMING

Figure 7-5 is a timing diagram for a cache miss cycle. We again start off with $AR setting ARF and again will time out for 50 to 60 nsec. for $ARD. If we do not have a HIT, we will generate $MISS and generate $PORT. $PORT will go to the ETCCM board and set F[$INT which sets $INT to request an SCU cycle.

The block load (the request by cache of four consecutive main memory words) has to have two complete memory accesses. F[$INT is reset, then set for the second cycle. Following the normal procedure, the port area will send a $PIN back following the $INT. After this first $PIN a $CX is sent back to the ETCCX board. This is to notify the DX board to get ready for another cycle. In all cases, the communication between port area and CX board is $AR with a $CX response. At $DA (Data Available) from the port, data is loaded into the CD board latch and also into cache. On a block load and MISS this word pair will contain the requested data. The processor continues through and requests the second half of the block load. On a cycle of this nature, ZD and the ZDLV switches are busy from $MISS until the second cycle is complete.

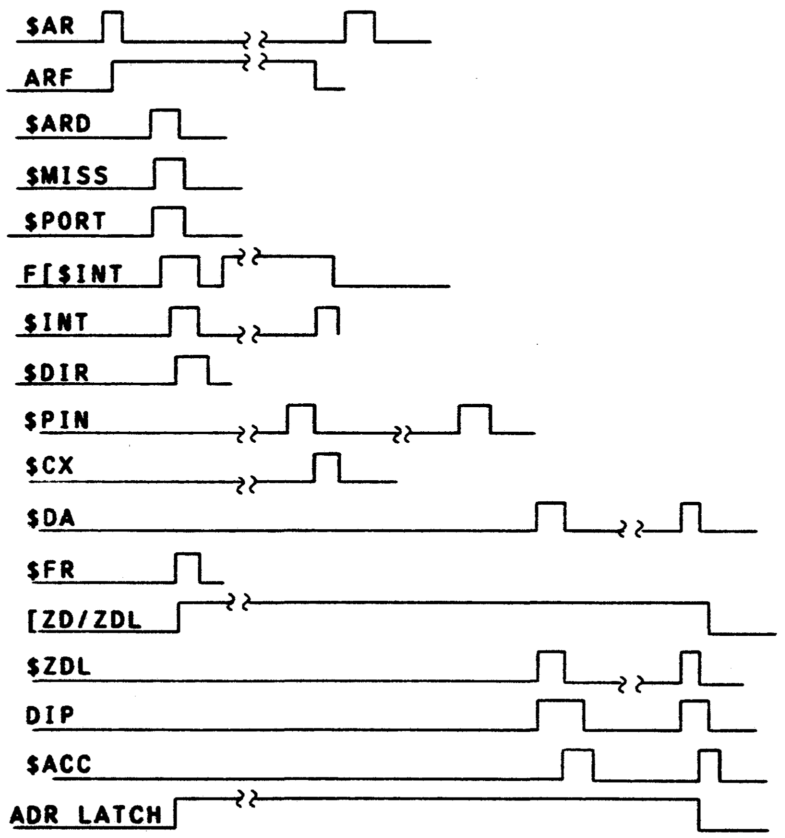


FIGURE 7-5. CACHE MISS TIMING

A direct cycle (RSW etc.) will look like a cache hit. Instead of indicating MATCH, DIRECT is ORed in to make the timing the same as the other port cycles.

There are times that we do not want to do a block load, such as read SCU registers or connect. $MISS would not be there. Cache is not checked, so the port cycle would continue with $INT to request an SCU cycle. F[$INT would not be set the second time, therefore, would not initiate a second $INT. The $DA from the SCU indicates data available and the end of the SCU cycle.

The timing chart does not show what happens on stores to the SCU. The main difference is that $PORT would be enabled by $DA. Other than that, everything would be the same.

7.2.10 ETCCC BOARD

The ETCCC board contains the cache directory, F/E array, LRU array and comparator. See Figure 7-6, Block Diagram, for ETCCC board.

Address input is the same in MULTICS and NSA through the eight section switch at the top of the diagram. The bottom pair of sections are for GCOS address. The top three pair of sections are from the append unit.

The directory at this point is organized in four levels with 512 columns per level, 2048 storage locations in the directory. Each directory entry points to a block of four words in the backing store. The 24 bit address is split. Bits 0-12 can be written into a level and bits 13-21 address the directory columns. Bits 22,25 are not used. Bits 0-12 are compared with what is read out of each level of the addressed column to determine which cache data storage location will be read out.

The address for the backing store is latched, then sent to the ETCCD boards. The address includes bits 13-22 to select the column and word pair of the block. In addition, level identification of the block is sent along.

The F/E array is addressed by bits 13-21 of the 24 bit address. The array is arranged in four levels of 512 columns each. Each of the 2048 bits correspond to a four word block in cache data store and defines whether the four word block is full or empty.

The four section switch to the BCU bus is for display of what is in the directory, F/E array and LRU array. To the left we have stop on address. The real address is used to stop on address compare. Input is via the DMP. There is another comparator that compares on segment and computed address.

Hardware clear is also located here on the ETCCC board.

The LRU array is arranged in 512 columns. There are six bits in the array that compare the four levels to determine which was least recently used. See section 7.2.5.

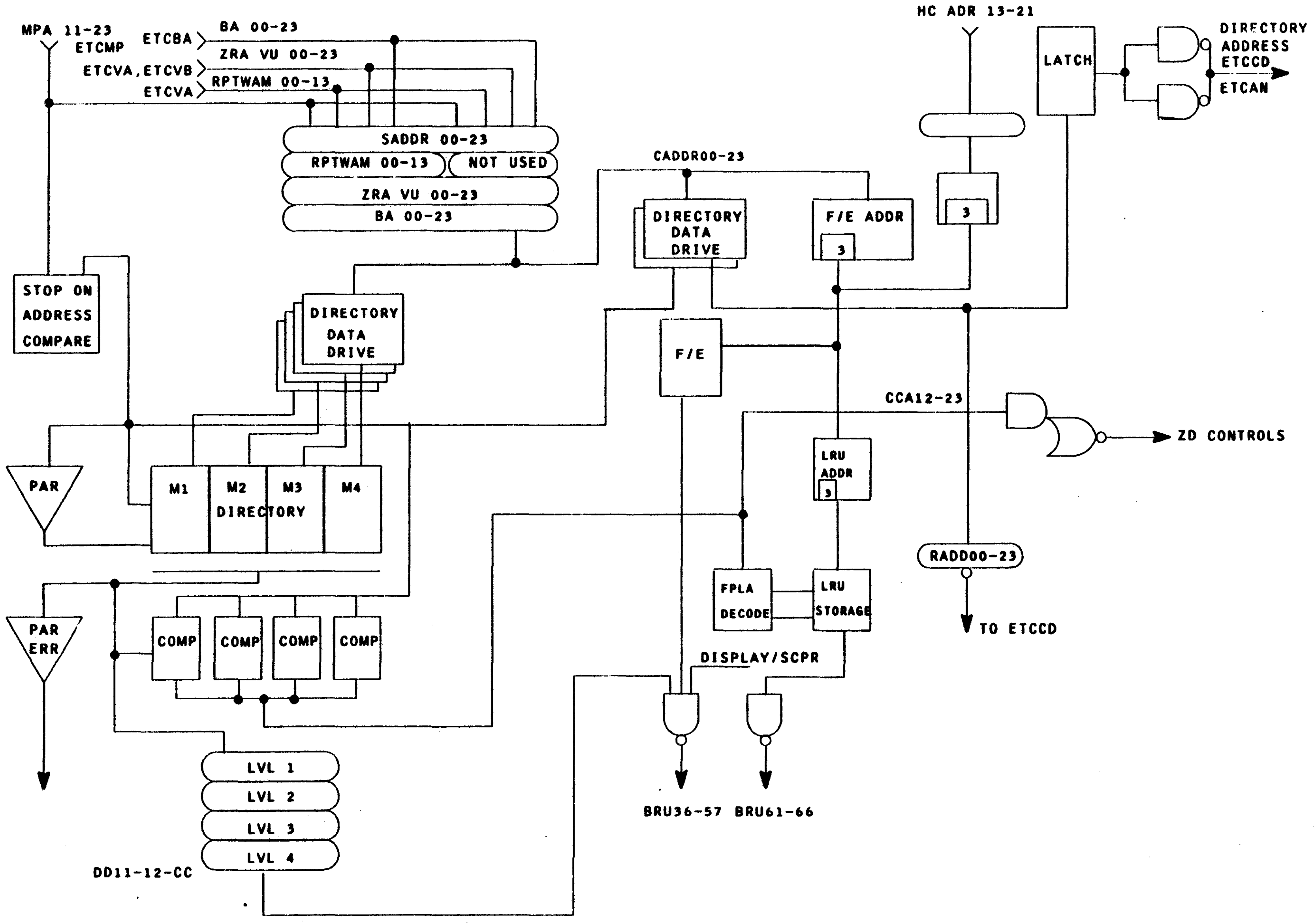


FIGURE 7-6. ETCCC BLOCK DIAGRAM

7.2.11 ETCCD BOARD

The ETCCD boards collect and disperse data words to the units of the CPU (OU,CU, DU and VU) and provide data interface to and from the SCUs. There are six CD boards and each board handles a six bit slice of the upper and lower word pair. In addition, the ETCCD board contains the appropriate slices of Cache Data Storage, the instruction buffer, ZD latch and store buffer. See Figure 7-7.

Cache Data Storage is organized in four levels of 512 columns each. A four word data block is stored at each level and column intersection. Cache Data Storage goes to the seven section switch near the center of the ETCCD Logic Block Diagram. The directory match controls the switch at this point and will select which level of cache data store we will read through to the ZD latch. Data for the OU,DU,AU all come out through the latch and are distributed.

If data is needed quickly for a CU load, transfer instruction or fast PIA the latch, is bypassed and the data goes directly through the four section switch ZI to the CU.

Data going to and from the four ports go over 80 bidirectional data lines to each SCU. This is indicated along the top edge of the ETCCD Logic Block Diagram. The L68 had port boards. The DPS 8/70M has the data port of the port cable come directly into each CD board. Data from an SCU is latched at the port latch. The three-state output of the latch provides a switch that selects the port data delivered to the seven section switch and passed via the eight section ZD bus switch to the store buffer.

The eight section switch supplying the store buffer is the collecting point for data from the CU,DU,OU,AU and ports. The store buffer provides data for transfer out to ports and cache data store.

Just below the store buffer are the bidirectional lines to the maintenance panel (DMP). Everything for the DMP goes through here: data gathered for the DMP and DMP supplied switch settings.

The instruction buffer is loaded from the ZD latch and is eight words deep.

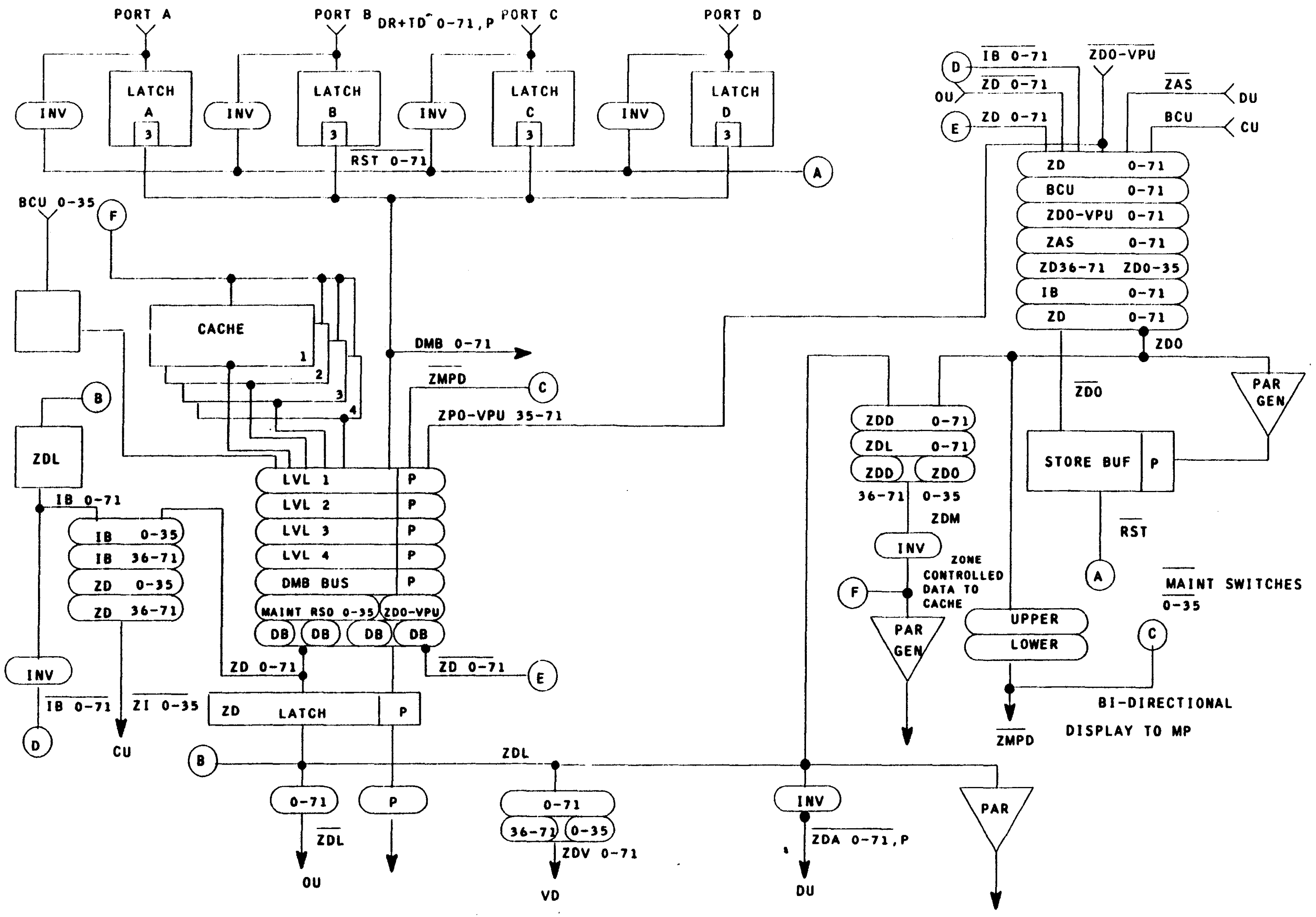


FIGURE 7-7. ETCCD BLOCK DIAGRAM

7.2.12 ETCCZ BOARD

The ETCCZ board is part of the hardware cache clear subsystem. In the L66 2K cache in GCOS mode a cache flush was done every time a new process was started or a fault or connect was taken. In MULTICS the method of clearing cache was more of a software method, but in both cases the result was excessive clearing of cache. See Figure 7-8.

With hardware cache clear, if any other active hardware device writes to a location in main memory, the cache is checked to see if it is an active location in cache. If it is an active location, the location is emptied so that the next time the CPU requires that location, the CPU will not find it in its cache and will have to go to main memory and fetch it.

Hardware cache clear is composed of address buffering and control for each port on the ETCCZ board and the duplicate cache directory on the ETCCZ board.

The SCU has one board, the SCU MH board, that transmits Write Notification to all units with a cache. Write Notification transmits the addresses for all stores that were made by all active modules not on that port of the SCU. If the CPU is connected to port 6 of the SCU it will get notification of all writes to ports 0-5,7 on a separate cable. The cable contains 10 address lines and one parity line and a couple of control lines. Bits 23,23 are not used because they refer to a word within a block. Bits 0,1 are not used because the maximum address of the SCU is four megawords. The address is sent over in two 10 bit groups, for every store cycle.

The address is registered and placed in a buffer. The signals can come in to any of the four ports asynchronously so we have to be able to buffer them. The buffer for each port is four words deep. The buffer in the path from the primary directory to update the duplicate directory is also four words deep. If any of these buffers overflow because they can't be handled fast enough, we lose track of what has happened and flush the cache.

The duplicate directory has to be loaded with the address every time we perform a block load or load the primary directory. This is the reason for the buffer and path from the primary directory.

If we find a comparison between one of these addresses and the contents of the duplicate directory, we go back and set the F/E bit to empty at the corresponding column and level in the duplicate directory. We then have to send that column address and level number back to the primary directory and set the corresponding F/E bit to empty. Because the primary directory may be tied up doing something, the primary directory is also buffered four deep.

HONEYWELL CONFIDENTIAL & PROPRIETARY

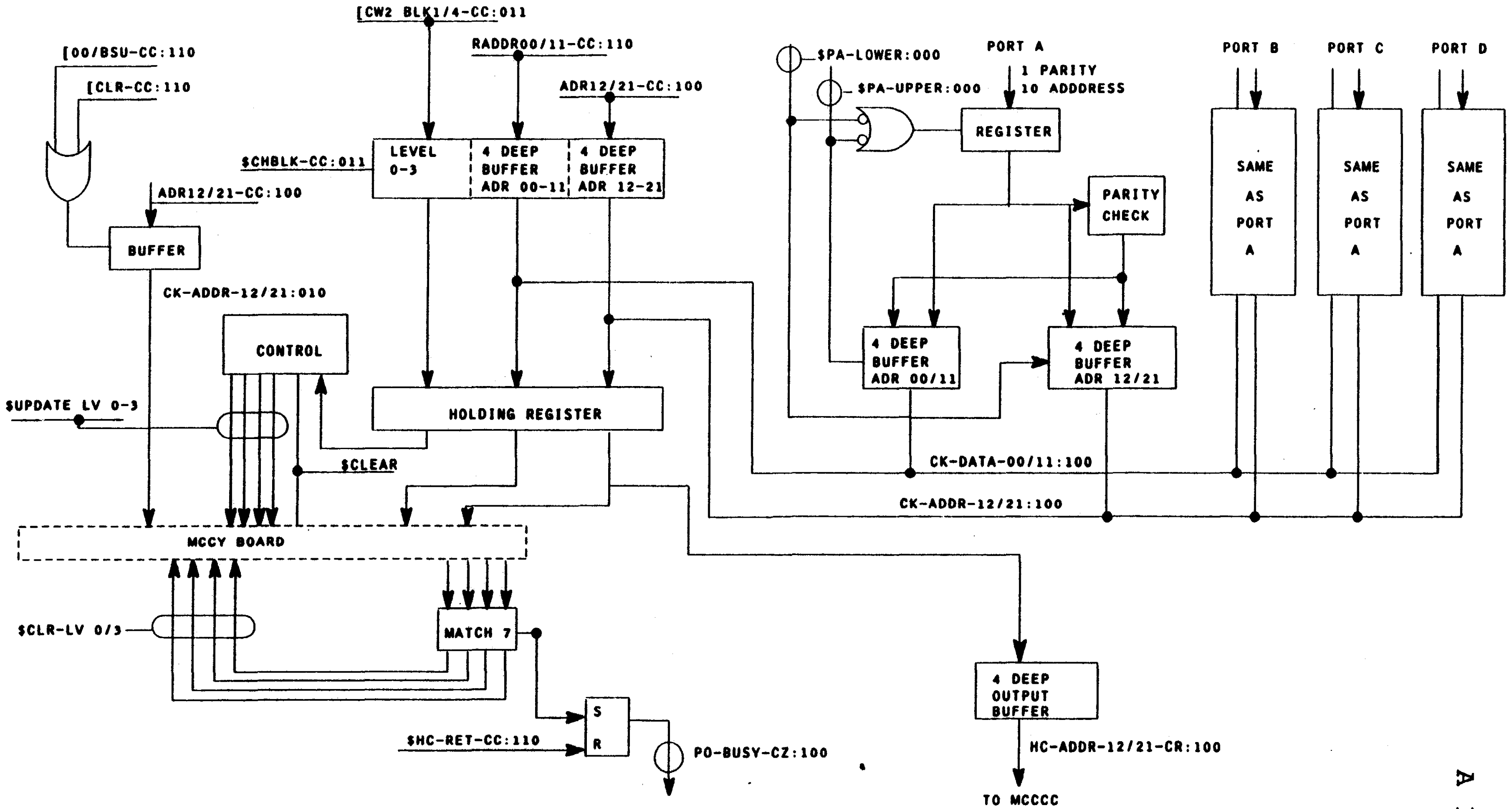


FIGURE 7-8. ETCCZ BLOCK DIAGRAM

7.2.13 ETCCY BOARD

The ETCCY board contains the duplicate directory, a Full/Empty array and the comparator. See Figure 7-9.

The duplicate directory is organized the same as the primary directory. It contains four levels with 512 columns each. Therefore, there are 2048 storage locations, each entry points to a block of four words in backing store (main memory).

The 24 bit address is split as follows: Address Bits 0-12 can be written into the duplicate directory or compared to the contents of the four levels of a column. Address bits 13-21 are used to address the directory column.

The F/E array is also organized as four levels with 512 columns for a total of 2048 storage locations. The bit at each storage location defines if the four word block in cache data storage is valid or not valid. The array is addressed with address bits 13-21.

In the lower left corner is a box labeled display select. This switch to the BCU bus is used to display the contents of the directory and the F/E array.

7.3 INSTRUCTION BUFFER MANAGEMENT

Figure 7-10 illustrates Instruction Buffer Management. There are eight instruction buffers. They are divided into two sections: the first four locations are the A buffer and the second four locations are the B buffer. The processor is set up to operate out of either the A half or the B half. If the processor were operating out of the first or A half, it would use location 0,1,2,3 and then go back to 0. The buffer is located on the CD boards.

There are two registers that reflect the address being executed out of the instruction buffer. Register RG1 reflects the address associated with the latest instruction fetch of the A half and RG2 reflects the latest address of the B half. The RG1 and RG2 registers are located on the BG board.

The processor starts off in the A buffer and when it goes out and fetches an instruction pair, if bit 16 is off, the pair would be put in location 0 and 1 of the buffer. If bit 16 is on, the pair would be put in location 2 and 3 of the buffer. The processor will continue to execute out of the A buffer until it gets the transfer go. When the processor gets the transfer go, it will switch to the B buffer. So basic operation of the instruction

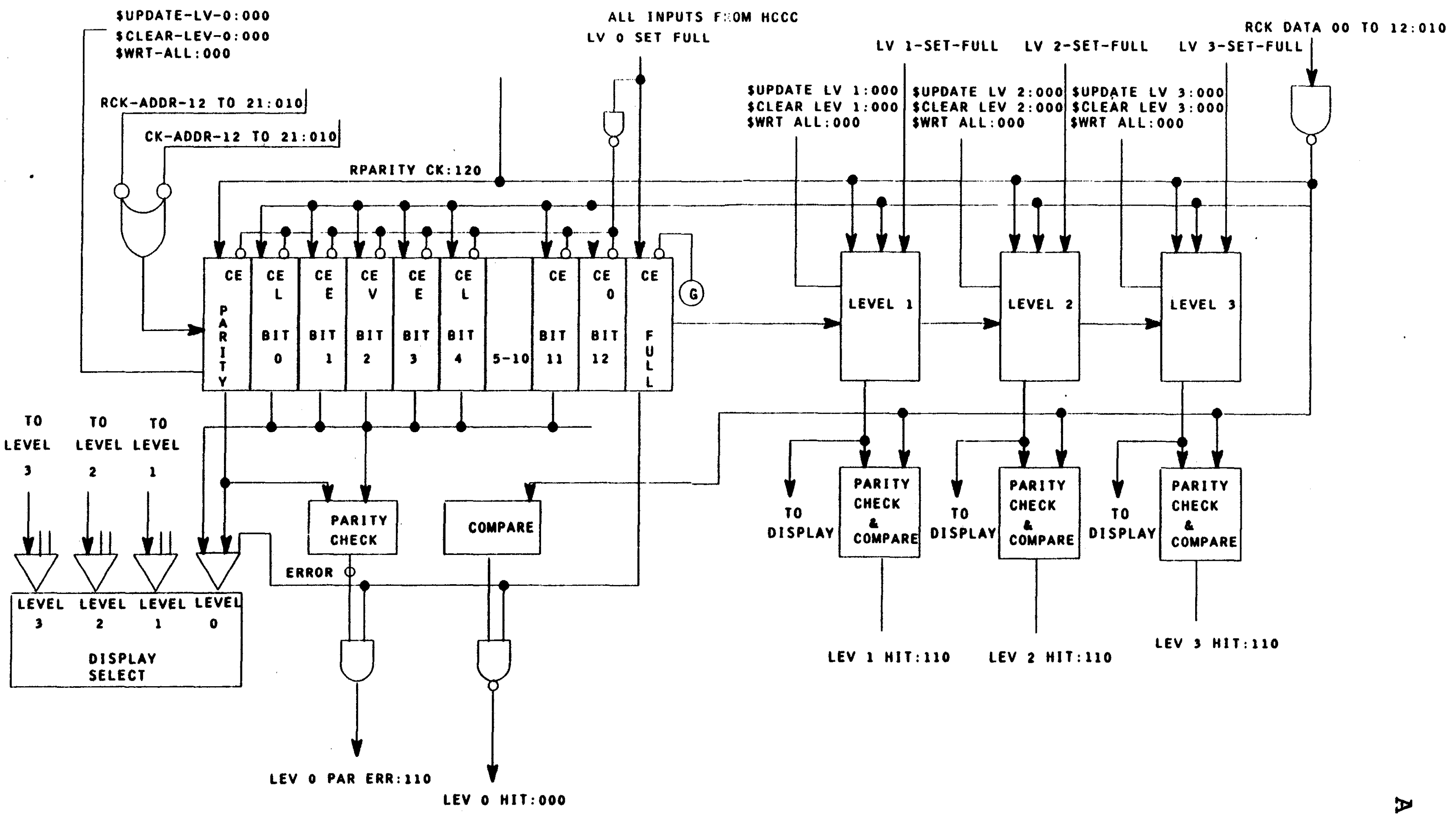


FIGURE 7-9. ETCCY BLOCK DIAGRAM

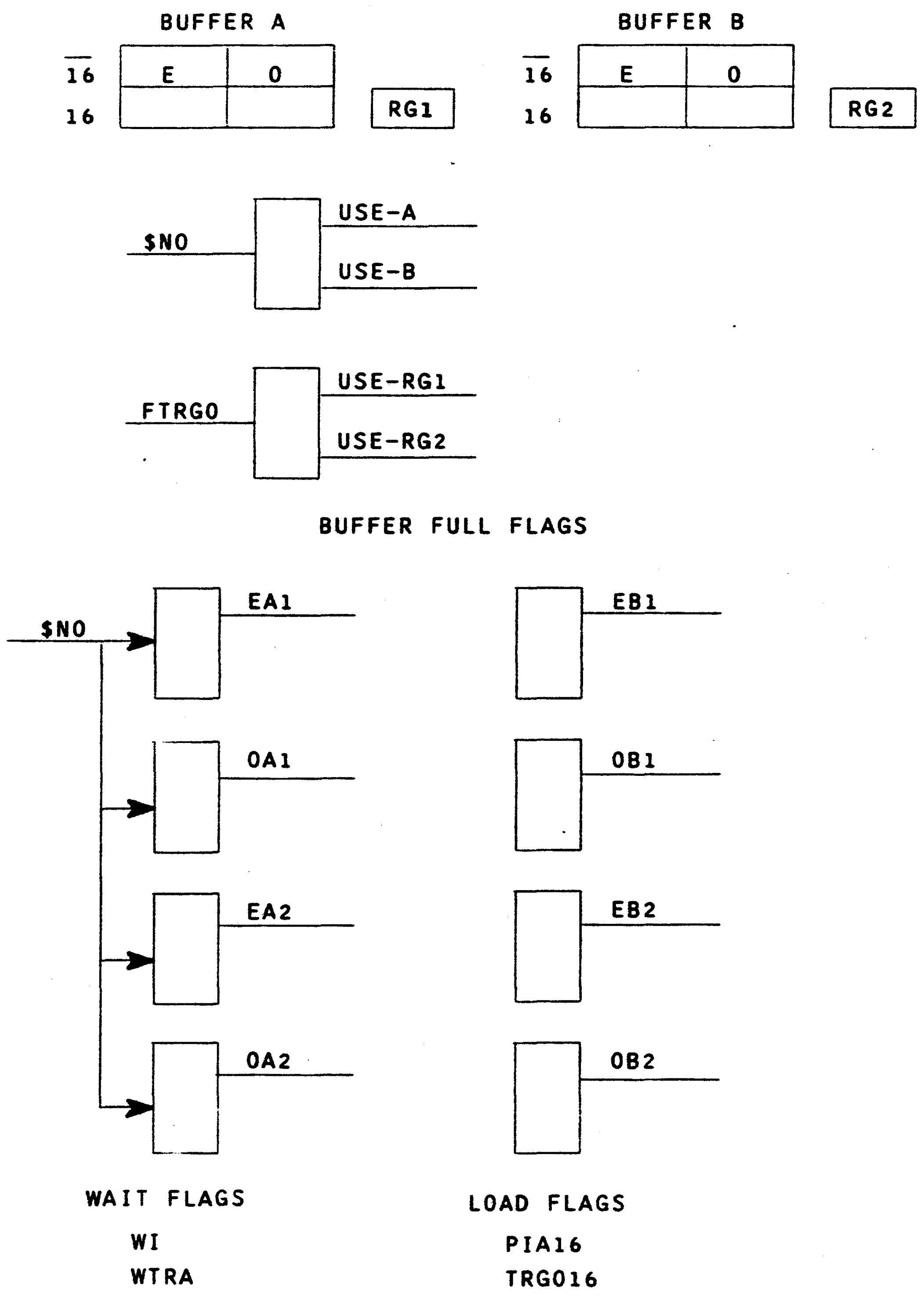


FIGURE 7-10. INSTRUCTION BUFFER MANAGEMENT

HONEYWELL CONFIDENTIAL & PROPRIETARY

buffer is that the processor will operate from either half and change halves on a transfer. If the processor executed several conditional transfers that were no go and the processor was operating out of the B half of the buffer, each conditional transfer would load into the A half of the buffer, but the instructions would not be used if the transfers were no go.

FTRGO is the signal that toggles use between the A and the B buffers. For each entry in the buffer there is a buffer full flag. EA1 and EA2 for even locations and OA1 and OA2 for odd locations of the A buffer. There is a corresponding set of flags EB1, EB2, OB1, OB2 for the B buffer. There are two wait flags WI (wait for instruction) and WTRA (wait for transfer).

When execute doubles are being executed, the execute double pair is pulled and put in the opposite buffer. The instructions are executed out of the opposite buffer, then the instruction stream execution switches back to the original buffer.

7.4 ETCBG BOARD

Figure 7-11 is a simplified block diagram of the BG board instruction address preparation paths.

This board contains the two RG registers. On a transfer either one or the other RG registers would have been loaded. To fetch the next instruction pair, the active RG register would have a constant added in the IAA adder to form the instruction address at IAA 0-17. The output of the IAA adder is also used to update the active RG register. The processor's instruction stream will switch between the RG 1 and RG 2 registers and the A & B instruction buffers as transfer conditions are satisfied and the transfer conditions taken.

The RH register is loaded directly from RG 1 or RG 2 registers. $ABI inhibits loading the RH register. The register is enabled at $NEW-OP, changing IC values, transfer go or not in hold waiting for instruction completion. Output of the RH register goes to the ITC trackers.

The ITC tracker provides the storage necessary for the instruction overlap depth used by the control unit. The tracker is made up of four 18 bit registers that are loaded from the RH register. One of the registers is strobed at the beginning of each instruction and will contain the 18 bit IC value until the last event has taken place for that instruction.

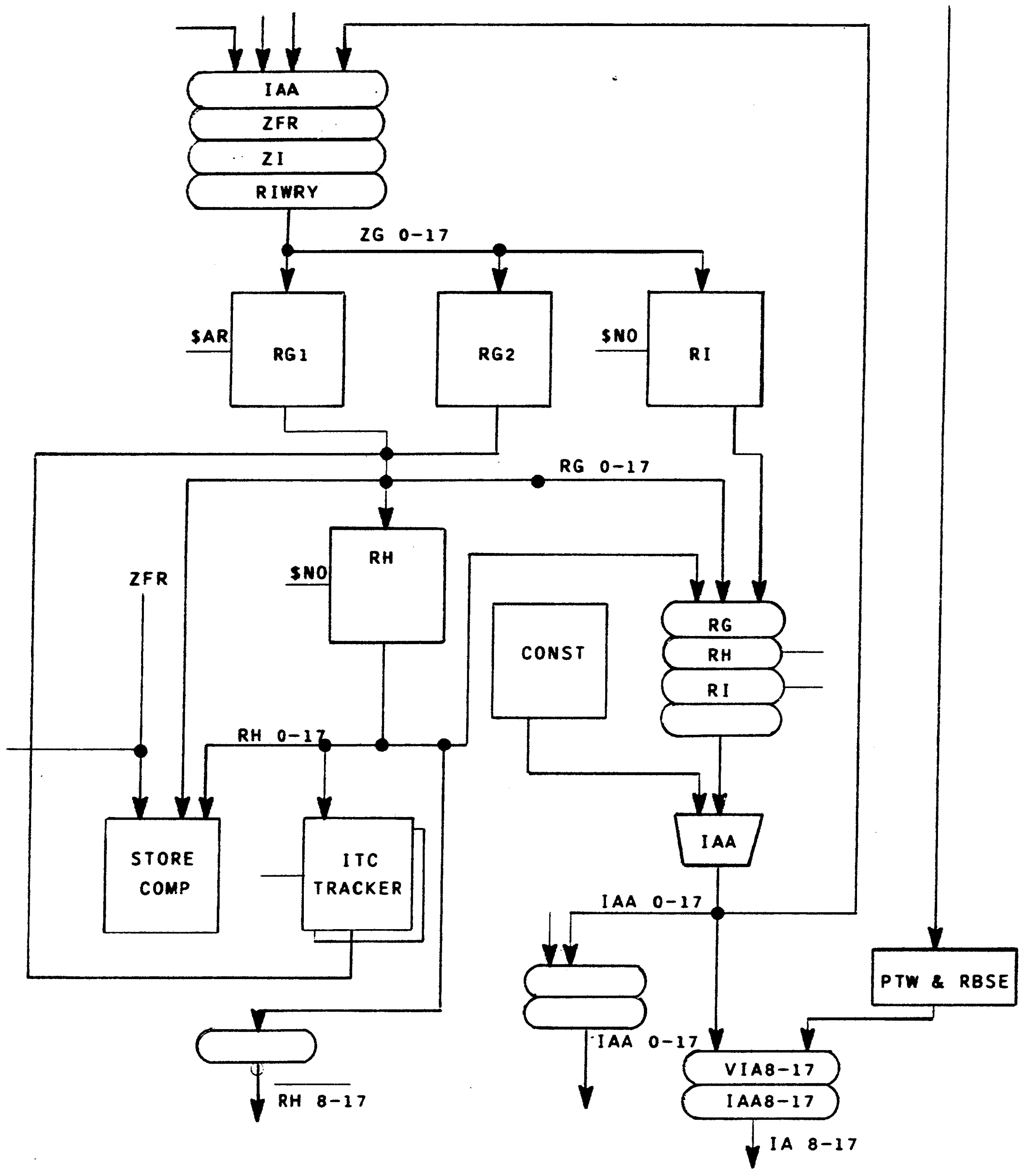


FIGURE 7-11. ETCBG BLOCK DIAGRAM

HONEYWELL CONFIDENTIAL & PROPRIETARY

The tracker register that receives the RH value is determined by a two stage gray code counter, FA and FB. A second gray code counter, FC and FD, is used to select the proper register for output.

The input counter is not advanced for CU operations. CU operations do not allow overlap to take place so there is no need to engage the tracker. The IC value is recorded but not advanced for the next $NEW-OP.

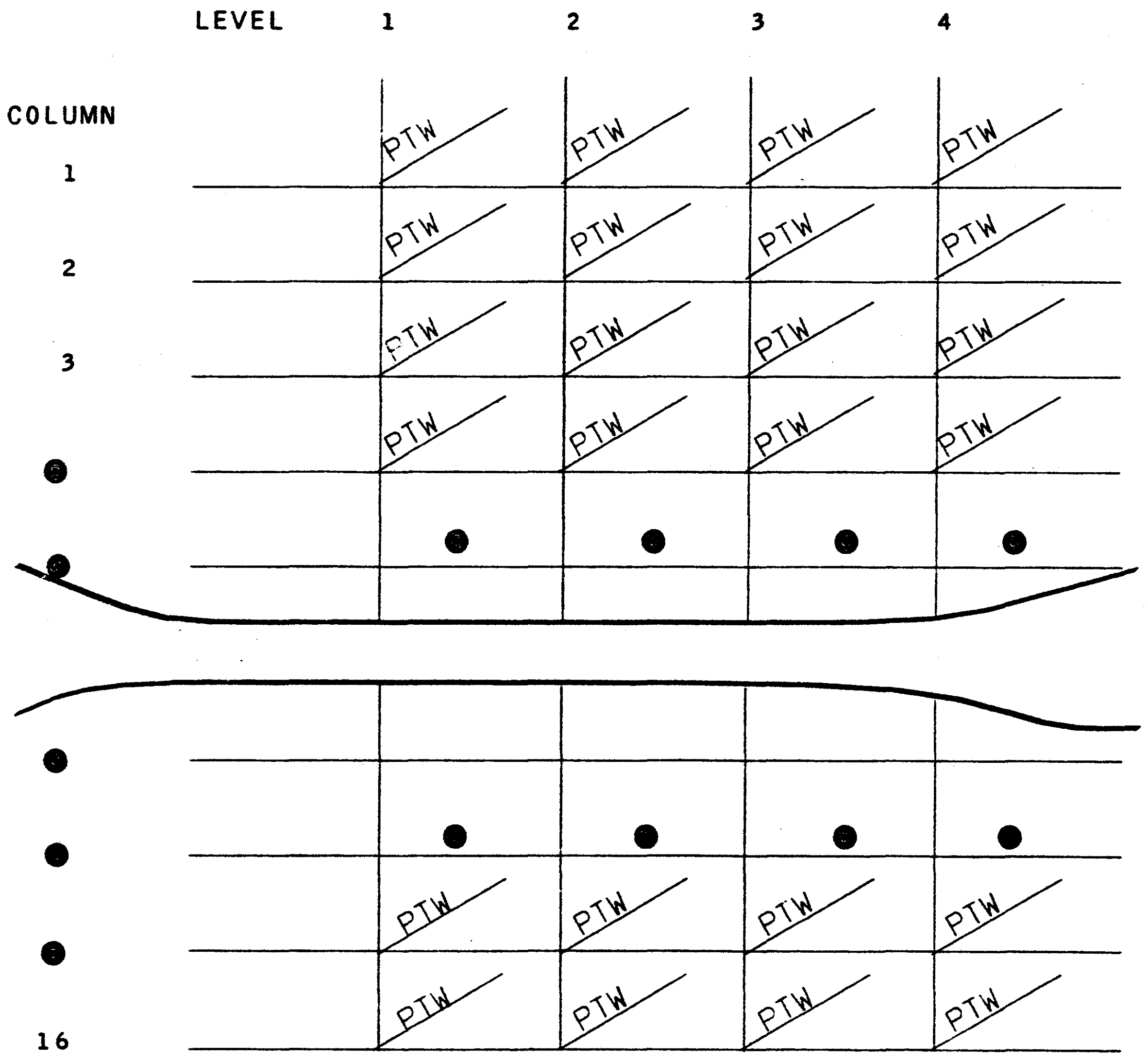
The store comparator checks to see if an instruction will store into or alter the contents of the next instruction in storage, or that .instructions already pulled from storage into the instruction buffer are not altered in main storage. The store address is taken from the ZFR switch while the instruction address comes from the RG 1, RG 2 registers or the RH register. A comparison is continually being made. The comparison is made over bits 00-15 of the address. Bits 16 and 17 not used means the comparison will make for any one of the 4 words in the addressed blocks. If stores are made into this four word window, the next instruction to be executed from this four word area of memory will be repulled from main storage. This is accomplished by forcing a transfer modified by IC value when a store compare is encountered. This results in the opposite instruction buffer being filled from storage and instructions being supplied from this opposite buffer.

The PTW and RBSE register reflects the lower 10 bits of the instruction counter. The register is updated by the Append Unit each time a branch instruction is taken or is updated after other actions that change the IC value. Use of this register makes it unnecessary to involve the paging mechanism every time an instruction is fetched. This allows the paging mechanism to be used entirely for operands. When the count goes past 1024, the signal [PG-OVFC is generated, the page boundary is being crossed by the IC. This forces a fault and a transfer to a new page.

7.5 ASSOCIATIVE MEMORY

The Append Unit contains a Page Table Word Associative Memory and a Segment Descriptor Associative Memory. Both of these memories are organized and operated in a manner similar to the way the cache memory is organized and operates. They are high speed buffer memories that contain PTWs and SDWs for the process currently under execution.

The PTWAM is composed of a Directory, Full/Empty array, Least Recently Used (LRU) array, data storage and control. The directory identifies which words are in both main memory and the associative memory. If the word is in the associative memory it is used from the associative memory data storage section. As the following diagram illustrates, each position of the PTWAM is organized into four levels, each level is further divided into 16 columns. Each column is individually addressable and a PTW is stored at the intersectiion of each level and column. The Full/Empty array identifies which level and column intersection contain a valid PTW. If a Full/Empty bit is set, the intersection is full and contains a valid entry; if 0, the intersection is not to be used (not valid).



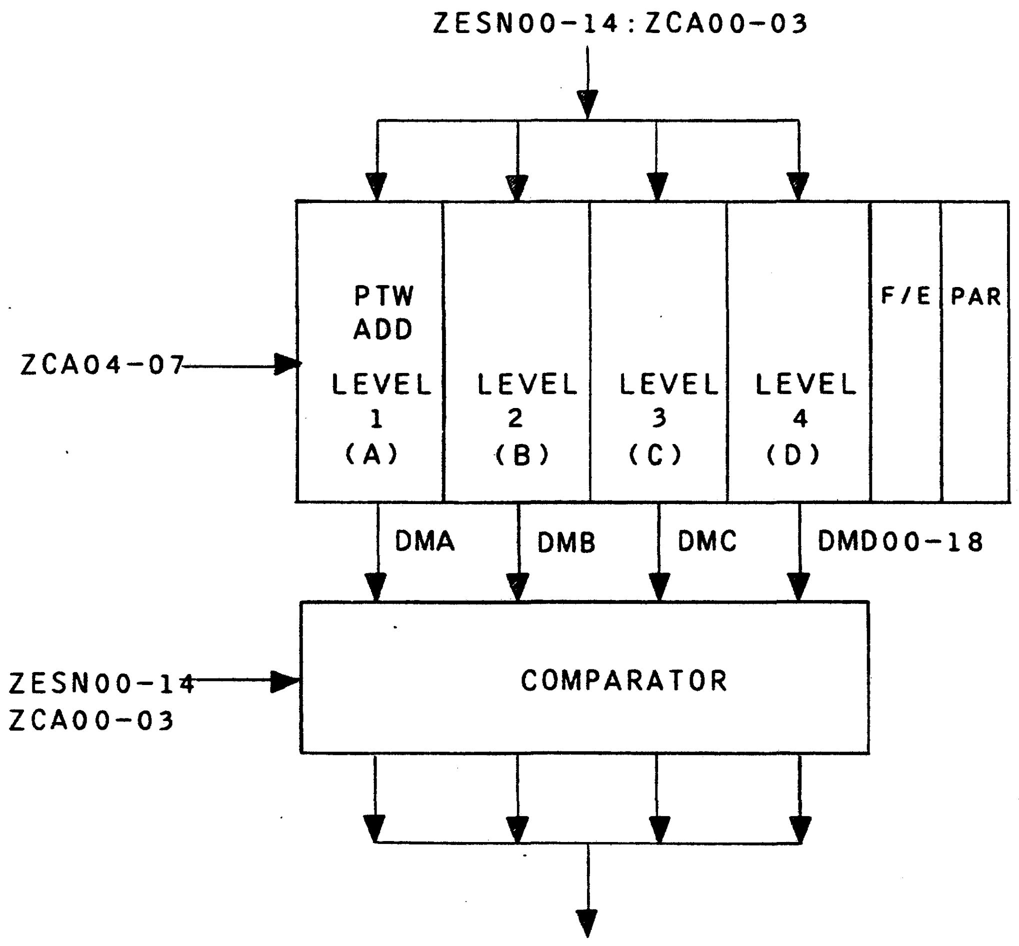
The Least Recently Used (LRU) array defines, for each level of each column, which level was LRU. Therefore, it defines which level would be replaced by a PTW if a new PTW was requested for a particular column. Data storage is composed of several storage arrays that contain the PTW.ADDR, PTW.M, PTW.POINTER, PTW.PAGENO, PTW.F and PTW.USE fields.

The SDWAM is similar to the PTWAM. Each section of the SDWAM is organized into four levels, each level is further divided into 16 columns. Each column is individually addressable. The hardware is similarly divided; it is composed of a Directory, Full/Empty array, LRU array, Data Storage and Control.

7.6 PAGE TABLE WORD ASSOCIATIVE MEMORY (PTWAM)

7.6.1 PTWAM DIRECTORY

The PTWAM is used to store the most recently used 64 PTWs. These words are from various Object Segment Page Tables, as described under section 5.2 Paged DSBR.U=0. The absolute address of the PTW is formed from SDW.ADDR and TPR.CA bits 00-07. This 24 bit absolute address is used to access the PTWAM Directory as follows to see if the PTW is in the PTWAM.



Whenever a PTW is stored in the associative memory, the first 19 bits of the absolute memory address of the PTW is also stored in a level of the PTWAM directory at a column determined by the last four bits. Whenever a check is made to see if a particular PTW is in the associative memory the last four bits, ZCA 04-07, address the selected column and the four levels at this column intersection are read out. The first 19 bits of the PTWs absolute memory address stored at each level is compared to the first 19 bits of the current PTW absolute memory address in the four 19 bit comparators.

If there is a comparison (match/hit) at any one of the four levels, the requested PTW is in the associative memory. The location of the PTW is identified by the four column address bits and level number. If there is no comparison (no match/no hit) the PTW is not in the associative memory and will be fetched from main memory.

7.6.2 F/E BIT

There is a Full/Empty bit for each level of each column in the associative memory. Bits in this array are set to indicate if the associated level and column intersection is Full or Empty.

On a compare operation, only those levels that have a Full/Empty bit set will be compared. When the directory is empty and loading starts, the levels are loaded sequentially with level 1 first. If the levels of a particular directory column are randomly empty the levels are loaded in alphabetical order.

FCNT is used to cycle through the 16 columns of the PTWAM as well as the 16 columns of the SDWAM. When the associative memories are cleared, the F/E bits are set empty. When the DPS 8/70 processor is first powered up both AMs are cleared, by setting all F/E bits to empty. The camp instruction clears the PTWAM by setting its F/E bits to empty and the cams instruction clears the SDWAM by setting its F/E bits to empty.

7.6.3 LRU ARRAY

The Least Recently Used array defines for each column the level that was least recently used. When there is an associative memory miss and a new PTW is fetched from main memory, the new PTW will be placed in the LRU level.

The procedure implemented in LRU control is to look for any empty level and, if an empty level is found, store the PTW in it. If there is no empty level the LRU level is determined and the level is overwritten with the new PTW.

The LRU array stores six bits for each column and its associated four levels. Three of the six bits are conditioned on each access to the PTWAM. Each bit represents the relative time of last usage of two of the four levels of a particular column.

7.6.3.1 LEVEL USAGE DECODE

The following chart indicates the Level Usage Decode:

	A/B	A/C	A/D	B/C	B/D	C/D
If level 1 is LRU the bits =	0	0	0	X	X	X
2	1	X	X	0	0	X
3	X	1	X	1	X	0
4	X	X	1	X	1	1
If level 1 is most RU the bits =	1	1	1	X	X	X
2	0	X	X	1	1	X
3	X	0	X	0	X	1
4	X	X	0	X	0	0

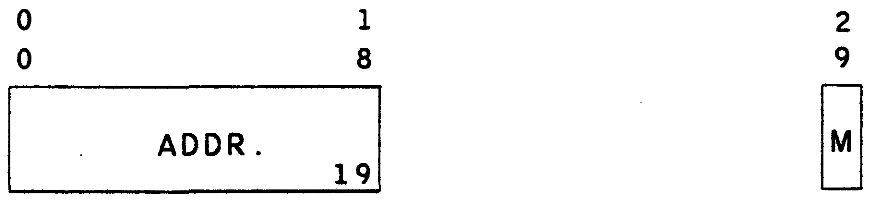
X = Don't care

Each bit specifies which of two levels is least recently used. If the bit = 0, the top level is least recently used. If the bit = 1, the bottom level is least recently used.

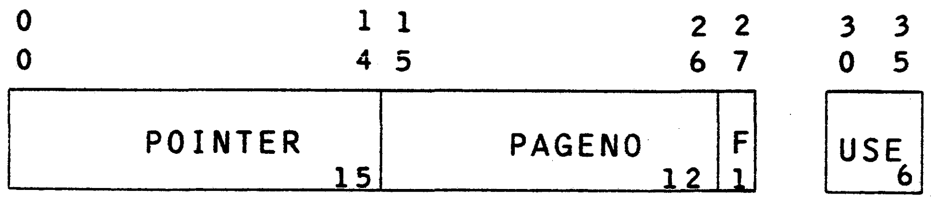
The LRU pattern is checked to see which level was least recently used. When a level is used (read or written), the most recently used pattern is written into the LRU array.

7.6.4 PTWAM DATA STORAGE

The PTWAM memory requires storage of several data fields. The following figures show the format stored by the Store Page Table Register (stpr)



and the Store Page Table Pointer (sptp) instructions



PTW.ADDR storage is the directory storage located on the PA board. Output is DMA00-18 for level A, DMB00-18 for level B, DMC00-18 for level C and DMD00-18 for level D.

PTW.M storage is located on the PA board. The output is PTW.M. The bit is set whenever the page is used for a store type operation. An extra cycle is generated to write the M bit back to main memory.

PTW.POINTER storage is located on the PA board. The output is PTWAM 00-13. This field is the effective segment number generated when this PTW was fetched from main memory

PTW.PAGENO storage is located on the PA board. The field is eight bits long (ZCA00-07) for 1024 word page size. ZCA00-03 are stored in the directory and ZCA04-07 address the directory.

PTW.F storage is located on the PA board. Output of the F/E array is F/EA, F/EB, F/EC, F/ED. When the F/E bit is set the PTW is valid; when reset, the PTW is not valid and a match cannot take place.

PTW.USE storage for this field is on the PA board. This field is now six bits long. Bits 30-35 and reflects the six bit LRU array, therefore the field reflects the use value for the four levels of a column and no longer reflects a FIFO queue among all the PTWs.

7.7 SECTOR DESCRIPTOR WORD ASSOCIATIVE MEMORY (SDWAM)

7.7.1 SDWAM DIRECTORY

The SDWAM is used to store the 64 most recently used SDWs. These words are from various Descriptor Segment Pages.

The organization of the SDWAM is similar to the PTWAM. The directory is used to see if the requested SDW is currently in the SDWAM. The F/E and LRU arrays keep track of valid SDWs and the relative usage of SDWs in columns on each level. Fields of the SDWs are stored in the SDWAM data storage.

As described under Section 5.2 paged DSBR.U=0, the DSPT.ADDR 00-13 contain the absolute address of the base of the descriptor Segment Page. The word number in the page is from TPR.SNR06-14 and appended DSPT.ADDR field to form the absolute address of the SDW in the Descriptor Segment Page.

Figure 7-12 illustrates the SDW directory, the Full/Empty array and the LRU array.

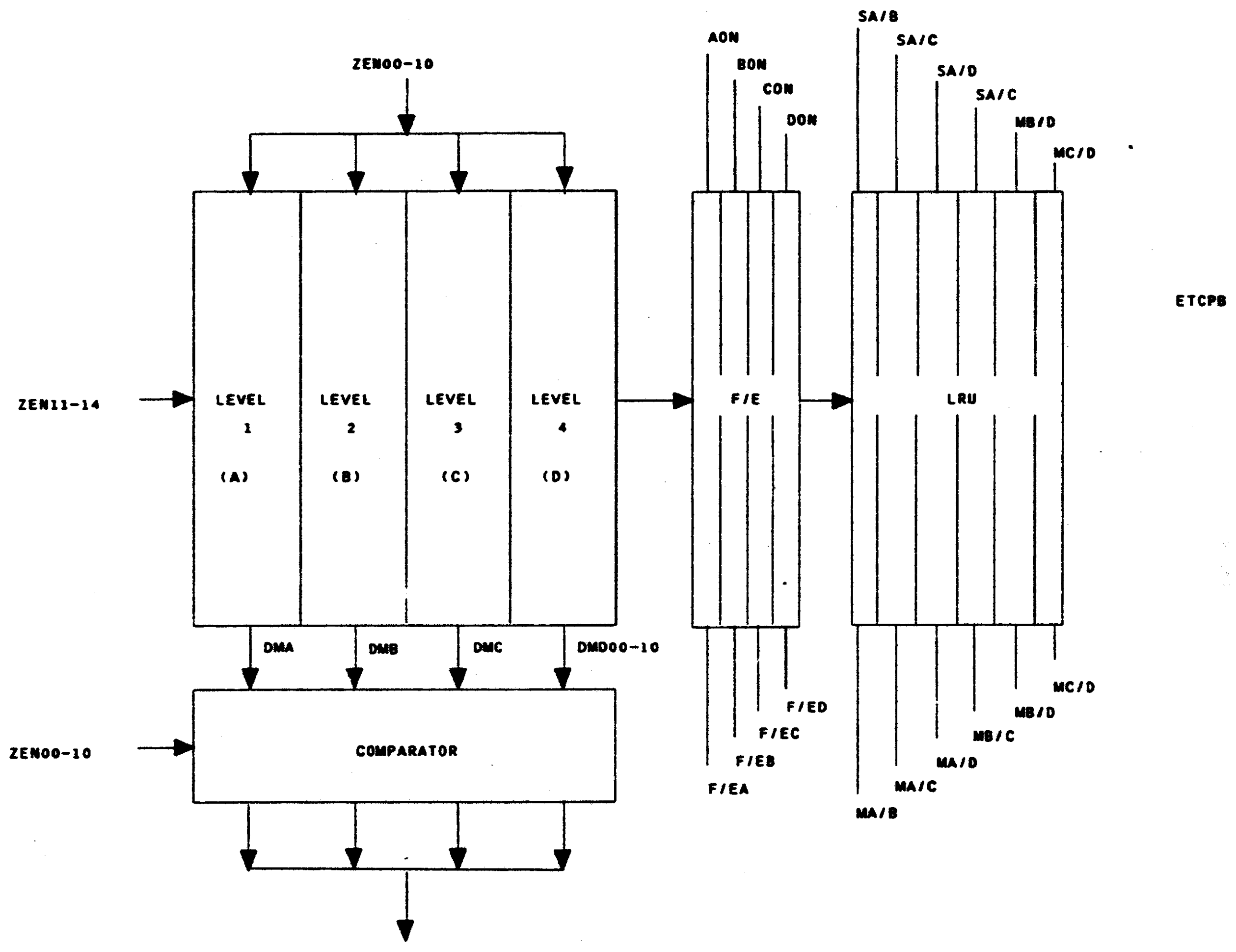


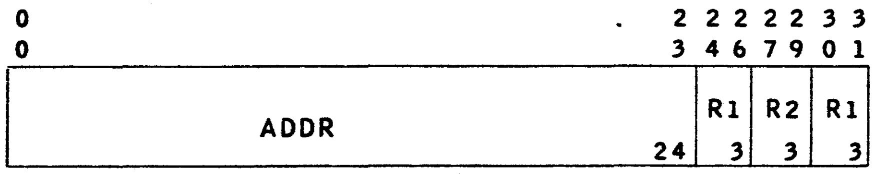
FIGURE 7-12. ETCPB BOARD, SDW DIRECTORY, F/E AND LRU ARRAY

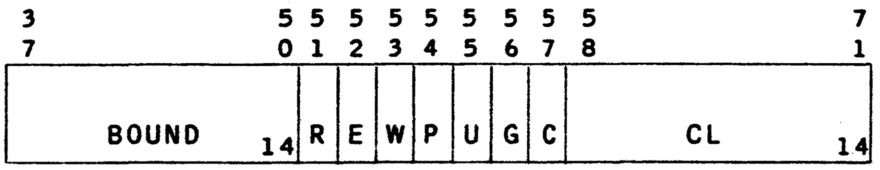
Input to the SDW directory is the effective segment number (TPR.TSR) ZESN00-14. ZESN11-14 address the selected column and ZESN00-10 is used as input to the comparator. If there is a hit, the SDW is in the associative memory.

The F/E array and LRU array function as they did in the PTWAM to keep track of the full or empty levels and the relative use of each column.

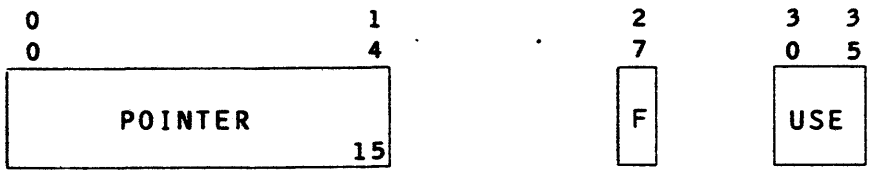
7.7.2 SDW ASSOCIATIVE MEMORY DATA STORAGE

The SDWAM memory requires the storage of several data fields. The following figure shows the format of the y-pair stored by the Store Segment Descriptor Register (ssdr)





and the word stored by Store Segment Descriptor Register (ssdp) instructions.



SDW.ADDR memory is located on the PB board; the output is SDWAM 00-23. Column selection in the associative memory is made by ZESN11-14 and level selection is based on directory match. This is the absolute address of the page table or origin of the targeted segment.

SDW.R1,R2 and R3 storage is located on the PE board; the output is RSDW.R1 00-02, R2 00-02 and R3 00-02. These fields are the upper limit of the Read/Write ring (R1), upper limit of read/execute ring bracket (R2) and upper limit of call ring bracket (R3) stored for each SDW in the AM.

SDW.BOUND storage is located on the PC board; the output is ZSDW.BND 00-13, the 14 most significant bits of the last y-blocks address within the segment that can be accessed without an out of segment bound fault.

SDW.R storage for the read permission bit located on the PE board.

SDW.E storage for the execute permission bit located on the PE board.

SDW.W storage for the write permission bit located on the PE board.

SDW.P storage for the privileged bit located on the PB board.

SDW.U storage for the unpaged flag bit located on the PB board.

SDW.G storage for the gate control bit is on the PE board.

SDW.C storage for the cache control bit is on the PE board.

SDW.CL storage for the call limiter is on the PC board. If SDW.G is off, transfers of control into the segment must be to a segment address no greater than this value.

SDWAM.POINTER storage is on the PB board. This field is the effective segment number generated when the SDW was fetched from main memory.

SDWAM.F storage is on the PB board and reflects the Full/Empty bit for the selected level of the column.

SDWAM.USE storage is on the PB board and reflects the six bits of the LRU array for the selected column.

7.7.3 SDW AM TIMING

Only the timing of the SDWAM is shown and described in this section. The PTWAM timing is the same except for the names. Figure 7-13 illustrates the SDW load sequence.

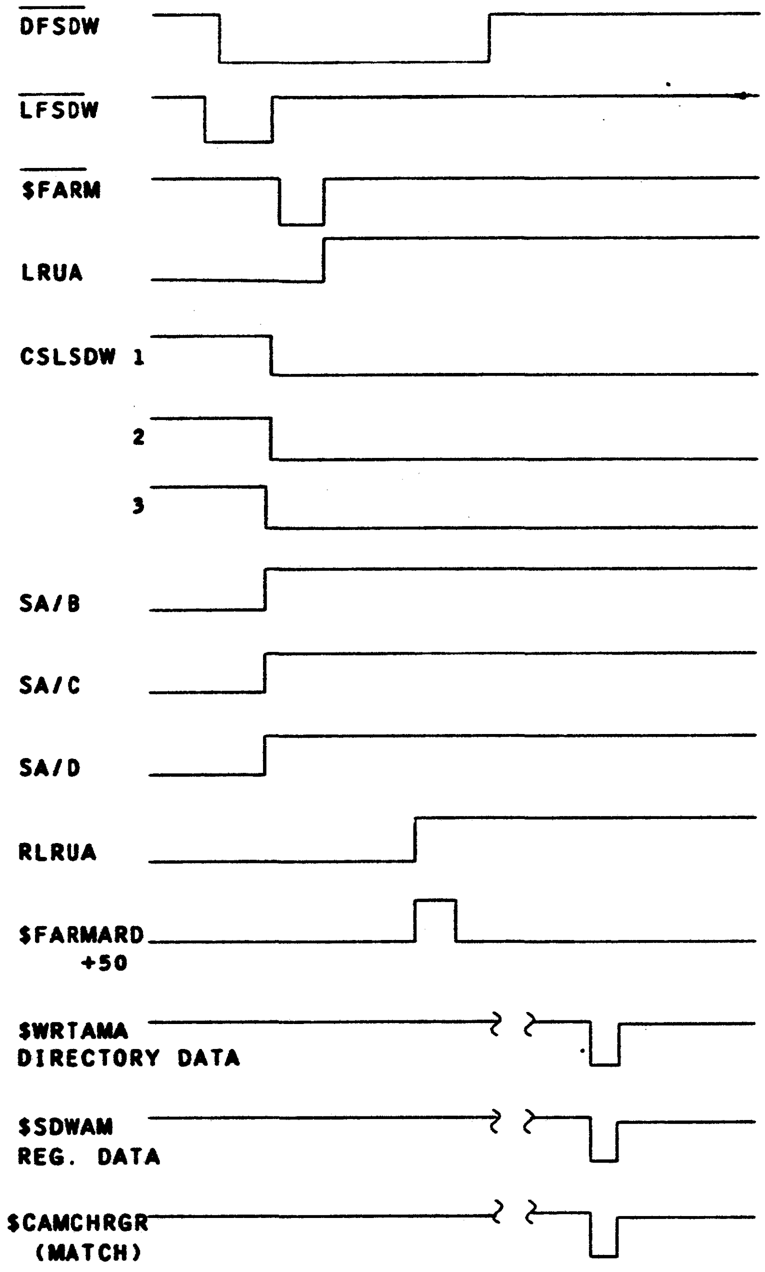


FIGURE 7-13. SDW LOAD SEQUENCE

The action of the load sequence is started by leading edge of DFSDW. DFSDW goes through a latch LFSDW that enables the FPLA (Field Programmable Logic Array). From the leading edge of DFSDW until the leading edge of $FARM the Append Unit FPLA is reading the six bit LRU code from the directory. The LRU code is being read to determine which level is LRU. At the leading edge of $FARM the FPLA shuts off and the LRU bits pertaining to whichever level should be loaded turns on the three enables for the RAMS. These outputs are what is actually stored into the LRU RAMS. This is latched up to $FARMARD+50 so at $FARMARD+50 everything is set up and the write to the directory is enabled. Then it waits for $DA to come back from memory saying that the SDW is in the port and available. At $DA time the directory data is clocked in.

In the compare sequence all four directories are enabled all the time and the check for comparison is made through the comparators. The matched level is selected in the register portion by the match signal itself at $CAMCHRGR time. In other words, if we get a match on level A, it goes over and enables the read enable for level A to read out SDW.ADDR. The LRU is stored at $CAMCHRGR.

7.7.4 ETMPA BOARD

ZRA 0-23, at the center of the ETMPA intermediate block diagram, is a 24 bit address generated on the PA board; it is used in append unit cycles, doing append unit load and stores and preparing the final operand address. See Figure 7-14.

An Appending Unit Operation Flowchart is located in Section 5.0 Address Appending.

First, DSBR is looked at to see if it is paged or unpaged, to determine what cycle to start with. If the DSBR is unpaged, a check is made for SDW match. If SDW match occurs and if the SDW is unpaged, then the APU takes the SDWAM and adds ZCA.

If the SDW is paged, a check is made for a PTW match. If match occurs, the APU takes the PTWAM and concatenates ZCA. (PTWAM 0-13 concatenates with ZCA 8-17).

If the DSBR is paged, we fetch a DSPTW that is an APU load. The processor takes DSBR 0-23 and adds in ESN 0-5 through the RS adder and enables that to ZRA. The DSPTW is loaded into the TPTW register. If the PSPTW write bit and modify bit are not on, the APU will do a modify DSPTW cycle.

The following cycle is a fetch SPTW which is the TPTW coming through again. The path is through the RS adder even though it is a concatenation. TPTW 0-13 is concatenated with ESN 6-14 to form a 23 bit address. Bit 24 is always a zero to address the first word of the SDW pair.

If there is an SDW match and the PTW is no match, the processor adds SDWAM 0-23 to ZCA 0-7.

If the processor is doing a store operation and the PTW.M bit isn't on, the processor will do a modify PTW cycle or if the access bit is not on then the processor will do a modify PTW cycle. The address preparation is the same.

A FPTW2 cycle is for decimal two or three descriptor instructions. These cycles are the same as the FPTW cycles except the Append Unit adds one to the sum in the RS adder.

7.7.5 ETMPB BOARD (ESN GENERATION)

See Figure 7-15 ETMPB Block Diagram and Figure 7-16 Effective Segment Generation Flowchart.

ZESN 0-14 are generated on the ETMPB board; they are used in the SDW AM to determine if there is an SDW match or a PTW match.

If the processor is in a page overflow condition where it has to append PIAs, the PSR register is used. If the processor is in append mode and bit 29 in the instruction word is off, PSR is also used.

If bit 29 in the instruction word is off the processor uses the SNRn register for ZESN generation. The particular SNRn register selected depends on bits 0-2 of the instruction word.

If the processor performs a load A indirect, the first cycle would use the SNRn register. Once EA is generated, the following cycles use TSR register.

HONEYWELL CONFIDENTIAL & PROPRIETARY

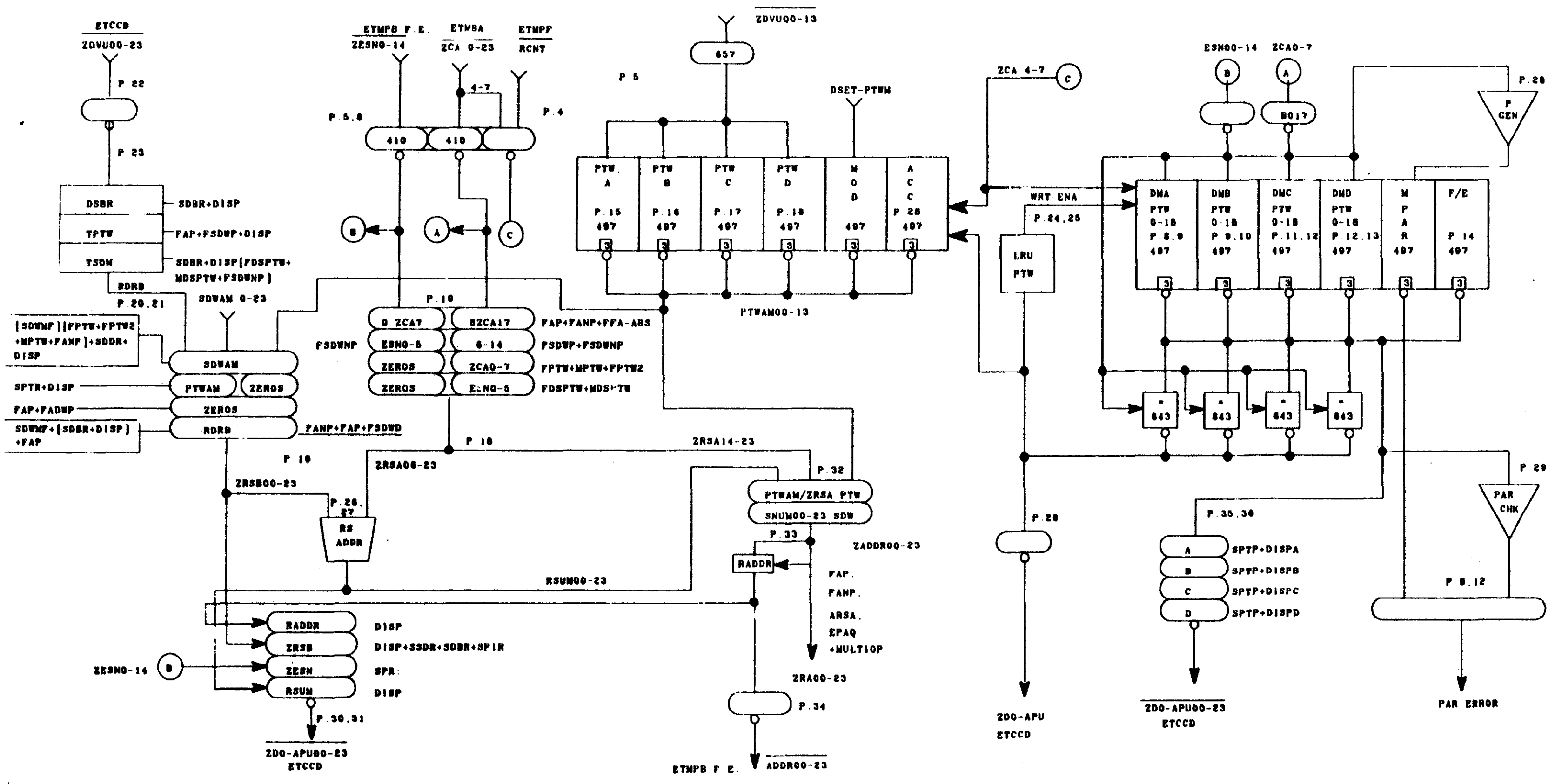


FIGURE 7-14. ETMPA BLOCK DIAGRAM

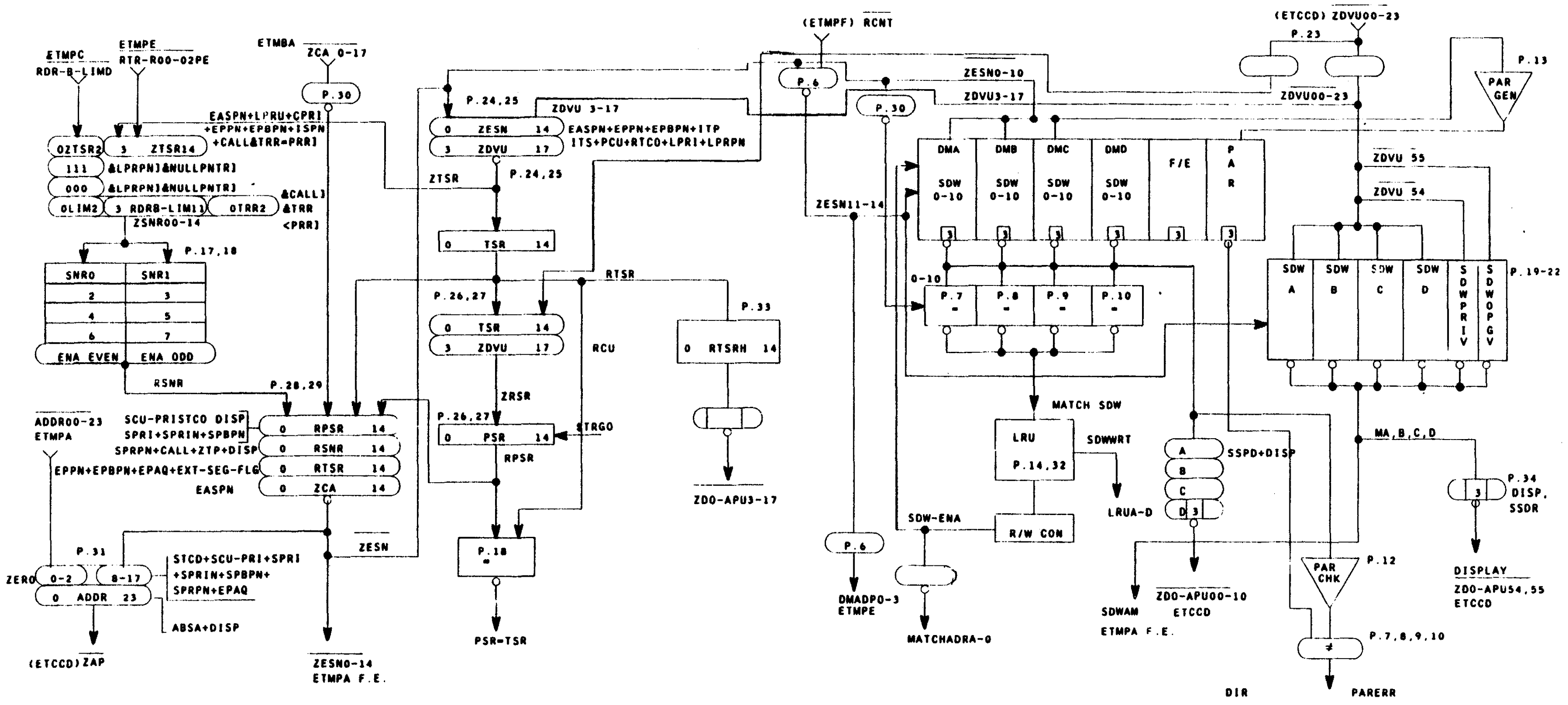


FIGURE 7-15. ETMPB BLOCK DIAGRAM

HONEYWELL CONFIDENTIAL & PROPRIETARY

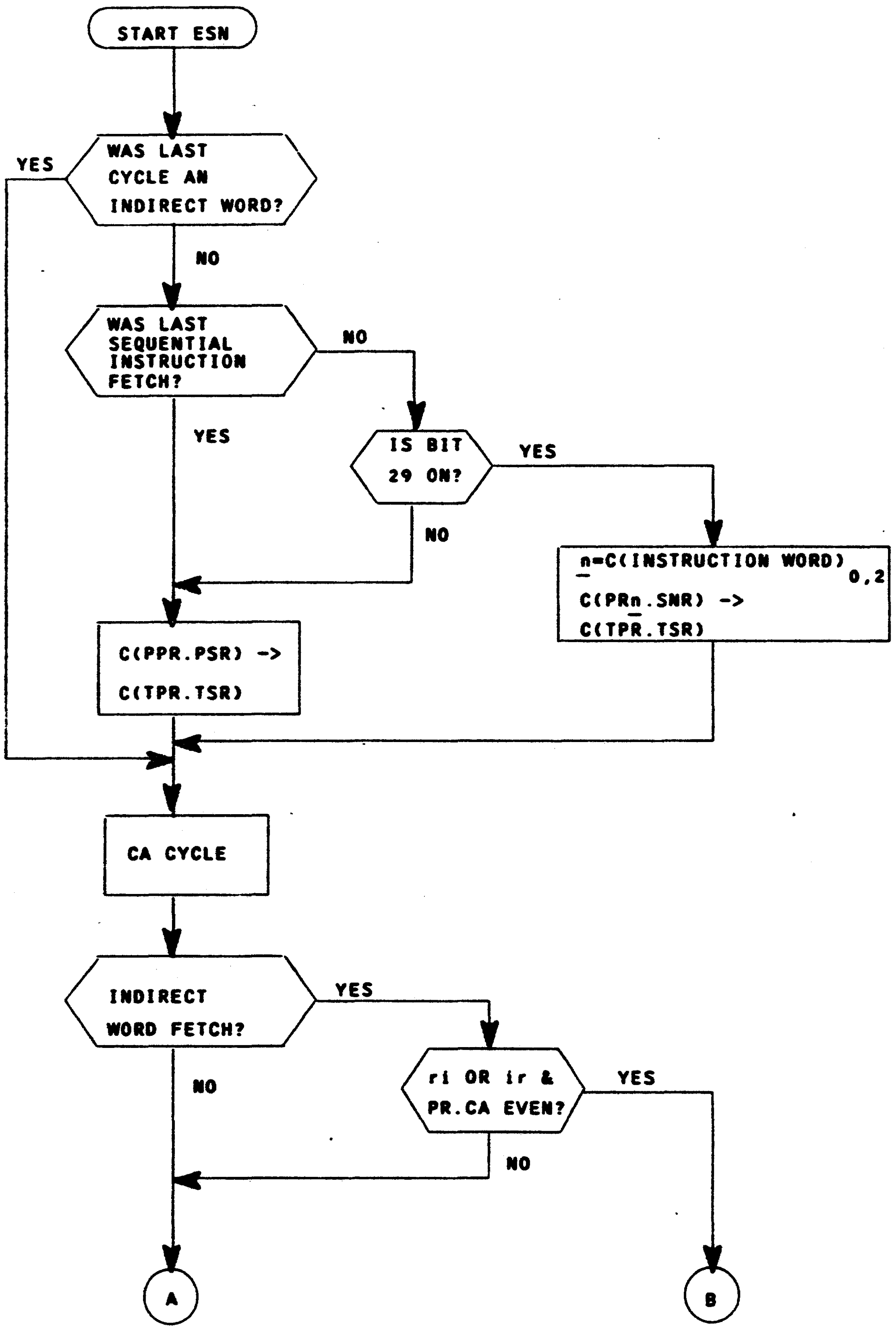


FIGURE 7-16. EFFECTIVE SEGMENT GENERATION FLOWCHART (SHEET 1 OF 2)

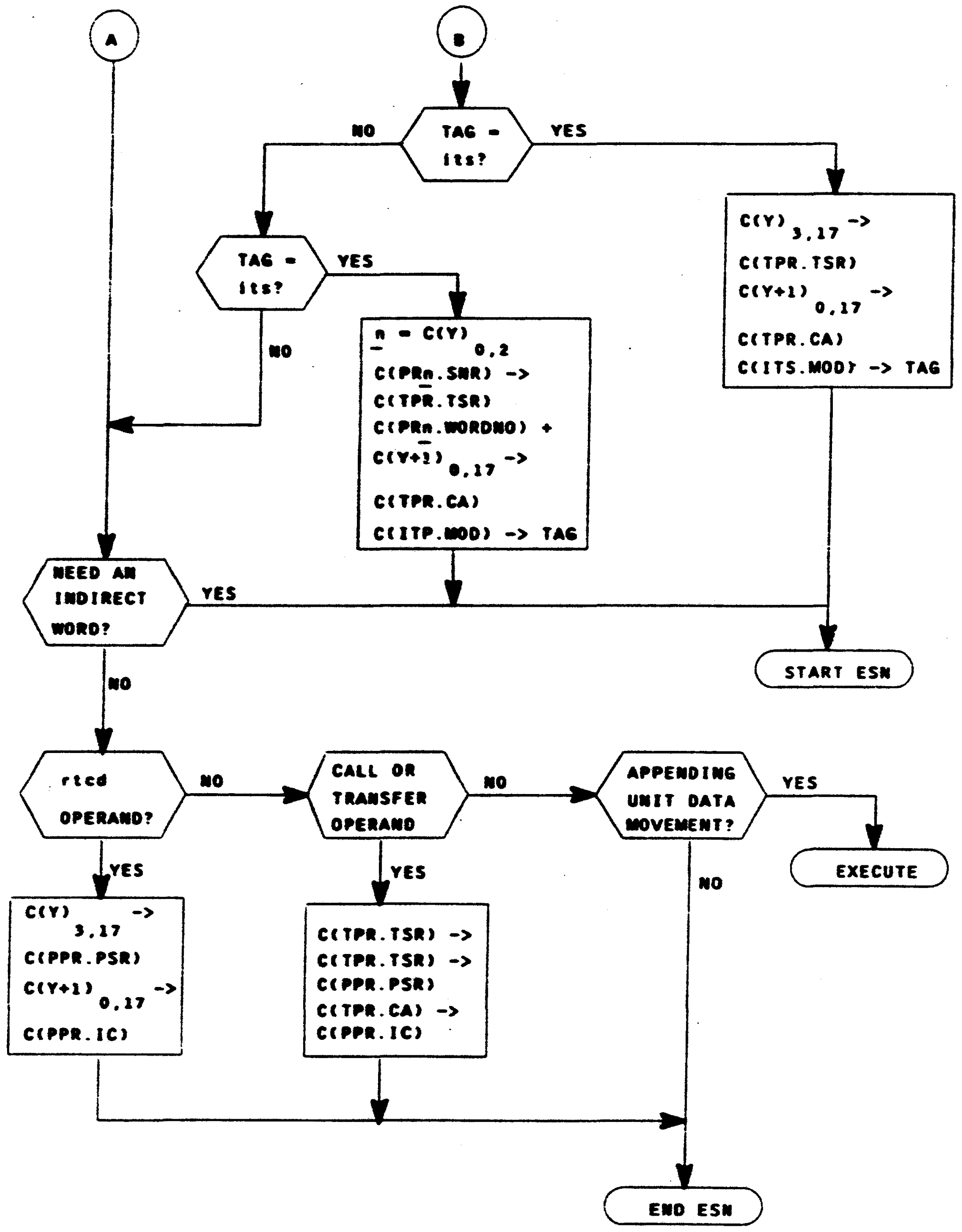


FIGURE 7-16. EFFECTIVE SEGMENT GENERATION FLOWCHART (SHEET 2 OF 2)

7.8 ETMBA BOARD (ADDRESS PREPARATION)

The ETMBA board generates an 18 bit address used for operand and instruction fetches if not in append mode. It will also generate an 18 bit computed address and a 23 bit final address for fetches used with the RCU and SCU instructions. See Figure 7-17.

The 18 bit computed address (ZCA 0-17) is from a latch on the BA board. Input to the latch is from the E2A, S1B or S2B adders. The SB1 or SB2 adders are used to add in the base address registers (BAR). Most of the time the processor is using the E2A adder for generating ZCA.

ZCA is generated by the SB1 adder only when the processor is in MULTICS BAR mode for operand fetch. The SB2 adder is used to prepare PIA during page busy, when the processor is appending on it, or if the processor is executing load/store register-type instructions, or if the RT register is in use for processor address preparation.

The PIAs that are not appended on (see BG board for PTW and RBSE register use) will use FRA. If it's a page overflow, the PIA is held off at the CX board. The E2A input to ZCA is enabled and appended on. (ZRA from the PA board is used until the final address is generated.)

The FRA is a counter used for PIAs in append mode. Its input is ZRA from the PA board. The counter is loaded on FTRGO and gets counted on TRGO delayed and $ARI delayed which puts the counter a cycle ahead.

For RCU instructions FRA is used for loading pair two and pair three whether the processor is in absolute or append mode. If the restart bit is off, FRA is also used for RCU pair four.

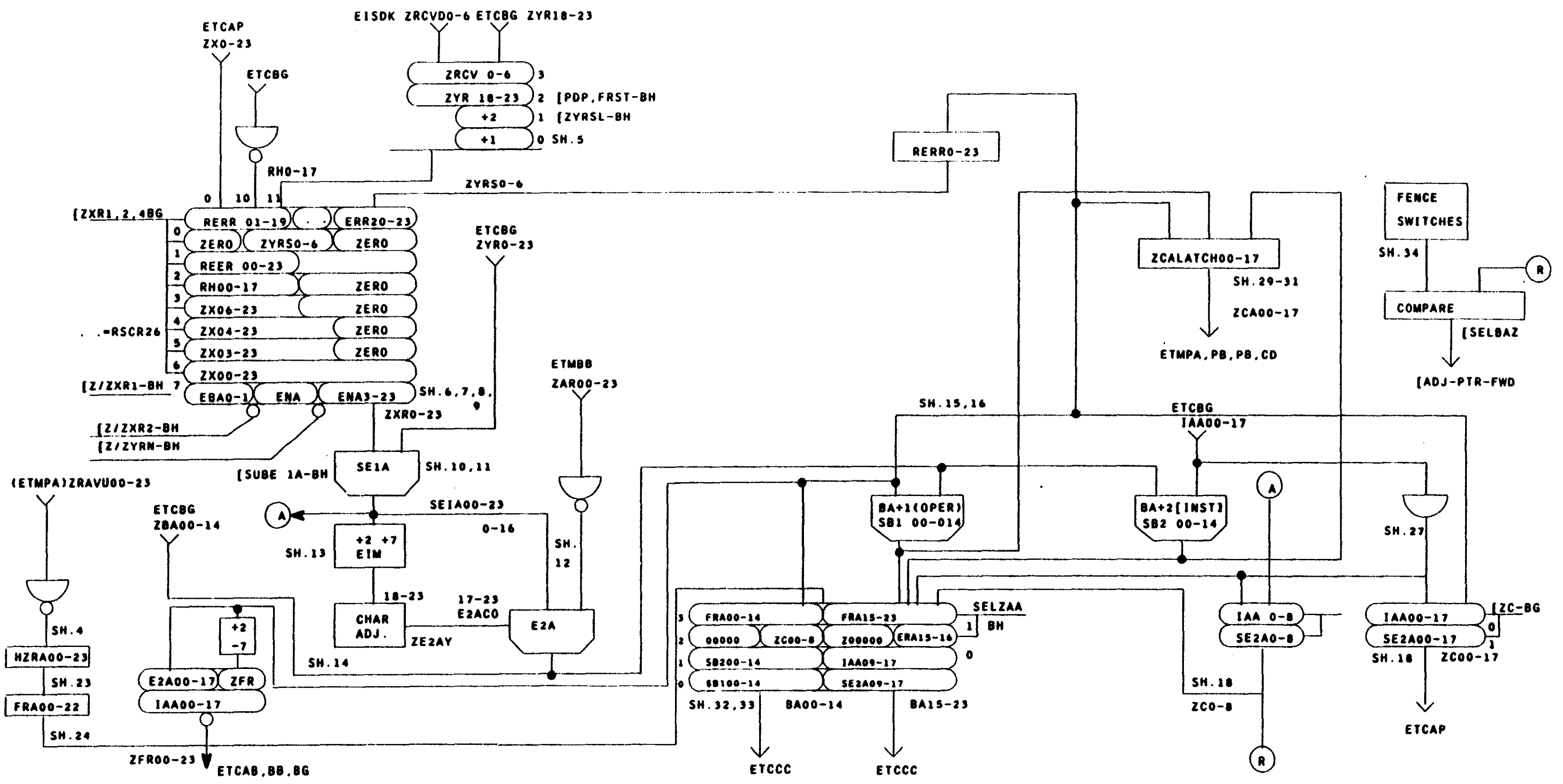
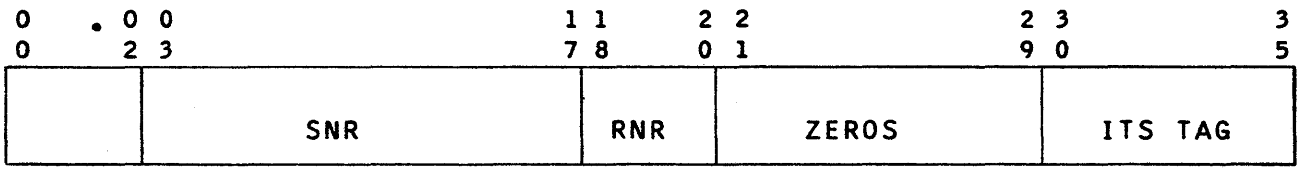


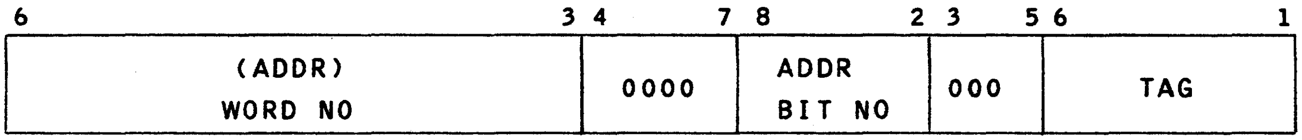
FIGURE 7-17. ETMBA BLOCK DIAGRAM

7.9 POINTER REGISTERS

There are eight pointer registers. The pointer registers are combinations of physical registers from the appending and control units.

The pointer registers hold information about data items in main memory that may be external to the segment containing the procedure being executed.





Bit	Field	Function
03-17	SNR	Segment Number Register. The segment number of the segment containing the data item described by the pointer register. The SNR registers are located on the PB board.
18-20	RNR	Ring Number Register. The final effective ring number value calculated during execution of the last instruction. Ring number registers are located on the PE board.

Bit	Field	Function
30-35	ITS TAG	Specifies indirect to segment
36-53	WORD NO (ADDR)	The offset in words from the base or origin of the segment to the data item. This is the RADRn registers on the BB board.
58-62	BIT NO (ADDR)	The number of the bit within PRn. WORDNO that is the first bit of the data item. Data items aligned on word boundaries always have the value 0. Unaligned data items may have any value in the range 1-35.
66-71	TAG	This field is not part of the pointer register, but in an ITS pointer pair, holds an address modifier for use in address preparation.

The data path to load the SNRs is on the PB board. The controls are generated on the ETMPF board as follows:

Instruction	Data Path			Control
eppn	RSNR/ZESN	ZESN/ZTSR		$TSR
	RTSR/ZESN	ZESN/ZTSR	ZTSR/ZSNR	$SNR
easpn	ZCA/ZESN	ZESN/ZTSR		
	RTSR/ZESN	ZESN/ZTSR	ZTSR/ZSNR	$SRN
lpri/lprpn	ZDVU/ZTSR	ZTSR/ZSNR		$SNR

The data path to load the RNRs is on the ETMPE board as follows:

Instruction	Data Path	Control
lpri	Max of (ZI 18-20, SDW.R1, or TRR)	$TRR
	/ZTRR	$RNR
eppn	RNRN/ZTRR	$TRR
	TRR/ZRNR	$RNR

The meaning of "Max of (ZI 18-20, SDW.R1 or TRR)/ZTRR" is that the append unit compares ZI 18-20 loaded from memory, SDW.R1, and TRR. Whichever is greatest will go into ZRNR.

The lower the ring number the more privilege the process has a right to. A process cannot increase its level of privilege, but it can decrease its level of privilege. The TRR holds the process's current ring number. A process can use an SDW to a segment to keep its privilege at the same level or decrease its privilege, or the process can load a pointer register to keep its privilege at the same level or decrease its privilege. The maximum ring number, which is the lowest level of privilege, is what the process gets.

7.9.1 ETMPH BOARD

The data path for APU load instructions that change the word portion of the pointer register is from the ETMPH board via the left free connector edge. Controls for this data are also generated on the ETMPH board. See Figure 7-18.

Bit to character bit conversion is done with PROMs on the PH board. The upper PROM decodes character to bit and the lower PROM converts bit to character. The Decimal Unit (DU) is character bit (talking about bits 18-23 of the instruction words descriptor) and the APU is in bit format. If the processor executes a lpr instruction the pointer register data is taken from bit format and converted to character/bit.

Following is a conversion chart for bit to character/bit conversion. The TBR is converted on ITS cycles and the TCR is converted if bit 29 of the instruction word is on. For example, if converting from character/bit 50, on a sprn, sprin, sprpn the bit format will be 50 and actually will point at bit 27 of the operand word.

CHAR/BIT	BIT	DECIMAL
0-10	0-10	0-8
20-30	11-21	9-17
40-50	22-32	18-26
60-70	33-43	27-35

ZPTR 18-23 is the bit portion going to the BB board. When the processor is executing an eppn with bit 29 on and it doesn't go indirect, the address latch FADR 18-23 is used. The address is not converted to bit then back to character bit format as was done in the L68. The zeros are for eppn DPs to zero out that portion of the address registers.

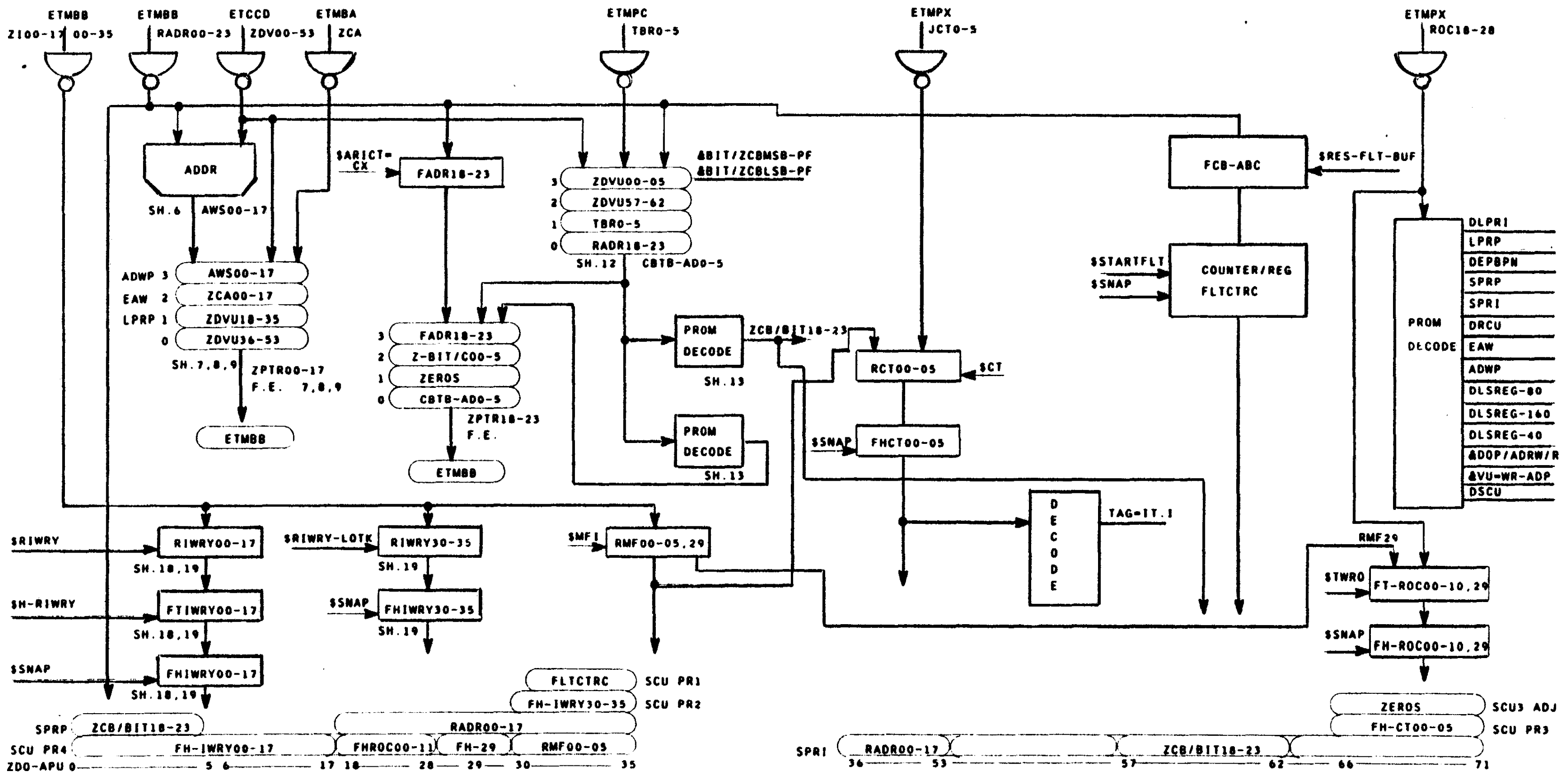


FIGURE 7-18. ETMPH BLOCK DIAGRAM

HONEYWELL CONFIDENTIAL & PROPRIETARY

ZPTR 00-17 is the 18 bit switch that affects the word portion of the address register on the BB board. The adder is used to add to the word pointer.

The bus illustrated at the bottom of the Logic Block Diagram is the ZDO bus in the APU supplied from the PH board.

7.10 ZDO - APU BUS

Table 7-1 illustrates the instructions that put data on the ZDO-APU bus, the data field put on the bus and the board where the data field originates. The stcd instruction is an exception.

The IC+2 Field, bits 36-53 is not stored out on the ZDO-APU bus, but on the BCU •bus. The ssdr instruction (Store Segment Descriptor Register) stores 16 double words of the Segment Descriptor Word associative memory because the ssdp is double precision bits 11,12 are used. The permission field consists of the SDW.R,E,W,P,U,G,C bits. The ITS TAG in the spri instruction is a 43 octal.

The SCU instruction is used just about every time the processor takes a fault or interrupt. When the processor takes a fault or interrupt, the first instruction in the vector is an scu instruction followed by a transfer. The scu instruction stores away eight machine words containing register information. The registers marked with an asterisk are restored on an rcu instruction, allowing the processor to continue from the point of interruption.

The fields stored in the eight machine words are examined in greater detail in Section 3.14.3 SCU(Store Control Unit.)

Bit 71 of the first word pair identifies if the processor has taken a fault or an interrupt. The fault address module 2 is located in bits 32-34 in the second word of the pair.

In word 0 the procedure ring register and procedure segment registers are reloaded. The fault counter is also loaded; this counter counts retries. In word 1 nothing is restored. In both word 0 and 1 the information not restored is for software use or can be used to trap on a fault.

In word 2 the temporary ring register, temporary segment register and delta field are restored. The delta field is the address increment for repeats and is reloaded on the PH board. In word 3 the TSN's field identify the pointer registers and descriptors used with the EIS multiaddress instructions.

In word 4, the ICT and Indicators at the point of interrupt are restored. The ICT can store the IC value or the IC+1 value. If the processor is executing an LDA and takes a page fault, it

TABLE 7-1. ZDO - APU BUS

```
INSTRUCTION                                   DATA FIELDS

           0              17 18           35 36             53 54                        71

                   PB & PA   PE              PE             PH               PH
spri       |000|< ZESN >|RNR| 0'S  | ITS TAG|<-- ADDR -->|000|ZCBB|<-- 0'S -->|
              3         13   20    30      35                  57   62

           PH  PB & PA          PH
sprpn      |ZCBB|< ZESN >|<-- ADDR ---->|
                5

                 PA                        PC            PD            PC
sdbr       |<-- DSBR -->|               <-BNDS->         U       CALL LIMITS
                                        37        50     55   60                          71

                     PB & PA              PE
spbpn      |000|<--- ZESN --->| ITS TAG|
                              30      35

                                          PE
sra                                     |RALM|
                                        33   35

                PB & PA     PE            ITS TAG     " BCU BUS "
stcd       |000|<-PSR->|PRR|<-- 0'S -->|        IC + 2      |

sptp       |<-- PTW DIRECTORY -->|        |LRU|

                 PA
sptr       |     PTW AM      |    |"A M"

ssdp       |        SDW DIRECTORY      "F"   |LRU|

INSTRUCTION                                          DATA FIELDS

                    PB                  PE              PC                             PC
ssdr       |       SDWAM          |R1|R2|R3|000|0|<BOUNDS>| PERMISSION |0|<LIMITS>|

LEVEL IS SELECTED BY RIWRY 11-12  DP                 | PE | PD | PE  --->:
                           12-15  SP                 "R E W P U G C"

           PE      PB & PA           PD          PH   PE/CP        PE         CP          CG   F/I
             *                *                   *
scu 1      |PRR|<-- PSR -->|< APU FLAGS>|FC|AU  FLTS |AU FLTS|CU FAULTS|FLT ADDR|0/1|
                                            CU

           PE      PB & PA          PD   PH                                PX         PC
             *               *            *                                  *
scu 2      |TRR|<-- TSR -->|< 0'S >|CPU|DELTA|<-- 0'S -->| TSN'S |  TBR  |
                                     *

                BG               BD BH            PC                 PX          PH
                             *               *                         *
scu 3      |<-- ICT -->|< INDICATORS >|<--- CA --->| CU FLAGS |  CTH  |

                PH                       PH                CD
                                           *                                     *
scu 4      |<-- RIWRY -->| OPCODE  |TAG|<------------ IBRA ----------->    |

                 CD
scu 4      |     IBRA
 *
```

NOTE: FIELDS SO IDENTIFIED ARE RESTORED ON A RCU INSTRCTION.

would store away the IC in its current state. If the processor answers an interrupt it would add one (IC+1) to the IC. Word 5 contains the computed address that is being worked upon. The remainder of the word contains the control unit flags. These flags are restored in an rcu instruction.

Word 6 contains the instruction register at the point of interrupt if the instruction is not an rpd. The IBRA from the CD board is stored away in word 7. If the processor took a page fault on an even instruction, the even instruction would be in Word 6 and the add instruction in word 7. If the processor took a page fault on an add instruction, a copy of the add instruction would be in both words 6 and 7. For an rpd instruction, the processor stores the IBRA even.

The first three cycles of an rcu instruction can be viewed as being CU loads. On rcu pair 4 the processor looks at the restart bit to determine if the last cycle is a transfer to the R1WRY+2 that was loaded or a transfer to IC.

If the restart bit is on (MIF) that indicates the processor took an interrupt. The processor will use the ICT to transfer to. If the restart bit is off, then FRA, which is counting up the next location, will be used.

Fault on Instruction Fetch (FIF) also gets set on an interrupt. If the interrupt is on an EIS instruction, the processor will not store away IC+1. In that case the processor will store away the IC because it will return to complete the EIS instruction.

7.11 APPENDING UNIT OPERATION CHARTS

Figure 7-19 illustrates effective segment number generation, including the hardware ring mechanism. The description of the access violation faults follow the figure. See Group 6 Faults, Section 7.12. The current instruction is in the instruction working buffer (IWB).

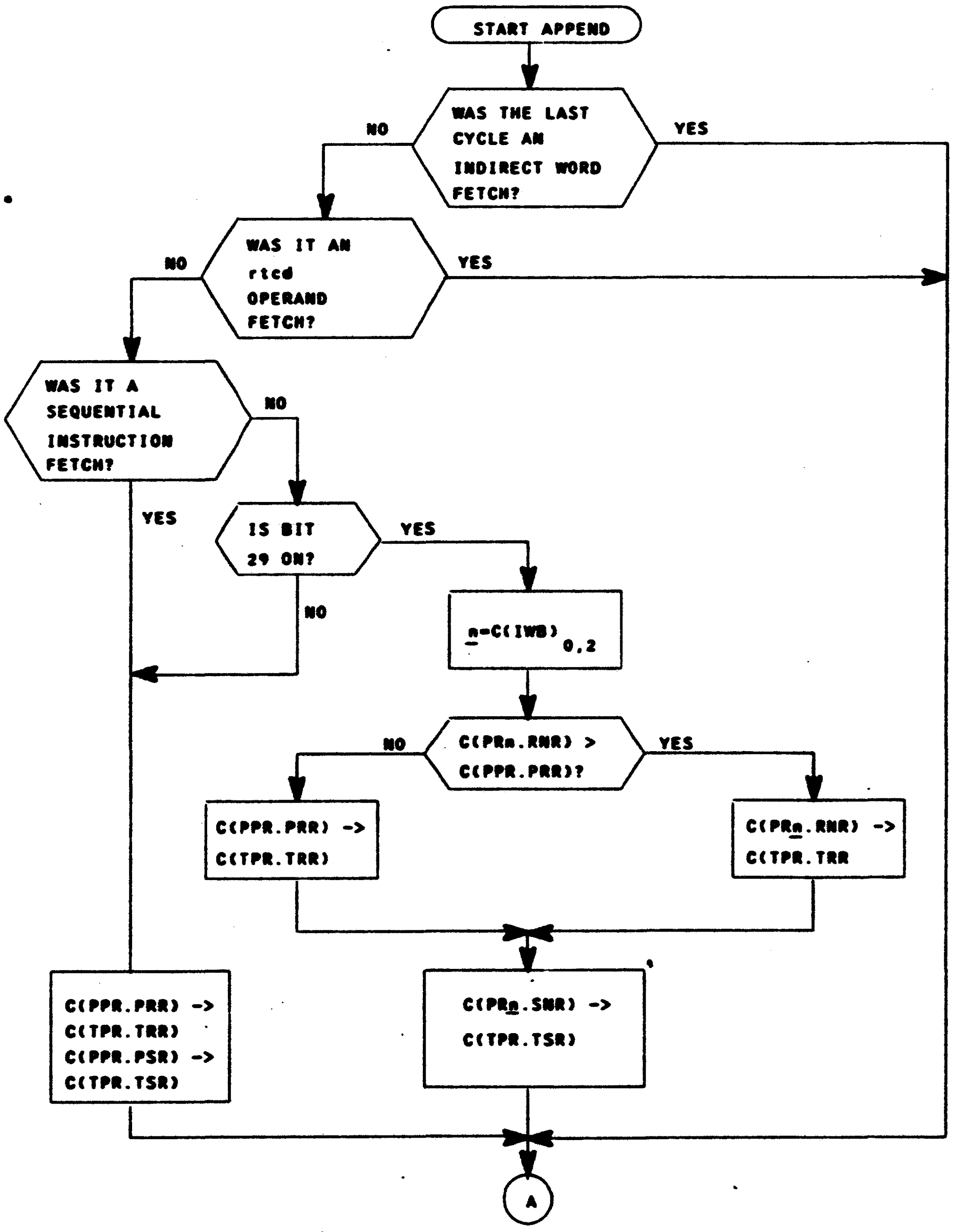

START APPEND
WAS THE LAST CYCLE AN INDIRECT WORD FETCH?
NO
YES
WAS IT AN rtcd OPERAND FETCH?
NO
YES
WAS IT A SEQUENTIAL INSTRUCTION FETCH?
NO
YES
IS BIT 29 ON?
YES
NO
n=C(IWB) 0,2
C(PRn.RNR) > C(PPR.PRR)?
NO
YES
C(PPR.PRR) -> C(TPR.TRR)
C(PRn.RNR) -> C(TPR.TRR
C(PPR.PRR) -> C(TPR.TRR) C(PPR.PSR) -> C(TPR.TSR)
C(PRn.SNR) -> C(TPR.TSR)
A

FIGURE 7-19. COMPLETE APPENDING UNIT OPERATION FLOWCHART (1 of 12)

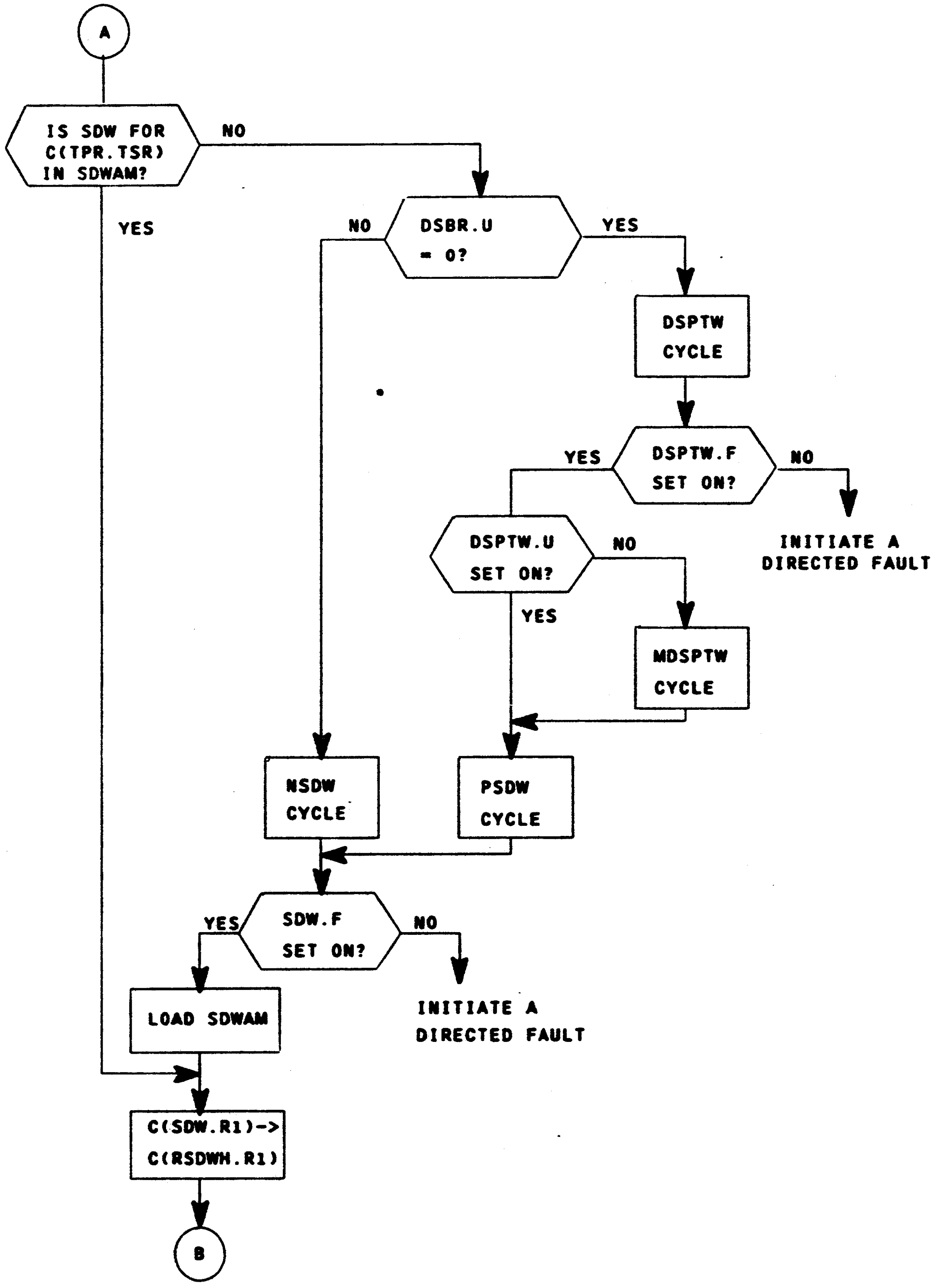


FIGURE 7-19. COMPLETE APPENDING UNIT OPERATION FLOWCHART (2 of 12)

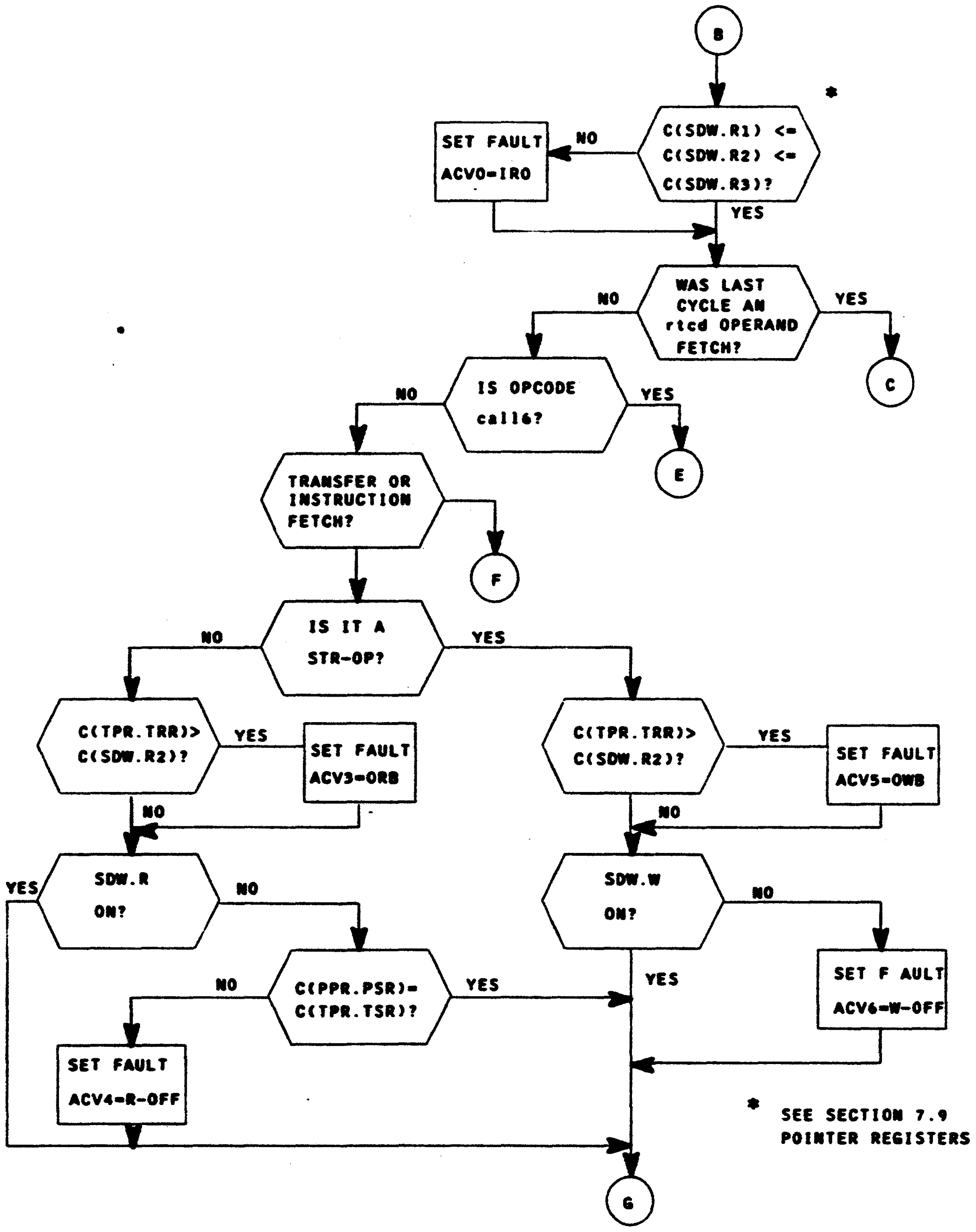


FIGURE 7-19. COMPLETE APPENDING UNIT OPERATION FLOWCHART (3 of 12)

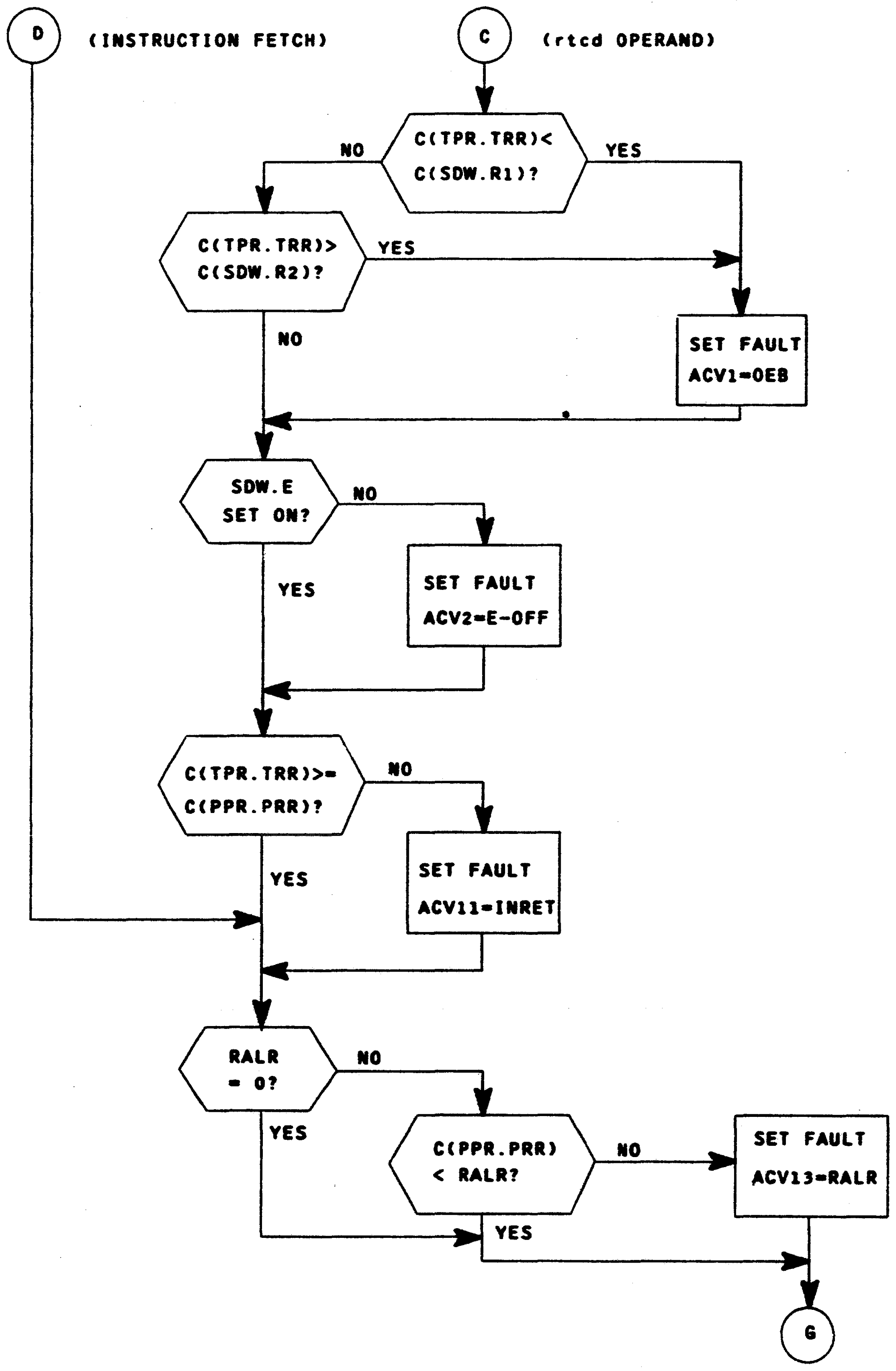


FIGURE 7-19. COMPLETE APPENDING UNIT OPERATION FLOWCHART (4 of 12)

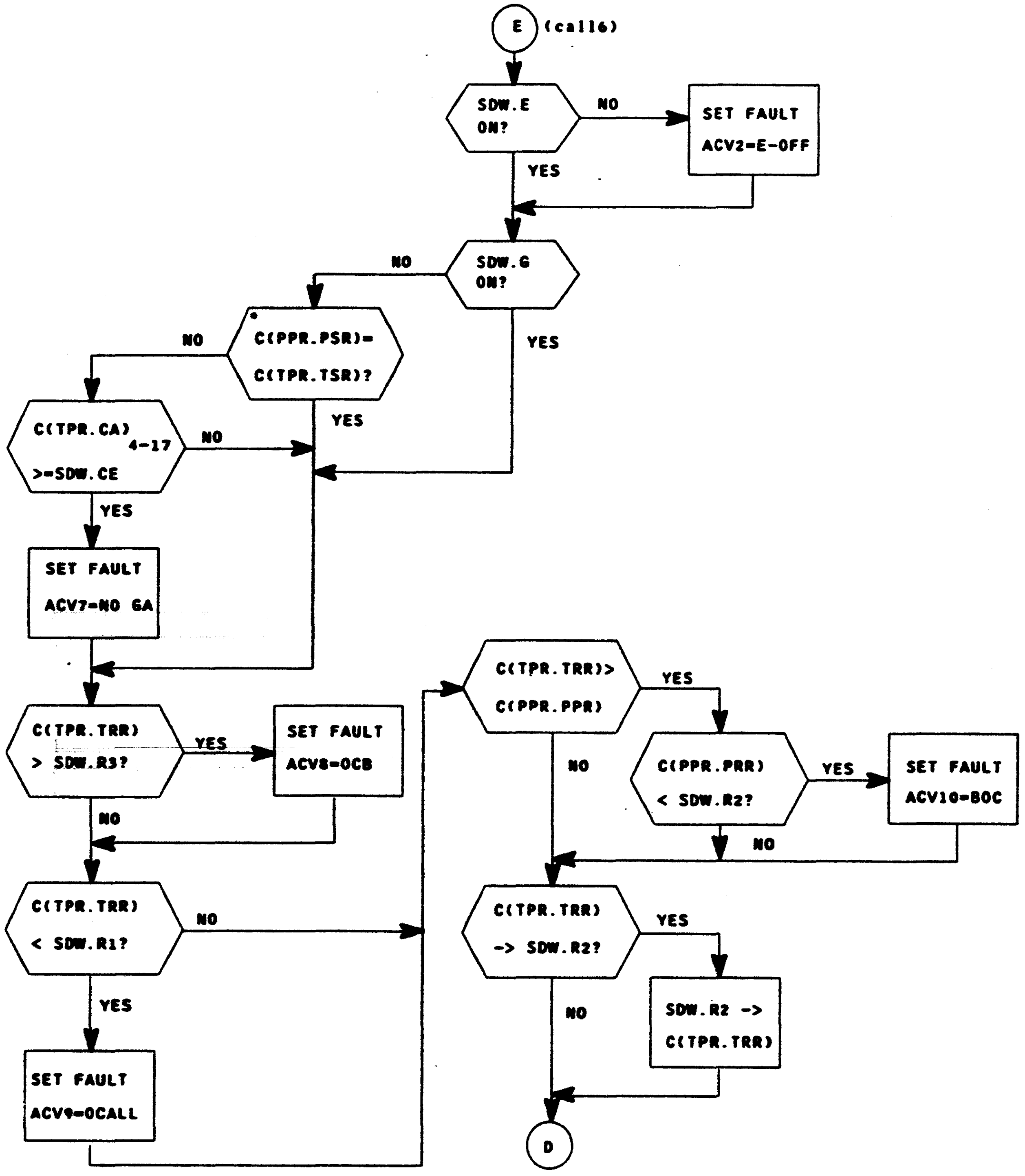


FIGURE 7-19. COMPLETE APPENDING UNIT OPERATION FLOWCHART
(5 of 12)

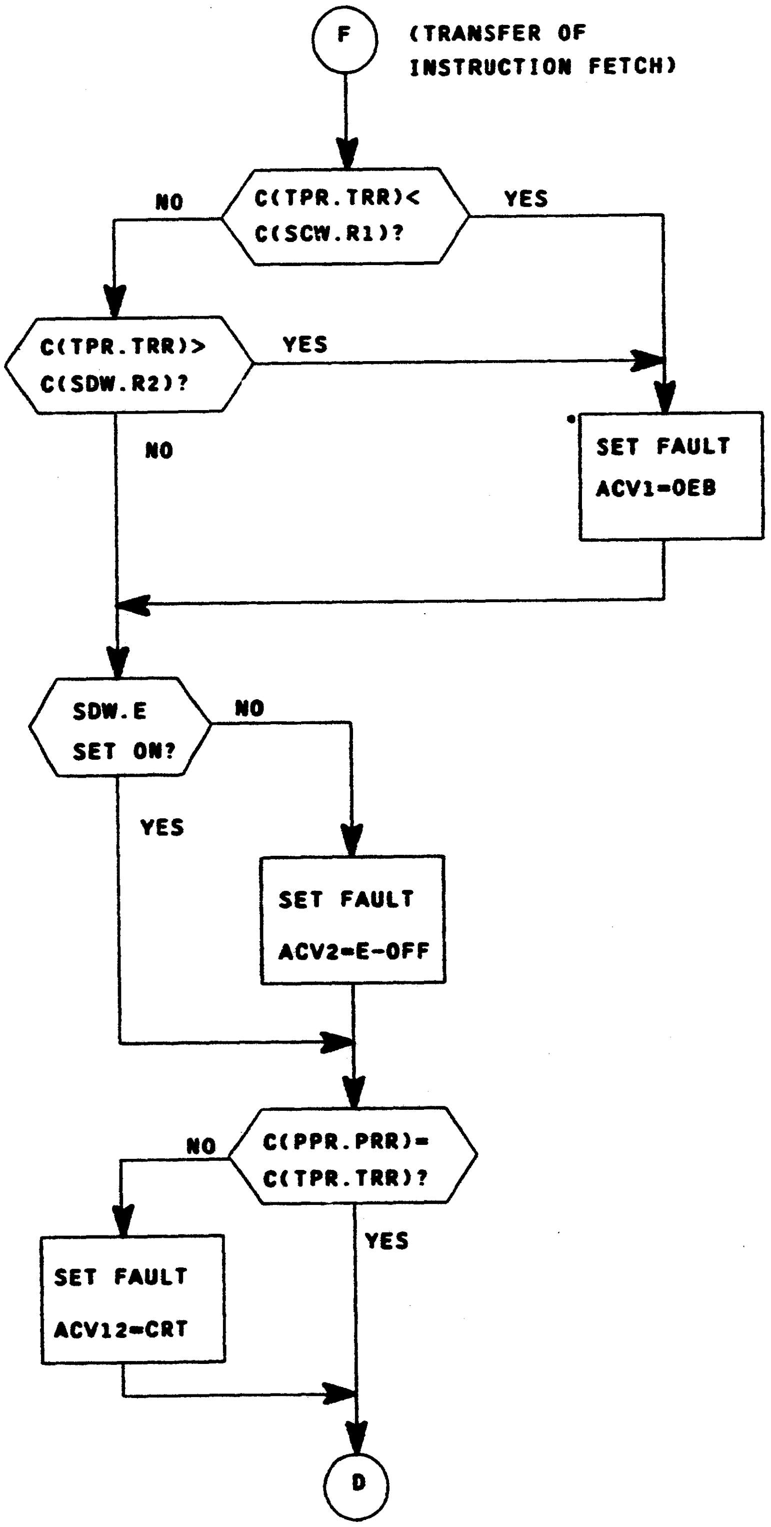


FIGURE 7-19. COMPLETE APPENDING UNIT OPERATION FLOWCHART (6 of 12)

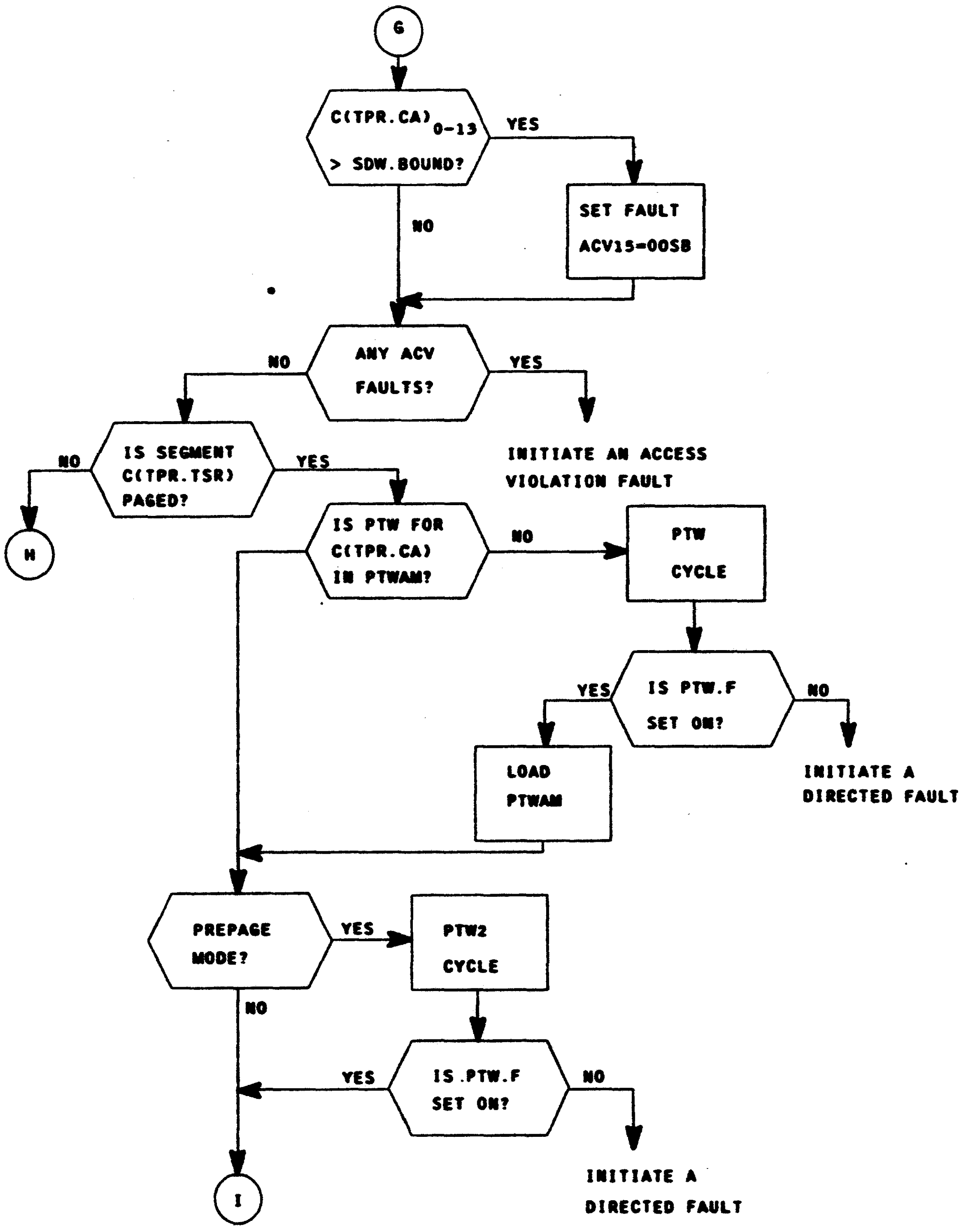

G
C(TPR.CA)0-13 > SDW.BOUND?
YES
NO
SET FAULT ACV15=OOSB
ANY ACV FAULTS?
NO
YES
INITIATE AN ACCESS VIOLATION FAULT
IS SEGMENT C(TPR.TSR) PAGED?
NO
YES
H
IS PTW FOR C(TPR.CA) IN PTWAM?
NO
PTW CYCLE
IS PTW.F SET ON?
YES
NO
LOAD PTWAM
INITIATE A DIRECTED FAULT
PREPAGE MODE?
YES
NO
PTW2 CYCLE
IS PTW.F SET ON?
YES
NO
INITIATE A DIRECTED FAULT
I

FIGURE 7-19. COMPLETE APPENDING UNIT OPERATION FLOWCHART (7 of 12)

HONEYWELL CONFIDENTIAL & PROPRIETARY

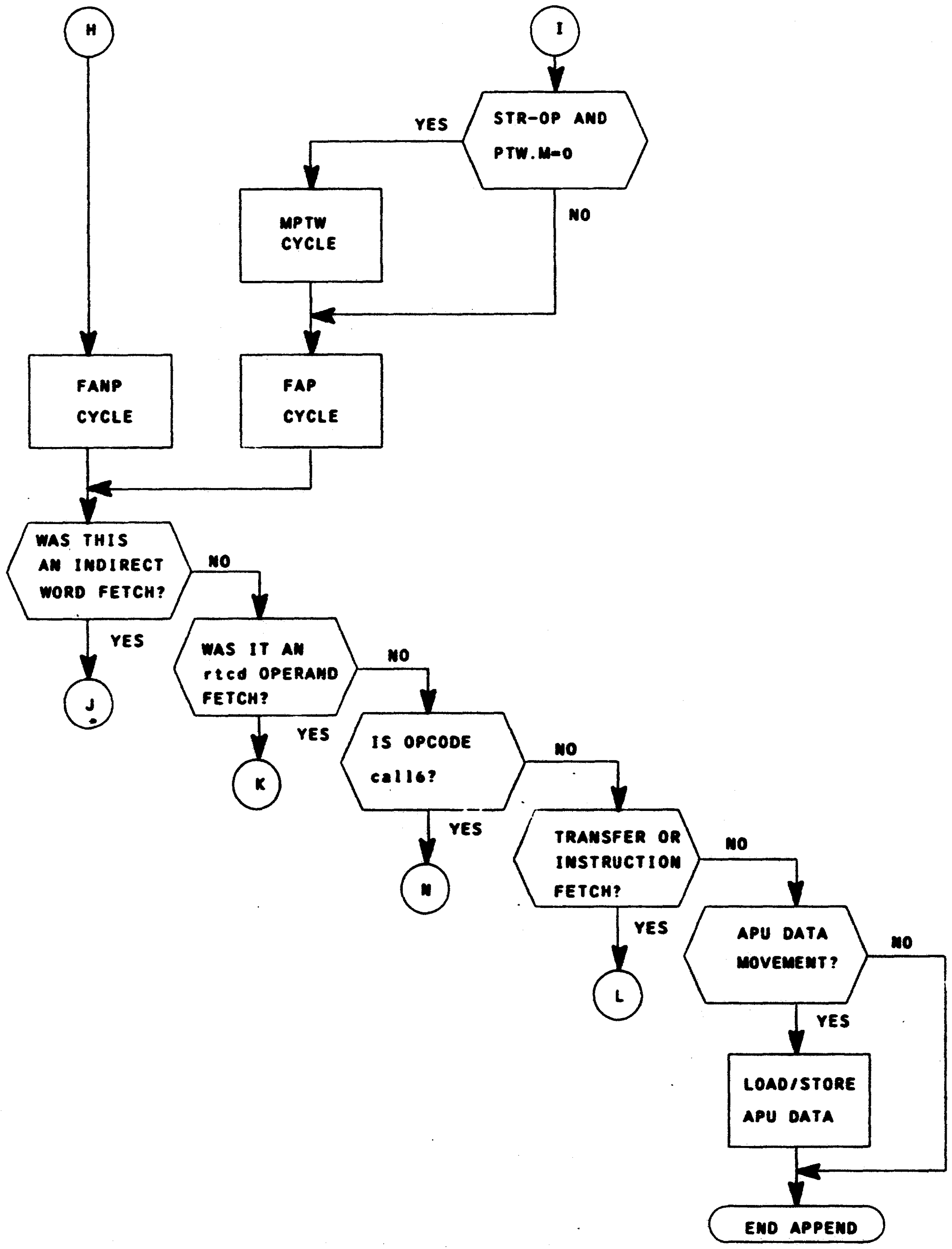


FIGURE 7-19. COMPLETE APPENDING UNIT OPERATION FLOWCHART
(8 of 12)

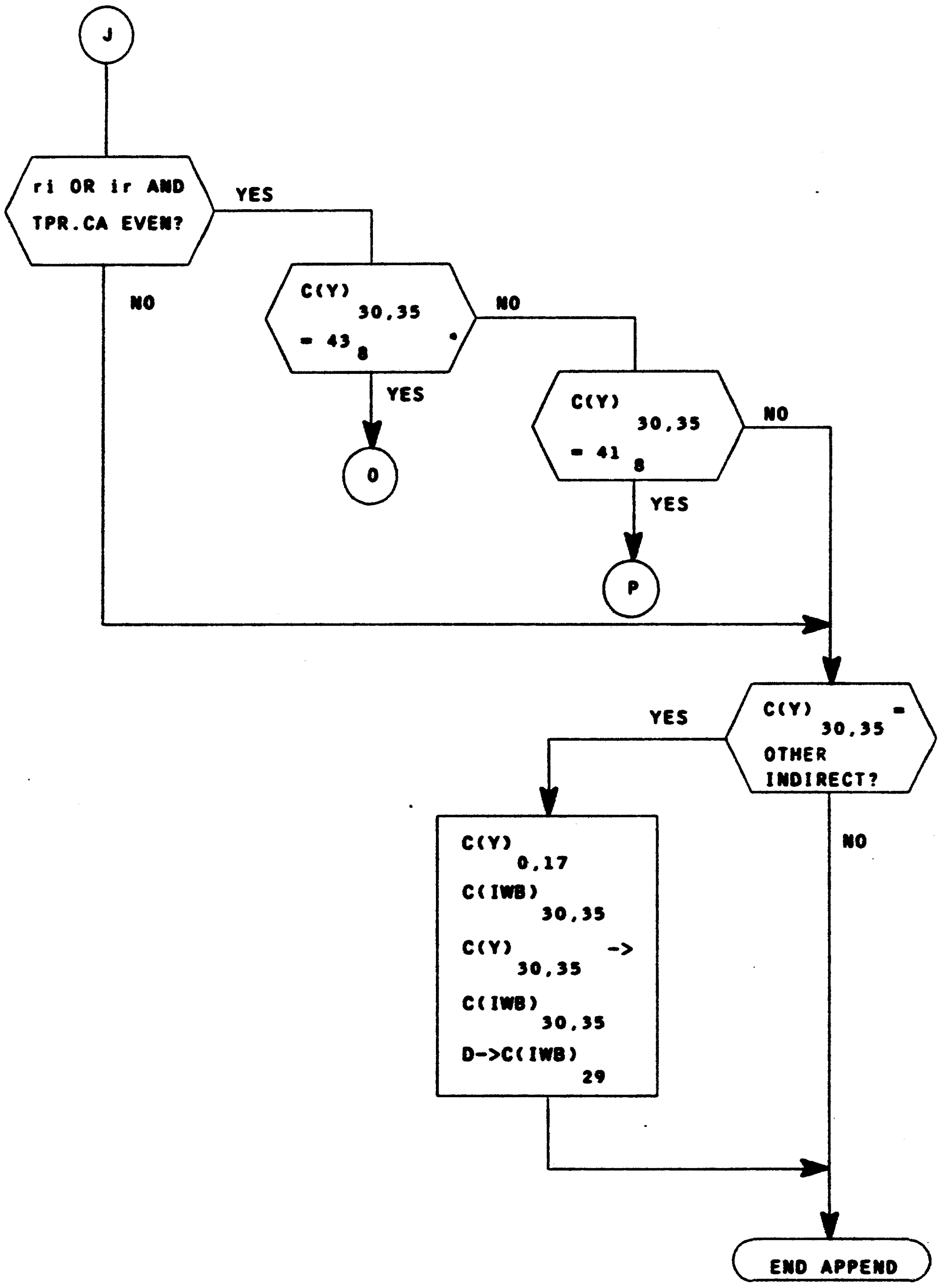

J
ri OR ir AND TPR.CA EVEN?
YES
NO
C(Y)30,35 = 43 8
NO
YES
O
C(Y)30,35 = 41 8
NO
YES
P
C(Y)30,35 = OTHER INDIRECT?
YES
NO
C(Y)0,17 C(IWB)30,35 C(Y)30,35 -> C(IWB)30,35 D->C(IWB)29
END APPEND

FIGURE 7-19. COMPLETE APPENDING UNIT OPERATION FLOWCHART (9 of 12)

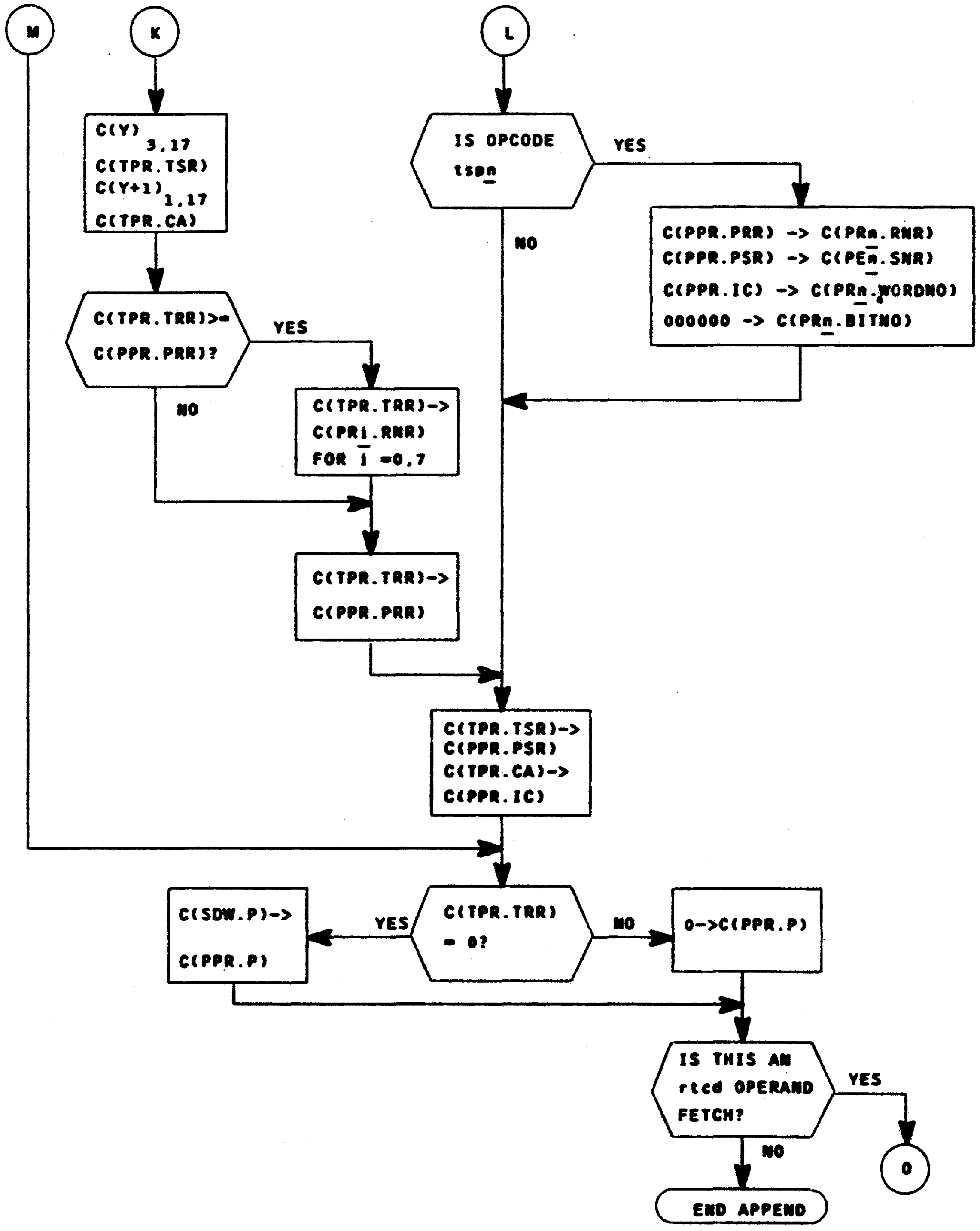


FIGURE 7-19. COMPLETE APPENDING UNIT OPERATION FLOWCHART (10 of 12)

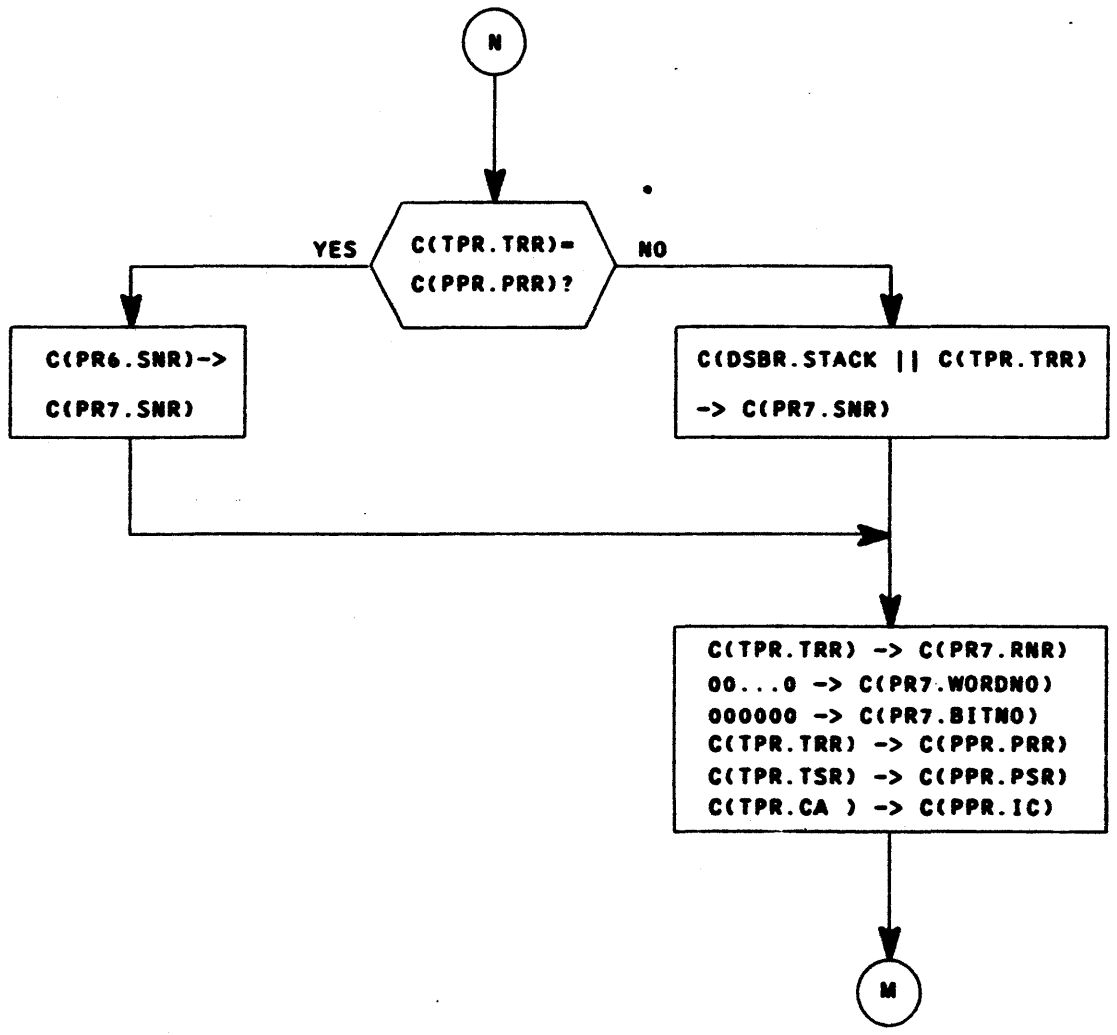
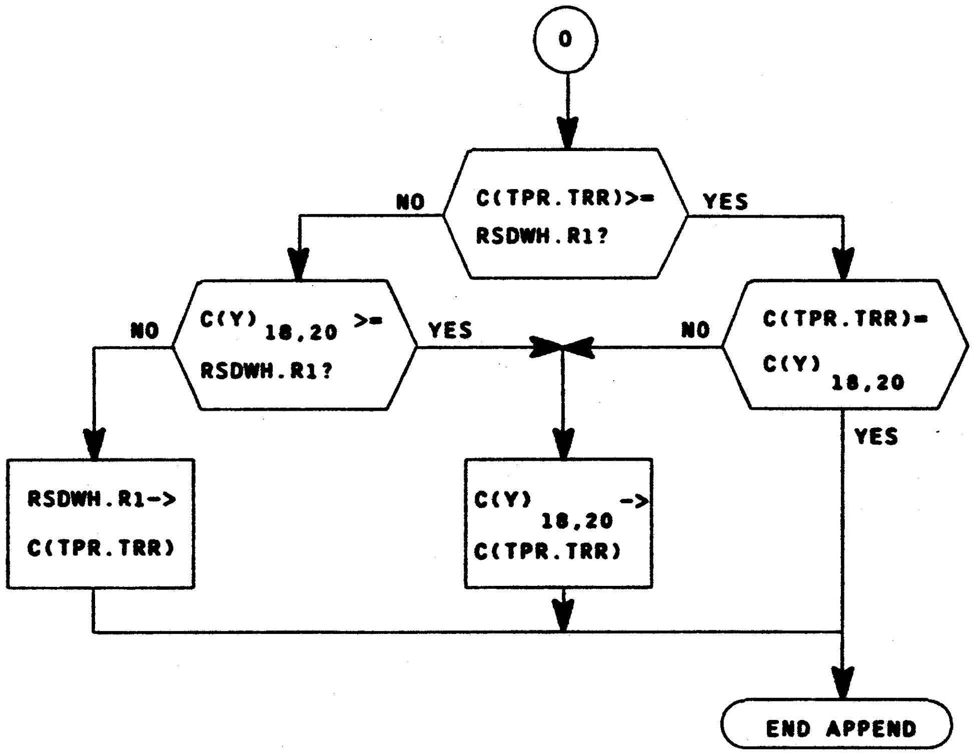


FIGURE 7-19. COMPLETE APPENDING UNIT OPERATION FLOWCHART (11 of 12)

HONEYWELL CONFIDENTIAL & PROPRIETARY



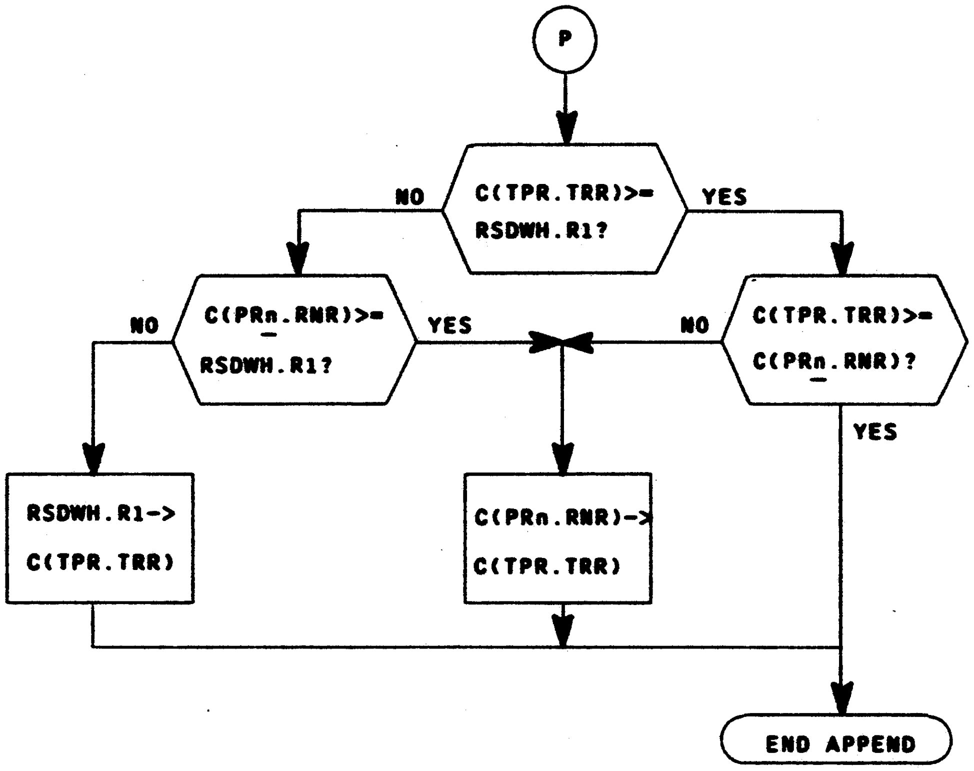


FIGURE 7-19 COMPLETE APPENDING UNIT OPERATION FLOWCHART (12 of 12)

HONEYWELL CONFIDENTIAL & PROPRIETARY

7.12 GROUP 6 FAULTS

Directed Faults 0-3

A faulted segment descriptor word (SDW) or page table word (PTW) with the corresponding directed fault number has been fetched by the appending unit.

Access Violation

The appending unit has detected one of the several access violations below. Word 1 of the Control Unit Data contains status bits for the condition.

1.	Not in read bracket	(ACV3=ORB)
2.	Not in write bracket	(ACV5=OWB)
3.	Not in execute bracket	(ACV1=OEB)
4.	No read permission	(ACV4=R-OFF)
5.	No write permission	(ACV6=W-OFF)
6.	No execute permission	(ACV2=E-OFF)
7.	Invalid ring crossing	(ACV12=CRT)
8.	Call limiter fault	(ACV7=NO GA)
9.	Outward call	(ACV9=OCALL)
10.	Bad outward call	(ACV10=BOC)
11.	Inward return	(ACV11=INRET)
12.	Ring alarm	(ACV13=RALR)
13.	Associative memory error	
14.	Out of segment bounds	(ACV15=OOSB)
15.	Illegal ring order	(ACV0=IRO)
16.	Out of call brackets	(ACV8=OCB)