IBM Maintenance Library

3125 Processing Unit Microinstructions

SY33-1058-1

Third Edition (October 1973)

This manual obsoletes SY33-1058-0. Changes are continually made to the information in this manual; any such changes will be reported in subsequent revisions or Technical Newsletters.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

Forms for reader's comments are provided at the back of the manual. If the forms have been removed, comments may be addressed to IBM Laboratories, Product Publications, Dept 3179, 703 Boeblingen/Wuertt., P.O. Box 210, Germany. Comments become the property of IBM.

Preface

This manual provides information on the IBM 3125 Processing Unit's instruction processing unit (IPU), input/output processor (IOP), and service processor (SVP) microprogram codes. Its main purpose is to explain the functions of:

- Each microinstruction group,
- Each microinstruction and
- Each microinstruction bit.

It also enables the reader to determine the mnemonic by analyzing the bit pattern of a given instruction word.

The reader should have a basic knowledge of the IPU, IOP and SVP data flow of the IBM System/370 Model 125.

Prerequisite Reading

3125 Processing Unit, General System Information, Maintenance Library Manual, Order No. SY33-1059.

Associated Publications

System Library Manuals

IBM System/370 Principles of Operation, Order No. GA22-6821.
IBM System/370 Model 125 Functional Characteristics, Order No. GA33-1506.

Maintenance Library Manuals

IBM 3125 Processing Unit, Power Supplies, Order No. SY33-1060.
IBM 3125 Processing Unit, Main Storage Controller, Order No. SY33-1061.
IBM 3125 Processing Unit, Instruction Processing Unit, Order No. SY33-1062.
IBM 3125 Processing Unit, Input/Output Processor, Order No. SY33-1063.
IBM 3125 Processing Unit, Magnetic Tape Adapter, Order No. SY33-1064.
IBM 3125 Processing Unit, Service Processor Subsystem, Order No. SY33-1065.
IBM 3125 Processing Unit, Main Storage, Order No. SY33-1066.
IBM 3125 Processing Unit, Multiplexer Channel, Order No. SY33-1067.
IBM 3125 Processing Unit, 2560 Attachment, Front End, Order No. SY33-1068.
IBM 3125 Processing Unit, 5425 Attachment, Front End, Order No. SY33-1069.
IBM 3125 Processing Unit, 3525 Attachment, Front End, Order No. SY33-1070.
IBM 3125 Processing Unit, 3504 Attachment, Front End, Order No. SY33-1071.
IBM 3125 Processing Unit, 1403 Attachment, Front End, Order No. SY33-1072.
IBM 3125 Processing Unit, Direct Disk Attachment, Order No. SY33-1073.
IBM 3125 Processing Unit, Integrated Console Printer Attachment, Order No. SY33-1074.
IBM 3125 Processing Unit, Integrated Communications Adapter, Part B/M 1876075.
IBM 3125 Processing Unit, Installation Instructions, Part 4014001.
IBM 3125 Central Test Manual. Contains pages appropriate to the individual 3125 Processing Unit.
IBM 3125 Processing Unit, Parts Catalog, Order No. S135-1000.

Contents

Section 1: IPU Microprogram Codes

IPU Microinstruction Group Determination

Instruction Bits 5 6 7	Instruction Bits 8 9	Instruction Bits 11	Instruction Group
0 0 0	x x	x	9
0 0 x	0 x	x	9
0 0 1	1 x	x	10
0 1 x	x x	x	10
1 0 0	0 0	x	13
1 0 0	0 1	0	11
1 0 0	0 1	1	8
1 0 0	1 0	x	12
1 0 1	x 0	x	1
1 0 1	x 1	x	2
1 1 0	x x	x	3
1 1 1	0 0	x	7
1 1 1	0 1	x	6
1 1 1	1 0	x	4
1 1 1	1 1	x	5

IPU Microinstructions by Group

Group 1

Bit 11	Mnemonic
0	LC(R)
1	LT(R)

Group 2

Bits 8	Bits 11	Bits 13	Mnemonic
0	0	0	IRC(R)
0	0	1	ILC(R)
0	1	0	IRT(R)
0	1	1	ILT(R)
1	0	0	ZIRC(R)
1	0	1	ZILC(R)
1	1	0	ZIRT(R)
1	1	1	ZILT(R)

Group 3

Bits 14 15	Bits 8 9 12 20	Mnemonic	
0 0	0 0 x x	OL(R)	direct addressing
	0 1 0 x	LTOL(R)	
	0 1 1 x	LTCOL(R)	
	1 0 x 0	OL(R)	indirect addressing
	1 0 x 1	OSL(R)	
	1 1 x 0	OBL(R)	
	1 1 x 1	OSBL(R)	
0 1	0 0 x x	AL(R)	direct addressing
	0 1 0 x	LTAL(R)	
	0 1 1 x	LTCAL(R)	
	1 0 x 0	AL(R)	indirect addressing
	1 0 x 1	ASL(R)	
	1 1 x 0	ABL(R)	
	1 1 x 1	ASBL(R)	
1 0	0 0 x x	NL(R)	direct addressing
	0 1 0 x	LTNL(R)	
	0 1 1 x	LTCNL(R)	
	1 0 x 0	NL(R)	indirect addressing
	1 0 x 1	NSL(R)	
	1 1 x 0	NBL(R)	
	1 1 x 1	NSBL(R)	
1 1	0 0 x x	XL(R)	direct addressing
	0 1 0 x	LTXL(R)	
	0 1 1 x	LTCXL(R)	
	1 0 x 0	XL(R)	indirect addressing
	1 0 x 1	XSL(R)	
	1 1 x 0	XBL(R)	
	1 1 x 1	XSBL(R)	

Group 4

Bits 14 15	Bits 16 22	Mnemonic
0 0	0 0	OMS(R)
	1 0	OSMS(R)
0 1	0 0	AMS(R)
	1 0	ASMS(R)
1 0	0 0	NMS(R)
	1 0	NSMS(R)
1 1	0 0	XMS(R)
	1 0	XSMS(R)
	0 1	XMSC(R)

Group 5

Bits 11 16	Mnemonic
0 0	MSC(R)
1 0	MST(R)
1 1	MSTIX(R)

Group 6

Bits 11 16 17	Mnemonic
0 0 0	MLC(R)
0 1 0	MKC(R)
1 0 0	MLT(R)
1 0 1	MSCTL(R)
1 1 0	MKT(R)

Group 7

Bits 14	Bits 15	Bits 16	Bits 19	Mnemonic
0	0	0	0	OML(R)
		0	1	OLML(R)
		1	0	OMK(R)
		1	1	OLMK(R)
0	1	0	0	AML(R)
		0	1	ALML(R)
		1	0	AMK(R)
		1	1	ALMK(R)
1	0	0	0	NML(R)
		0	1	NLML(R)
		1	0	NMK(R)
		1	1	NLMK(R)
1	1	0	0	XML(R)
		0	1	XLML(R)
		1	0	XMK(R)
		1	1	XLMK(R)

Group 8

The mnemonic is 'T' or 'TR', depending on the status of bit 10.

Group 9 (Branch type I)

Bits 7	Bits 8	Mnemonic
0	0	BT
0	1	BTS
1	0	BTM

Group 10

The mnemonic is 'BC' (Branch conditional).

Group 11

Bits 16	Bits 18	Bits 19	Mnemonic
0	0	0	SRC(R), or NOP(R) if bits 16...23 = all zeros
0	0	1	SRT(R)
0	1	0	SRCN(R)
0	1	1	SRTN(R)
1	0	0	SLC(R)
1	0	1	SLT(R)
1	1	0	SLCN(R)
1	1	1	SLTN(R)

Group 12

Bits 11	Bits 13	Bits 16	Mnemonic
0	0	0	SNSCR(R)
0	1	0	SNSCL(R)
1	0	0	SNSTR(R)
1	1	0	SNSTL(R)
x	x	1	CTL(R)

Group 13

The mnemonic is;
TRB = Translate and branch (Bit 10 = off), or
TRBR = Translate, branch and return (Bit 10 = on).

Listing of IPU Mnemonics

Mnemonic	Description	Group
ABL(R)	Add to IPU local storage, suppress ALU Bits 0...7, (then return).	3
AL(R)	Add to IPU local storage, (then return).	3
ALMK(R)	Add to IPU local storage and key storage, (return).	7
ALML(R)	Add to IPU local storage and MSC local storage, (then return).	7
AMK(R)	Add to MSC key storage, (then return).	7
AML(R)	Add to MSC local storage, (then return).	7
AMS(R)	Add to main storage, (return).	4
ASBL(R)	Add with Six correction to IPU local storage, suppress ALU Bits 0...7, (then return).	3
ASL(R)	Add with Six correction to IPU local storage, (then return).	3
ASMS(R)	Add with Six correction to MSC main storage, (then return).	4
BC	Branch Conditional.	10
BT	Branch on Test (no level switching).	9
BTM	Branch on Test to Main routine.	9
BTS	Branch on Test to Sub routine.	9
CTL(R)	Control, (then return).	12
ILC(R)	Immediate data Left adjusted to CDR, (then return).	2
ILT(R)	Immediate data Left adjusted to TDR, (then return).	2
IRC(R)	Immediate data Right adjusted to CDR, (then return).	2
IRT(R)	Immediate data Right adjusted to TDR, (then return).	2
LC(R)	Load CDR, (then return).	1
LT(R)	Load TDR, (then return).	1
LTAL(R)	IPU Local storage to TDR, Add with CDR into IPU Local storage, (then return).	3
LTCAL(R)	IPU Local storage to TDR and CDR, Add both into IPU Local storage, (then return).	3
LTCNL(R)	IPU Local storage to TDR and CDR, logically AND into IPU Local storage, (then return).	3
LTCOL(R)	IPU Local storage to TDR and CDR, logically OR into IPU Local storage, (then return).	3
LTCXL(R)	IPU Local storage to TDR and CDR, exclusively OR into IPU Local storage, (then return).	3
LTNL(R)	IPU Local storage to TDR, logically AND into IPU Local storage, (then return).	3
LTOL(R)	IPU Local storage to TDR, logically OR into IPU Local storage, (then return).	3
LTXL(R)	IPU Local storage to TDR, exclusively OR into IPU Local storage, (then return).	3
MKC(R)	Read from MSC Key storage into CDR, (then return).	6
MKT(R)	Read from MSC Key storage into TDR, (then return).	6
MLC(R)	Read from MSC Local store into CDR, (then return).	6
MLT(R)	Read from MSC Local store into TDR, (then return).	6
MSC(R)	Read from Main storage into CDR, (then return).	5
MSCTL(R)	Main Storage Control.	6
MST(R)	Read from Main Storage into TDR, (then return).	5
MSTIX(R)	Read from Main Storage into TDR, test for I-Phase exception, (then return).	5
NBL(R)	AND to IPU Local storage, suppress ALU Bit 0...7, (then return).	3
NL(R)	AND to IPU Local storage, (then return).	3
NLMK(R)	AND to IPU Local storage and MSC Key storage, (then return).	7
NLML(R)	AND to IPU Local storage and MSC Local storage, (then return).	7
NMK(R)	AND to MSC Key storage, (then return).	7
NML(R)	AND to MSC Local storage, (then return).	7
NMS(R)	AND to Main storage, (then return).	4

NOP(R)	No operation (then return).	11
NSBL(R)	AND with Six correction to IPU Local storage, suppress ALU Bit 0...7, (then return).	3
NSL(R)	AND with Six correction to IPU Local storage, (then return).	3
NSMS(R)	Add with Six correction to Main storage, (return).	4
OBL(R)	OR to IPU Local storage, suppress ALU Bit 0...7, (then return).	3
OL(R)	OR to IPU Local storage, (then return).	3
OLMK(R)	OR to IPU Local storage and MSC Key storage, (then return).	7
OLML(R)	OR to IPU Local storage and MSC Local storage, (then return).	7
OMK(R)	OR to MSC Key storage, (then return).	7
OML(R)	OR to MSC Local storage, (then return).	7
OMS(R)	OR to Main storage, (then return).	4
OSBL(R)	OR with Six correction to IPU Local storage, suppress ALU Bit 0...7, (then return).	3
OSL(R)	OR with Six correction to IPU Local storage, (then return).	3
OSMS(R)	OR with Six correction to Main storage, (return).	4
SLC(R)	Shift Left to CDR, (then return).	11
SLCN(R)	Shift Left to CDR, using Negative shift amount, (then return).	11
SLT(R)	Shift Left to TDR, (then return).	11
SLTN(R)	Shift Left to TDR, using Negative shift amount, (then return).	11
SNSCL(R)	Sense into CDR Left, (then return).	12
SNSCR(R)	Sense into CDR Right, (then return).	12
SNSTL(R)	Sense into TDR Left, (then return).	12
SNSTR(R)	Sense into TDR Right, (then return).	12
SRC(R)	Shift right into CDR, (then return).	11
SRCN(R)	Shift Right into CDR, using Negative shift amount, (then return).	11
SRT(R)	Shift Right into TDR, (then return).	11
SRTN(R)	Shift Right into TDR, using negative shift amount, (then return).	11
T(R)	Test, (then return).	8
TRB(R)	Translate and Branch, (then return).	13
XBL(R)	Exclusive OR to IPU Local storage, suppress ALU Bit 0...7, (then return).	3
XL(R)	Exclusive OR to IPU Local storage, (then return).	3
XLMK(R)	Exclusive OR to IPU Local storage and MSC Key storage, (then return).	7
XLML(R)	Exclusive OR to IPU Local storage and MSC Local storage, (then return).	7
XMK(R)	Exclusive OR to MSC Key storage, (then return).	7
XML(R)	Exclusive OR to MSC Local storage, (then return).	7
XMS(R)	Exclusive OR to Main storage, (then return).	4
XMSC(R)	Exclusive OR to Main Storage, read from Main Storage into CDR, (then return).	4
XSBL(R)	Exclusive OR with Six correction to IPU Local storage, suppress ALU Bit 0...7, (then return).	3
XSL(R)	Exclusive OR with Six correction to IPU Local storage, (then return).	3
XSMS(R)	Exclusive OR with Six correction to Main storage, (then return).	4
ZILC(R)	Zero set, then place Immediate data Left adjusted into CDR, (then return).	2
ZILT(R)	Zero set, then place Immediate data Left adjusted into TDR, (then return).	2
ZIRC(R)	Zero set, then place Immediate data Right adjusted into CDR, (then return).	2
ZIRT(R)	Zero set, then place Immediate data Right adjusted into TDR, (then return).	2

Explanation of IPU Microinstruction Groups

Group 1: Halfword from Local Store to TDR or CDR

Primary Function: One halfword is fetched from the IPU local storage and set into either the True Data Register (TDR) or Complement Data Register (CDR).

Secondary Functions:

- The contents of TDR can be propagated to the local storage address registers (LSARs) 0 and 1; or 2 and 3; or 0,1,2,and 3 (see under LSAR setting cases).
- The invert switch can be set to true, invert, force ones, force zeros.
- The local storage can be addressed direct or indirect.
- A return from a subroutine can be initiated.

Common Layout of Group 1 Instructions

```
           ----------
           .
           1        Invert Bit
           2        Parity Bit
Byte0      3       PFCN
           .
           5  1
           6  0  Op Code Group 1
           7  1
         ------------------------
           8  0 =direct/1=indirect
           9  0 =LS to TDR or CDR (op code)
          10  1 =Leave subroutine
          11  1 =TDR/0=CDR
Byte1     12  1 =set LSAR0 and 1
          13  1 =set LSAR2 and 3
          14      invert switch function
          15
         ------------------------------------
          16                        16
          17                        17 High Order
          18                        18 Portion
          19 direct      indirect   19
Byte2     20 LS addr     LS addr    20 no function
          21                        21 1=LSAR0
          22                        22
          23                        23 LSAR addr
         ------------------------------------
```

Note: Bits 0 and 4 of Byte 0 do not exist

Bit Function Description

Bit 1, Invert. Generated internally by hardware. (When instruction is inverted by SVP.)
Bit 2, Parity Bit. Generated by assembler program. (To obtain an odd number of zero bits.)
Bit 3, PFCN - Parity Function Bit. Generated by assembler program. (To obtain an odd number of control gates.)
Bit 5,6,7 and 9, Op Code. These bits represent a unique pattern that is common to all group 1 instructions.

Bit 8, Direct/Indirect. This bit determines how the IPU local storage is to be addressed. Direct addressing is employed when bit 8 is 0, indirect addressing is used when bit 8 is 1.

Direct. When direct is specified, bits 16 to 23 of the instruction represent a binary number (from 0 to 255) which is used to address the local storage. Bit 16 has the highest binary value, bit 23 has the lowest binary value.

Indirect. When indirect is specified, either one of two methods is used as determined by bit 21 (LSAR 0).

Bit 21 off (0). With bit 21 off, instruction bits 16 through 19 represent the high order portion of the address. The contents of the local store address register (LSAR) addressed by instruction bits 22 and 23 represent the low order portion of the address. Both portions are said to be concatenated (chained) to form one logical bit string.

Bit 21 on (1). With bit 21 on, instruction bits 16 through 19 are ignored. The high order portion of the address then is provided by the contents of LSAR 0. The low order portion is provided by the LSAR addressed by instruction bits 22 and 23. These bits may specify any LSAR including LSAR 0. Both LSAR contents are said to be concatenated to form one logical bit string.

Bit 10, Leave Subroutine. This bit allows the IPU microprogram to re-enter the program level that was in effect prior to the last level switching operation. In this manner, the program may return from a subroutine to its original point of continuation.

The instruction which has bit 10 on is still part of the subroutine and so is the next following instruction. However, the instruction that follows thereafter is the first one of the continued previous routine.

Bit 11, Set TDR/CDR. This bit determines the register into which the data (fetched from local store) will enter. (0 = CDR / 1 = TDR)

Bit 12, Set LSAR 0 and 1. Bit 12 provides the means to propagate the left half (one byte) contents of TDR into LSAR's 0 and 1. When bit 12 is on (1) bits 0-3 of TDR are set into LSAR 0, bits 4-7 of TDR are set into LSAR 1. The readout from TDR is non-destructive.

Bit 13, Set LSAR 2 and 3. When bit 13 is on (1) bits 8-11 of TDR are set into LSAR 2, bits 12-15 of TDR are set into LSAR 3. The readout from TDR is non-destructive.

Bit 14 and 15. These bits provide four different patterns which determine the function of the invert switch for some later arithmetic/logic operation, (see parameters).

Bits 22 and 23, LSAR Address. When indirect addressing is used, both bits represent a binary number (from 0 to 3) that addresses one of the LSAR's (see parameters).

Valid Parameters for Group 1 Instructions

The parameters are listed as cases C1....CXX which will also appear in the microprogram listings to allow orientation.

LSAR Setting Cases

	Bit 12	Bit 13	
C1 =	0	0	= no LSAR setting
C2 =	1	0	= TDR bits 0....3 to LSAR0, and TDR bits 4....7 to LSAR1
C3 =	0	1	= TDR bits 8...11 to LSAR2, and TDR bits 12..15 to LSAR3
C4 =	1	1	= TDR bits 0....3 to LSAR0, and TDR bits 4....7 to LSAR1, and TDR bits 8...11 to LSAR2, and TDR bits 12..15 to LSAR3

Invert Switch Cases

	Bit 14	Bit 15	
C5 =	0	0	= invert
C6 =	0	1	= true
C7 =	1	0	= force ones
C8 =	1	1	= force zeros

Local Store Address Cases

	Bit 8	Bit 21	Bit 22	Bit 23	
C9 =	0	-	-	-	= instr.bits 16....23
C10 =	1	0	0	0	= instr.bits 16....19//LSAR0
C11 =	1	0	0	1	= instr.bits 16....19//LSAR1
C12 =	1	0	1	0	= instr.bits 16....19//LSAR2
C13 =	1	0	1	1	= instr.bits 16....19//LSAR3
C14 =	1	1	0	0	= LSAR0//LSAR0
C15 =	1	1	0	1	= LSAR0//LSAR1
C16 =	1	1	1	0	= LSAR0//LSAR2
C17 =	1	1	1	1	= LSAR0//LSAR3

Group 2: Immediate Data to TDR or CDR

Primary Function: A data byte provided by the instruction is set into either TDR or CDR in either right or left adjusted position.

Secondary Functions:

- The remainder of TDR or CDR may either keep its original data or may be set to zeros.
- Data from TDR may be further distributed to LSAR's 0 and 1.
- The invert switch can be set to true, invert, force ones, force zeros.

Common Layout of Group 2 Instructions

```
    ----------
          .
          1        Invert Bit
          2        Parity Bit
Byte0     3_______PFCN________
          .
          5  1
          6  0    Op code Group 2
          7  1
    ------8--0-----------------------
          9  1 = keep/1=reset
         10  1 = immediate to TDR/CDR (Op Code)
Byte1    11  1 = Leave subroutine
         12  1 = TDR/0=CDR
         13  1 = set LSAR0 and 1
         13  1 = left/0=right_______
         14
    -----15------invert switch function
         16
         17
         18       immediate
Byte2    19       data byte
         20
         21
         22
    -----23--------------------------
```

Bit Function Description

Bit 1, Invert. Generated internally by hardware.

Bit 2, Parity Bit. Generated by assembler program.

Bit 3, PFCN - Parity Function Bit. Generated by assembler program.

Bits 5,6,7 and 9, Op Code. These bits represent a unique pattern that is common to all group 2 instructions.

Bit 8, Keep/Reset. This bit determines whether the eight bits in TDR or CDR into which no data is set keep their original data or change to zero.

Bit 10, Leave Subroutine. This bit provides the means to return from a subroutine to the program level that was in effect prior to the last level switching operation. The instruction that has bit 10 on is still part of the subroutine and so is the next instruction. However, the instruction thereafter is the first one in the previous level.

Bit 11, Set TDR/CDR. This bit determines whether the immediate data byte enters into TDR or CDR.

Bit 12, Set LSAR 0 and 1. This bit causes the contents of TDR bit 0-3 to be set into LSAR 0 and TDR bits 4-7 to be set into LSAR 1, when turned on (1).

Bit 13 Left/Right. Since TDR or CDR are halfword wide, bit 13 determines whether the immediate byte enters bits 0-7 or bits 8-15 of the respective register.

Bits 14 and 15, Invert Switch Function. These bits represent four patterns that set the invert switch for a later arithmetic/logic operation, (see parameters).

Bits 16-23, Immediate Byte. These bits have binary values assigned, running from bottom to top in ascending value, capable of representing any value from 00 to FF.

Valid Parameters for Group 2 Instructions

LSAR Setting Cases:

```
          Bit 12
C1 =        0       = no LSAR setting

C2 =        1       = TDR bits 0....3 to LSAR0, and
                      TDR bits 4....7 to LSAR1
```

Invert Switch Cases:

See C5....C8 of group 1 parameters

Group 3: Arithmetic/Logic Operations to Local Storage

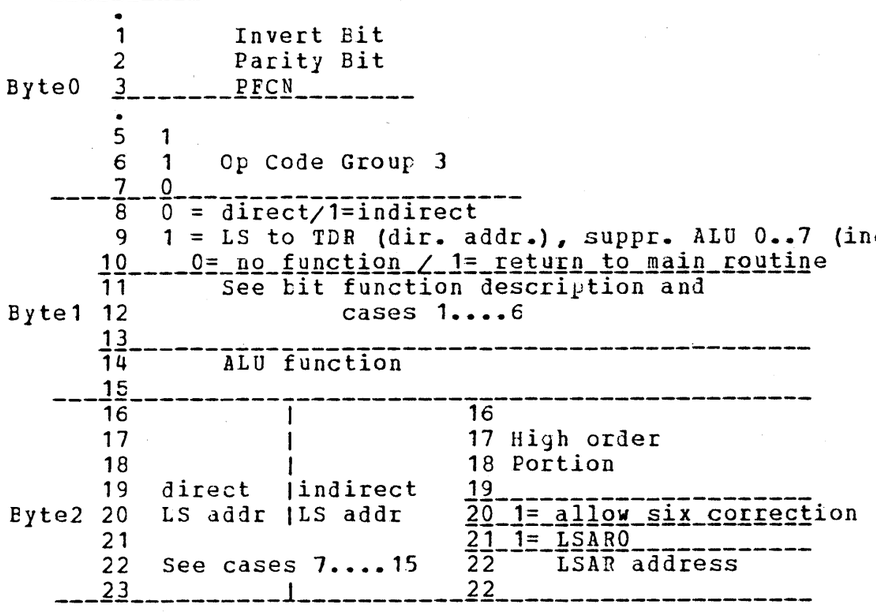
Primary Function: The contents of TDR represent an operand that is either added, ANDed, ORed, or EXCLUSIVE ORed with the contents of CDR. The result is stored into the IPU local storage. Before the ALU operation (in the same cycle) TDR and CDR may be loaded from the same LS location which receives the result from the ALU output afterwards.

Note: The contents of CDR pass through the invert switch before they enter the ALU. The result, therefore, depends on the microinstruction that sets the invert switch prior to the arithmetic/logic operation.

Secondary Functions

- the local storage can be addressed either direct or indirect.
- various ALU conditions can be saved and propagated to other ALU operations which need not be in consecutive order.
- six correction on byte basis can be performed.
- the left byte of the result can be suppressed.
- The program can be made to return from a subroutine.

Common Layout of Group 3 Instructions



Bit Function Description

Bit 1, Invert. Generated internally by hardware.

Bit 2, Parity Bit. Generated by assembler program.

Bit 3, PFCN - Parity Function Bit. Generated by assembler program.

Bits 5,6,7, Op Code. These bits represent a pattern that is unique and common to all group 3 instructions.

Bit 8, Direct/Indirect. This bit has a dual function. It determines whether the IPU local storage is addressed direct or indirect.

Direct (bit 8=0). When direct is specified, instruction bits 16....23 represent a binary number that is used as address. The lowest binary value is assigned to bit 23, the highest binary value is assigned to bit 16.

Indirect (bit 8=1). When indirect is specified, bit 21 determines which of the two indirect addressing methods are used, as follows:

Bit 21 off (0). Instruction bits 16....19 represent the high order portion of the address. The low order portion is provided by the contents of the local store address register (LSAR) that is addressed by instruction bits 22 and 23. Both address portions are said to be concatenated (chained) to form one logical bit string.

Bit 21 on (1). With bit 21 on, instruction bits 16....19 are ignored. The high order portion of the address is provided by LSAR 0. The low order portion is provided by the LSAR addressed by instruction bits 22 and 23. This may be any LSAR including LSAR 0. Both address portions are chained to form one logical bit string.

Note: Six correction can be specified only with indirect addressing.

Bit 9, 'Suppress ALU Positions 0....7' or 'Local Storage to TDR'. With indirect LS addressing this bit provides a means to suppress the left byte of the ALU (the ALU is halfword wide). This function is used when 24-bit addresses are calculated via two passes through the ALU, such as for load register type operations. With direct LS addressing bit 9 is used to indicate that the TDR has to be loaded from local storage before the ALU performs its operation.

Bit 10, Leave Subroutine. This bit allows the microprogram to return (from a subroutine) to the program level that was in effect prior to the last level switching operation The instruction that has bit 10 on is still part of the subroutine and so is the next instruction. However, the instruction that follows thereafter is the first one in the previous level.

Bit 11, Accumulate Condition Code. This bit provides the means to logically link any number of ALU operations with each other so as to obtain one result and one result condition from which a final condition code (for the PSW) can be derived. Conversely, by turning bit 11 off, any number of independent ALU operations can be interspersed in a string of linked operations. Bit 11 thus eliminates the need for processing long operands in consecutive sequence, and allows manipulation of operands or portions thereof inbetween the main string.

The following detail functions are involved:

The first ALU operation of a string must have bit 11 on and either bit 12 or bit 13 on. This bit combination sets the ALUZERO latch to the zero state. The latch remains in the zero state as long as all operations of the string (all those that have bit 11 on) including the first one produce zero results. If any operation in the string (including the first one) produces a result greater than zero, the latch changes to the "not zero" state and remains in this state until it is reset, irrespective of how many zero results may follow in the string.

Only those operations which have bit 11 on are treated as part of the string. Bit 11 also ensures that the signs of the operands and carries out of ALU position 0 and 1 are saved and propagated to the next operation that has bit 11 on. In this manner a common condition is accumulated for the string. Interspersed operations (which have bit 11 off) cannot disturb the accumulated condition because the latter is saved. The accumulation ends with the first instruction that has bit 11 and either bit 12 or 13 on because this combination deliberately set the ALUZERO latch to the zero state (reset), thus starts a new string.

Operations which have bit 11 off may also form a string because carries can be propagated, however, no common condition is accumulated (also see bit 12 and bit 13).

Bit 12, 'Force Carry' or 'Local Storage to CDR'. When on, this bit causes a carry to be generated and entered into ALU position 15 which is the low order position. Bit 12 sets the ALUZERO latch to zero if bit 11 is also on. If the mnemonic is 'LTxxx' (bits 8 and 9 = 01), bit 12 being on causes CDR to be loaded from local storage prior to the ALU operation.

Bit 13, Reset Carry Latch. This bit provides the means for controlling the carry that may emerge from ALU position zero (the high order position). When bit 13 is off (0), a carry from the preceding operation automatically enters ALU position 15 during the next ALU operation. This action is prevented if bit 13 is on (1). Bit 13 sets the ALUZERO latch if bit 11 is also on.

Bit 14 and 15, ALU Function. These bits are capable of providing four different patterns, which specify the ALU functions OR, Add, AND, Exclusive OR as follows:

Bit 14	Bit 15	
0	0	= OR
0	1	= Add
1	0	= AND
1	1	= Exclusive OR

Bit 20, Allow Six Correction. This bit provides the means to convert a hexadecimal value to a decimal value. When bit 20 is on, a binary 6 is subtracted from ALU positions 12....15 (units digit) if no carry emerged from ALU position 12. The same occurs with ALU positions 8....11 (tens digit) if no carry emerged from ALU position 8. ALU positions 0....7 do not participate in six correction.

Note: Six correction is possible only in conjunction with indirect LS addressing.

Valid Parameters of Group 3 Instructions

ALU Control Cases	Function Performed
C1 = Bit 11 Bit 12 Bit 13 0 0 0	• allow carry from ALU pos 0 that was previously saved by an instruction that had bit 11 off to enter ALU pos 15. • save carry out of ALU pos 0.
C2 = Bit 11 Bit 12 Bit 13 0 0 1	• prevent carry from entering ALU pos 15. • save carry out of ALU pos 0.
C3 = Bit 11 Bit 12 Bit 13 0 1 0 (See Note below)	• force carry into ALU pos 15. • save carry out of ALU pos 0.
C4 = Bit 11 Bit 12 Bit 13 1 0 0	• allow carry from ALU pos 0 that was previously saved by an instruction that had bit 11 on to enter ALU pos 15. • save contents of ALU pos 0 (sign). • save carry out of ALU pos 0. • save carry out of ALU pos 1. • reset ALUZERO latch if result not zero.

C5 = Bit 11 Bit 12 Bit 13
1 0 1

- set ALUZERO latch to zero prior to operation.
- prevent carry from entering ALU pos 15.
- save contents of ALU pos 0 (sign).
- save carry out of ALU pos 0.
- save carry out of ALU pos 1.
- reset ALUZERO latch if result not zero.

C6 = Bit 11 Bit 12 Bit 13
1 1 0
(See Note below)

- set ALUZERO latch to zero prior to operation.
- force carry into ALU pos 15.
- save contents of ALU pos 0 (sign).
- save carry out of ALU pos 0.
- save carry out of ALU pos 1.
- reset ALUZERO latch if result not zero.

Note: Cases 3 and 6 are not valid for LTxxx instructions (instruction bits 8 and 9 = 01). With these mnemonics instruction bit 12 is used to indicate 'local store to CDR'.

Local Store Address Cases

Case	Instruction bits 8	21	22	23	Source of local storage address high order	low order	
7	0	x	x	x	Instruction bits 16...23		direct
8	1	0	0	0		LSAR 0	
9	1	0	0	1	Instr. bits	LSAR 1	
10	1	0	1	0	16...19	LSAR 2	indirect
11	1	0	1	1		LSAR 3	
12	1	1	0	0		LSAR 0	
13	1	1	0	1	LSAR 0	LSAR 1	addressing
14	1	1	1	0		LSAR 2	
15	1	1	1	1		LSAR 3	

Note: The parameters actually used will be shown in the microprogram listings.

Group 4: Arithmetic/Logic Operations to Main Storage

Primary Function. The contents of TDR are Added, ANDed, ORed, or Exclusive ORed with the contents of CDR and the result is placed into main storage.

Secondary Functions

- the format can be specified as halfword or byte.
- the main storage address can be automatically incremented, decremented, or left as it is.
- the ALU can be controlled so as to propagate carries (or not) and to accumulate a common result condition (or not).
- the result can be subjected to six correction.
- the program can be made to return from a subroutine.
- dynamic address translation can be enabled or disabled.
- For mnemonics XMSC/XMSCR see special paragraph at the end of this group description.

Common Layout of Group 4 Instructions

```
          .
          1        Invert Bit
          2        Parity Bit
Byte0     3        PFCN
          .
          5  1
          6  1     Op Code Group 4
          7  1
          8  1 = main storage
          9  0 = write
         10  1 = leave subroutine
Byte1    11  1 = accumulate condition code
         12  1 = force carry
         13  1 = reset carry latch
         14
         15        ALU = Function
         16  1 = allow six correction
         17  1 = halfword/0=byte
Byte2    18  1 = increment
         19  1 = decrement
         20
         21        MSC LS address
         22  0 = no function
         23  1 = relocate/0=no relocate
```

Bit Function Description

Bit 1, Invert. Generated internally by hardware.

Bit 2, Parity Bit. Generated by assembler program.

Bit 3, PFCN - Parity Function Bit. Generated by assembler program.

Bits 5....9, Op Code Group 4. These bits represent a unique pattern that is common to all group 4 instructions. Bit 8 at 1 level designates the operations as pertaining to main storage, while bit 9 at 1 level specifies the direction as "to" main storage.

Bit 10, Leave Subroutine. This bit allows the IPU microprogram to re-enter the program level that was in effect prior to the last level switching operation. In this manner, the program may return from a subroutine to its original point of continuation.

The instruction which has bit 10 on is still part of the subroutine and so is the next following instruction. However, the instruction that follows thereafter is the first one of the continued previous routine.

Bit 11, Accumulate Condition Code. This bit provides the means to logically link any number of ALU operations with each other so as to obtain one result and one result condition from which a final condition code (for the PSW) can be derived. Conversely, by turning bit 11 off, any number of independent ALU operations can be interspersed in a string of linked operations. Bit 11 thus eliminates the need for processing long operands in consecutive sequence, and allows manipulation of operands or portions thereof inbetween the main string.

The following detail functions are involved:

The first ALU operation of a string must have bit 11 on and either bit 12 or bit 13 on. This bit combination sets the ALUZERO latch to the zero state. The latch remains in the zero state as long as all operations of the string (all those that have bit 11 on) including the first one produce zero results. If any operation in the string (including the first one) produces a result greater than zero, the latch changes to the "not zero" state and remains in this state until it is reset, irrespective of how many zero results may follow in the string.

Only those operations which have bit 11 on are treated as part of the string. Bit 11 also ensures that the signs of the operands and carries out of ALU position 0 and 1 are saved and propagated to the next operation that has bit 11 on. In this manner a common condition is accumulated for the string. Interspersed operations (which have bit 11 off) cannot disturb the accumulated condition because the latter is saved. The accumulation ends with the first instruction that has bit 11 and either bit 12 or 13 on because this combination deliberately set the ALUZERO latch to the zero state (reset), thus starts a new string.

Operations which have bit 11 off may also form a string because carries can be propagated, however, no common condition is accumulated (also see bit 12 and bit 13).

Bit 12, Force Carry. When turned on, this bit causes a carry to be generated and entered into ALU position 15 which is the low order position. Bit 12 sets the ALUZERO latch to zero if bit 11 is also on.

Bit 13, Reset Carry Latch. This bit provides the means for controlling the carry that may emerge from ALU position zero (the high order position). When bit 13 is off (0), a carry from the preceding operation automatically enters ALU position 15 during the next ALU operation. This action is prevented if bit 13 is on (1). Bit 13 sets the ALUZERO latch if bit 11 is also on.

Bit 14 and 15, ALU Function. These bits are capable of providing four different patterns, which specify the ALU functions OR, Add, AND, Exclusive OR as follows:

Bit 14	Bit 15	
0	0	= OR
0	1	= Add
1	0	= AND
1	1	= Exclusive OR

Bit 16, Allow Six Correction. When turned on (logical 1), this bit allows ALU bits 12....15 (units digit) and 8....11 (tens digit) to be subjected to six correction. Six correction consists of a subtraction of a binary 6 from ALU pos 12....15 provided there was no carry out of pos 12. The same occurs for ALU pos 8....11 if no carry emerged from pos 8. Since the boolean functions AND, OR, and XOR do not produce carries, a binary 6 is, in effect, subtracted unconditionally when bit 16 of the instruction is on, for these operations. ALU pos 0....7 do not participate in six correction.

Bit 17, Halfword/Byte. This bit has a dual function because it determines either of two data transmission formats. When at logical 1 level, halfword is specified which means that ALU positions 0.....15 are transferred to main storage. If at logical 0 level, byte is specified which means that ALU positions 8....15 are transferred to main storage.

Bit 18, Increment. This bit specifies how the main storage address that is used for the store operation is to be updated. The update amount corresponds to the specified format (halfword/byte) and is thus either 2 or 1.

Note 1: When operand 1 is stored with bit 18 on, the address of operand 1 is incremented and, simultaneously, the length count of operand 2 is decremented. When operand 2 is stored with bit 18 on, the address and the length count of operand 2 are decremented.

Note 2: Bit 18 must be on when a halfword or a byte is to be stored on odd boundary.

Bit 19, Decrement. This bit specifies how the main storage address that is used for the store operation is to be updated. The update amount corresponds to the specified format.

Note: When operand 1 is stored with bit 19 on, the address of operand 1 is decremented, and simultaneously, the length count of operand 2 is also decremented. When operand 2 is stored with bit 19 on, both the address and the length count of operand 2 are decremented.

Special Note for Bits 18 and 19. Bits 18 and 19 cannot both be on in the same instruction. However, both bits can be off simultaneously which means that neither an address nor a length count is changed.

Bits 20 and 21, MSC Local Store Address. These bits specify the MSC local store register the contents of which are used to address the main storage. The following registers with fixed assignment are thus addressed as follows;

Number	Bit 20	Bit 21	
Value	4	2	
Pattern	0	0	= IAR (Machine instruction address reg.)
	0	1	= Operand 1 address register
	1	0	= Operand 2 address register
	1	1	= I/O Common register

Note: The actual addresses of these registers are, in hex notation, 18,1A, 1C,1F and proper addressing is accomplished by the MSC which forces the missing bits.

Bit 22, No Function.

Bit 23, Relocate/No Relocate. This bit provides the means to access a real (physical) main storage location directly without going through the relocate (dynamic address translation) mechanism. Conversely, when on (1), the bit provides for dynamic address translation if the extended control mode bit in the current PSW is on.

Valid Parameters of Group 4 Instructions

ALU Control Cases	Function Performed
C1 = Bit 11 Bit 12 Bit 13 0 0 0	• allow carry from ALU pos 0 that was previously saved by an instruction that had bit 11 off to enter ALU pos 15. • save carry out of ALU pos 0.
C2 = Bit 11 Bit 12 Bit 13 0 0 1	• prevent carry from entering ALU pos 15. • save carry out of ALU pos 0.
C3 = Bit 11 Bit 12 Bit 13 0 1 0	• force carry into ALU pos 15. • save carry out of ALU pos 0.
C4 = Bit 11 Bit 12 Bit 13 1 0 0	• allow carry from ALU pos 0 that was previously saved by an instruction that had bit 11 on to enter ALU pos 15. • save contents of ALU pos 0 (sign). • save carry out of ALU pos 0. • save carry out of ALU pos 1. • reset ALUZERO latch if result not zero.
C5 = Bit 11 Bit 12 Bit 13 1 0 1	• set ALUZERO latch to zero prior to operation. • prevent carry from entering ALU pos 15. • save contents of ALU pos 0 (sign). • save carry out of ALU pos 0. • save carry out of ALU pos 1. • reset ALUZERO latch if result not zero.
C6 = Bit 11 Bit 12 Bit 13 1 1 0	• set ALUZERO latch to zero prior to operation. • force carry into ALU pos 15.

- save contents of ALU pos 0(sign).
- save carry out of ALU pos 0.
- save carry out of ALU pos 1.
- reset ALUZERO latch if result not zero.

Format Cases

C 7 = Bit 17=0 Byte
C 8 = Bit 17=1 Halfword

Update Cases

	Bit 18	Bit 19	
C 9 =	0	0	= no update
C 10 =	1	0	= increment
C 11 =	0	1	= decrement

XMSC = Exclusive OR to Main Storage, Read from Main Storage into CDR

This is a combination of the two mnemonics XMS (group 4) and MSC (group 5). The IPU places a main storage request. When the request is honored, the main storage performs a read and a write cycle. The IPU executes two cycles (see figure below). During the first IPU cycle the ALU performs an Exclusive OR function. The result of which will be stored into main storage during the MS write cycle. The byte fetched from main storage earlier (during the MS read cycle) is placed into CDR bits 8...15 during the second IPU cycle. CDR bits 0...7 are set to zeros.
The IPU performs during its first cycle the same functions as for an XMS instruction. For the second cycle control word bit 9 will be inverted to a logical one, which indicates main storage read. Thus the IPU performs the same operations as for the mnemonic MSC and stores the data coming from main storage into CDR (bit 11 = 0). Besides bit 9 no other bit of the control word is changed or inverted after the first IPU cycle and the Neither of the IPU cycles is interruptible. Bit 10 of the microinstruction is off.

XMSCR = Exclusive OR to Main Storage, Read from Main Storage into CDR, then Return

The basic functions are exactly the same as for XMSC (see previous paragraph). The only difference is in bit 10, which is a logical one. This causes the microprogram to return (branch) to the main routine two micro-instructions later.

	MS Read cycle	MS Write cycle
MSC	Read data to be loaded into CDR.	Write data originating from Ex OR function (first IPU cycle) into main storage.
	First IPU cycle	**Second IPU cycle**
IPU	ALU Exclusive OR, result will be written into main storage during MS Write cycle.	Data read out of main storage during MS Read cycle is loaded into CDR.

Layout of XMSC/XMSCR

		First IPU-Cycle = XMS	Second IPU Cycle = MSC
		. __ pattern fixed for XMSC(R)	
		. |	
		. V	
Byte0	3		
	5	1	
	6	1	
	7	1 Op code = group 4	Op code = group 5
	8	1	
	9	0 = write	1(inverted zero) = read
	10	0 = XMSC / 1 = XMSCR	same as first cycle
Byte1	11	0	0 = read into CDR
	12	0 no function	0
	13	0	0 no function
	14	1	1
	15	1 ALU = Exclusive Or	1 invert switch = force zeros
	16	0 = no function	
	17	0 = byte format	
	18	0 = not increment	same as
Byte2	19	0 = not decrement	
	20	1 MSC LS address =	
	21	0 Op. 2 addr.-register	first cycle
	22	1 = XMSC(R)	
	23	1 = relocate	

Group 5: Read from Main Storage into TDR or CDR

Primary Function. A halfword or a byte is fetched from main storage and placed into either TDR or CDR. If a byte is fetched, this byte is automatically set into bits 8....15 of the selected register (right adjusted) and bits 0....7 of the selected register are set to zero.

Secondary Function

- The contents of TDR may be propagated into LSARs 0 and 1; or 2 and 3; or 0,1,2 and 3.
- The length count or the main storage address may be updated.
- A test on exceptional conditions may be performed so that a branch to a specific address can be initiated upon finding exceptional conditions.
- Dynamic address translation can be employed or circumvented.
- The invert switch can be set to true, invert, force ones or force zeros.
- A return from a subroutine can be initiated.
- Dynamic address translation can be enabled or disabled.

Common Layout of Group 5 Instructions

```
          .
          1        Invert Bit
          2        Parity Bit
Byte0     3        PFCN
          .
          5  1
          6  1                      Op Code Group 5
          7  1
          8  1 = Main Storage
          9  1 = Read
         10  1 = Leave subroutine
Byte1    11  1 = TDR/0=CDR
         12  1 = set LSAR 0 and 1
         13  1 = set LSAR 2 and 3
         14
         15     invert switch function
         16  1 = test exception
         17  1 = halfword/0=byte
         18  1 = increment
         19  1 = decrement
Byte2    20
         21     MSC LS address
         22  1 = Length count/0=address
         23  1 = relocate/0=no relocate
```

Bit Function Description

Bit 1, Invert. Generated internally by hardware.

Bit 2, Parity Bit. Generated by assembler program.

Bit 3, PFCN - Parity Function Bit. Generated by assembler program.

Bits 5 through 9, Op Code. These bits represent a pattern that is unique and common to all group 5 instructions.

Bit 10, Leave Subroutine. This bit provides the means to branch back from a subroutine to the program level that was in effect prior to the last level switching operation. The instruction that has bit 10 on is still part of the subroutine and so is the next instruction. However, the instruction that follows thereafter is the first one in the previous level.

Bit 11, TDR/CDR. This bit determines the register into which the fetched data is placed. Logical 1 level specifies TDR, logical 0 level specifies CDR.

Bit 12, Set LSAR 0 and 1. When this bit is on, TDR bits 0....3 are propagated to LSAR0 and TDR bits 4....7 are propagated to LSAR1.

Timing Note. If main storage data enters TDR the propagated data is not valid in the LSARs until 450 nano sec (1 IPU cycle) after the read from main storage has ended. This means that the microinstruction that follows immediately after the read from main storage cannot use the LSAR contents. However, the propagated data is available to the instruction thereafter (the second after the read from main storage).

This timing restriction does not apply if main storage data is placed into CDR because then the TDR contains valid data from a previous operation.

Bit 13, Set LSAR 2 and 3. When on (1), TDR bits 8....11 are propagated to LSAR 2 and TDR bits 12.....15 are propagated to LSAR 3. The same timing restriction as specified under "Bit 12" applies when main storage data is fetched in TDR and this data is propagated.

Bits 14 and 15, Invert Switch Function. These bits determine the function of the invert switch, as follows:

Bit 14	Bit 15	
0	0	= invert
0	1	= true
1	0	= force ones
1	1	= force zeros

Bit 16, Test Exception. This bit provides the means for checking on interrupts, address stops, and similar exceptional conditions. When bit 16 is on (1), data is fetched from main storage and simultaneously exceptional conditions are checked. If exceptional conditions are found, the microprogram branches to a fixed address where the exceptional condition handling routine begins. If exceptional conditions are not found, the microprogram proceeds with the next sequential microinstruction.

Bits 17, Halfword/Byte. This bit provides the means to determine the format of the data to be fetched. If either form of updating (increment or decrement) is also specified in the microinstruction, bit 17 implicitly determines the updating amount as either 2 or 1 as required for the selected format.

Note: If the main storage read operation uses the contents of the machine instruction IAR as main storage address, the format must be specified as "halfword", because the smallest machine instruction (e.g. RR format) has halfword size.

Bit 18, Increment. This bit determines the updating modus as plus 2 or plus 1 as required for the selected format. Whether the updating pertains to the main storage address or to the length count depends on bit 22 which specifies either length count or address.

Note 1: If "length count" is specified, bit 18 must be off because the length count can be decremented only.

Note 2: If a byte or a halfword is to be fetched from odd boundary, bit 18 must be on. If either decrement or no update is specified for an odd address, data is fetched from the even boundary below the odd address.

Bit 19, Decrement. This bit determines the updating modus as minus 2 or minus 1 as required by the selected format. Whether the updating pertains to the main storage address or to the length count depends on bit 22 which specifies either length count or address. Either facility may be specified for decrement.

Special Note for Bits 18 and 19. Bits 18 and 19 cannot both be on in the same instruction. However both bits may be off simultaneously which means "no change" to length counts or addresses.

Bits 20 and 21, MSC Local Store Address. These bits are used to address four MSC LS registerswith fixed assignments as follows:

Bit 20	Bit 21	
0	0	= IAR (machine instruction address register)
0	1	= Operand 1 address register
1	0	= Operand 2 address register
1	1	= I/O Common register

Bit 22, Length Count/Address. This bit provides the means to specify either the length count or the address as the facility to be updated by the main storage controller. The updated value is available when the main storage operation has ended (i.e. for the next operation).

Bit 23, Relocate/No Relocate. This bit provides the means to read from a fixed or known main storage location directly without going through the relocate mechanism. Conversely, the relocation mechanism can be employed, provided the extended control mode bit is on in the current PSW.

Group 6: Read from MSC Local Storage or Key Store

Primary Function: The contents of either the right or left portion of an MSC local store register or the contents of a key storage position are fetched and placed into either the TDR or CDR.

Secondary Functions

- the contents of TDR can be propagated into LSAR 0 and 1; or 2 and 3; or 0,1,2 and 3.
- the invert switch can be set to true, invert, force ones, force zeros.
- the MSC can be addresssed directly or indirectly.
- with MSCTL(R) control information as tc page sizes and storage limits can be transferred to the main storage controller. (For details see under 'Bit function description' for bit 17.)

Common Layout of Group 6 Instructions

```
          .
          1        Invert Bit
          2        Parity Bit
Byte0     3 _______PFCN________
          .
          5  1
          6  1      Op Code Group 6
        ____7  1
          8  0 = MSC LS
          9  1 = read___________
         10  1 = leave subroutine
Byte1    11  1 = TDR/0=CDR_____
         12  1 = set LSAR 0 and 1
         13  1 = set LSAR 2 and 3
         14
        ___15_____invert switch function
         16  0 = MSC LS/1=key store
         17  1 = immediate (MSCTL/MSCTLR)
Byte2    18  0 = direct/1=indirect_____________
          19       direct     | 19_______ignored_
          20       MSC LS     | 20___0 = TDR select
          21       address    | 21_______ignored_
          22                  | 22
        ___23  1 = left/0=right| 23__LSAR address___
```

Bit Function Description

Bit 1, Invert. Generated internally by hardware.

Bit 2, Parity Bit. Generated by assembler program.

Bit 3, PFCN - Parity Function Bit. Generated by assembler program.

Bits 5....9, Op code. These bits represent a pattern that is unique and common to all group 6 instructions. Bit 8 at zero specifies the source as being the Local Storage of the main storage controller.

Bit 10, Leave Subroutine. This bit provides the means to branch back from a subroutine to the program level that was in effect prior to the last level switching operation. The instruction that has bit 10 on is still part of the subroutine and so is the next instruction. However, the instruction that follows thereafter is the first one in the previous level.

Bit 11, TDR/CDR. This bit determines the register into which the fetched data is placed. Logical 1 level specifies TDR, logical 0 level specifies CDR.

Bit 12, Set LSAR 0 and 1. When on, this bit causes TDR bits 0...3 to be propagated to LSAR0 and TDR bit 4....7 to be propagated to LSAR 1.

Bit 13, Set LSAR 2 and 3. This bit causes TDR bits 8....11 to be propagated to LSAR2 and TDR bits 12....15 to be propagated to LSAR 3.

Note to bits 12 and 13. When reading from MSC Local Storage or Key Storage, there is no timing restriction associated with LSAR propagation. This means that the propagated data is available at the end of the MSC LS or Key Store read operation.

Bits 14 and 15, Invert Switch Function. These bits determine the function of the invert switch, as follows:

Bit 14	Bit 15	
0	0	= invert
0	1	= true
1	0	= force ones
1	1	= force zeros

Bit 16, MSC Local Store/Key Store. This bit specifies the facility from which data is to be fetched. Bit 16 at zero specifies MSC Local Store, bit 16 at 1 specifies key store.

Note: If either key store or left portion of an MSC register is specified, seven data bits are fetched. These seven data bits are always placed right adjusted into TDR or CDR (whichever is applicable). The remainder of the receiving register (bits 0....8) is set to zero.

Bit 17, Immediate. (MSCTL/MSCTLR). When this bit is on, the entire character of the read instruction is changed. The primary function is then the transmission of control information to latches and special registers in the MSC and/or Relocation unit whereas the reading of data from MSC local store registers or key storage becomes the secondary function. Actually, a true operation takes place because coded control information is sent to the MSC and/or Relocation unit and simultaneously the same coding addresses the MSC local storage or key storage and reads from it.

The control information is taken from the ALU-output, which depends on:

a) the data loaded into CDR and TDR,
b) the last ALU operation called for prior to the MSCTL instruction, and
c) the invert switch function specified by the MSCTL instruction.

For more details, see description of bit 18.

Bit 18, Direct/Indirect. This bit specifies the source that is to supply the control and/or MSC local store address, as follows:

Bit 18=0 (Direct). Instruction bits 19....23 represent the source. Whether this source represents an MSC LS address alone or an MSC LS address and, at the same time, control immediate information depends on the state of bit 17. If bit 17 is off (0) bits 19....22 represent the address and bit 23 specifies left or right. Bits 19 to 22 have the binary values 8,4,2,1 assigned and are thus capable of addressing the upper half of the MSC Local Storage because a bit with value 16 is forced by the MSC itself. Consequently, MSC LS registers ranging from address 10 to 1F can be addressed.

If bit 17 is on (1), the meaning of bits 19.....23 is a control code.

The control code function of bits 19....23 is as follows:

Bit Numbers						
Forced	19	20	21	22	23	
1	1	1	0	0	0	= increment relocation counter by 1
1	1	0	0	1	0	= write all, meaning all of the sixteen associative registers
1	1	0	0	0	0	= set relocation mode, meaning details such as page size, and relocation yes/no which are taken from PSW and control registers
1	1	0	1	1	0	= set relocation counter to hex 7

Bit 18=1 (Indirect). When indirect is specified, bits 19....23 are ignored as address or control code. Instead, two choices exist as to the source, as specified by bit 20 (TDR select):

Bit 20=1. When bit 20 is at 1 level, LSAR0 chained with one of the four LSAR's furnish the address/control code.

Bit 20=0 (TDR Select). When bit 20 is at 0 level, six specific TDR bits provide the address/control code.

The indirect method of specifying the control code (bit 17=1) allows for the following control operations via the following source bits:

either LSAR0 bits 2 and 3 + Selected LSAR bits 0,1,2,3

or TDR bit 6 and 7 + TDR bits 0,1,2,3

Bit Number

LSAR0 LSAR0,1,2,or3
2 3 0 1 2 3

TDR TDR
6 7 0 1 2 3

0 0 x x x 1 = set main storage size, the main storage size information stored by a previous IPU operation into the MSC local storage will be transferred to latches in the MSC

0 0 x x x 0 = write associative array, the ALU output is transferred to an AA register addressed by the relocation counter

0 1 x x x 1 = write real address local store, the ALU output is transferred to a relocation local storage register addressed by the relocation counter

1 1 1 0 0 0 = increment relocation counter by 1

1 1 0 0 1 0 = write all, meaning all of the sixteen associative registers. Write all is used for purging (invalidating) the TLB

1 1 0 0 0 0 = set relocation mode, meaning page size, segment protection, and relocation yes/no, details that are fetched from the PSW and the control registers

1 1 0 1 1 0 = set relocation counter to hex 7

Note: bits identified by XXX are ignored

Immediate Control Operation Details

Set Main Storage Size. The three bits denoted as XXX represent the values 4, 2, 1, and are thus capable of specifying eight different MSC local storage registers. However, as an engineering convention, MSC LS-Reg 0 is always the one that has been loaded with the main storage size. The XXX bits will, therefore, be zero. Either the SVP or the 2311 or 2314 emulator supplies the main storage size.

Write Associative Array. The three bits denoted as xxx in the pattern represent the values 4,2,1 and are thus capable of addressing any of the eight registers in the associative array. The selected register is loaded with the ALU output which is interpreted as follows:

```
ALU bits --->      | 0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|12|13|14|15|
Logic Address bits|16|17|18|19|20|  | x| x| 8| 9|10|11|12|13|14|15|
                                     ^
                                     |
                                     '--- 0 = validate
                                          1 = invalidate
```

Note: x is ignored.

Write Real Address Local Store. The contents of the relocation address counter are used to address a specific register. The selected register is loaded with the ALU output which is interpreted as follows:

```
ALU bits --->      | 0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|12|13|14|15|
Real Address bits |16|17|18|19|20|x |x |x | 8| 9|10|11|12|13|14|15|
```

Note: x is ignored.

Write ALL. Each of the sixteen registers in the associative array is loaded with the ALU output and will thus contain identical data. The ALU output is interpreted as follows:

```
ALU bits --->      | 0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|12|13|14|15|
                   | n| n| n| n| n|  | x| x| n| n| n| n| n| n| n| n|
                                     ^
                                     |
                                     '--- 0 = validate
                                          1 = invalidate
```

Note: x is ignored
n is any value

Set Relocation Mode. The ALU output is transferred to latches in the MSC which uses the information to control the access operations accordingly. The ALU output is interpreted as follows:

```
ALU bits ---->   0 1 2 3  4  5  6 7 8 9 10 11 12 13 14 15

PSW and Control  x x x x 12  5 24 9 x x  x  x  x  x  x  x
Reg bits
```

ALU bit 4 represents PSW bit 12 which specifies:

EC Mode when 0
BC Mode when 1

ALU bit 5 represents PSW bit 5 which specifies:

Dynamic Address Translation when 1
System/360 Addressing Mode when 0

ALU bit 7 represents bit 9 of control register 0 which specifies:

2K Page Size when 1
4K Page Size when 0

Invalidate/Validate Matching Entry. One of the sixteen registers in the associative array which matches ALU output 0...4, 8...15 is loaded with the ALU output 0...5, 8...15. ALU bit 5 invalidates the entry if it is at logical 1 level.

```
ALU bits --->       0  1  2  3  4  5 6 7 8 9 10 11 12 13 14 15

Logic Addr.bit -->16 17 18 19 20    x x 8 9 10 11 12 13 14 15
                                    ^
                                    |
                                    '--- 0 = validate
                                         1 = invalidate
```

Group 7: Arithmetic/Logic Operations to MSC Local Storage or Key Storage

Primary Function. The contents of TDR are added, ANDed, ORed or Exclusive ORed with the contents of CDR and the result is placed into either the local storage or key storage of the Main Storage Controller.

Secondary Functions

- TDR bits 0....3 can be suppressed (set to zero) or not.
- The ALU can be controlled so as to force a carry into the low order position or to prevent a carry from a previous operation from entering the low order position.
- Addresses alone or addresses plus storage key can be placed into an MSC LS register.
- Results that are to be stored into MSC Local store or key store can simultaneously be placed into IPU local storage.

Common Layout of Group 7 Instructions

```
          ----------
               .
               1        Invert Bit
               2        Parity Bit
Byte0          3_______PFCN________
               .
               5  1
               6  1      Group 7 Op Code
          _____7  1
               8  0 = MSC LS
               9  0 = write_________________
              10  1 = Leave subroutine
Byte1         11  1 = suppress TDR 0....3
              12  1 = force carry
              13  1 = reset carry latch____
              14
          ____15_______ALU function_________
              16  0 = MSC LS/1=key store
              17  1 = alter key
Byte2         18  0 = direct/1=indirect__
              19  1 = IPU LS store  | 19   ignored______
              20                    | 20  0 = TDR select
              21       direct       | 21   ignored_____
              22     MSC LS address | 22   LSAR
          ____23  1 = left/0=right  | 23  address______
```

Bit Function Description

Bit 1, Invert. Generated internally by hardware.

Bit 2, Parity Bit. Generated by assembler program.

Bit 3, PFCN - Parity Function Bit. Generated by assembler program.

Bits 5 through 9, Op Code Group 7. These bits represent a pattern that is unique and common to all group 7 instructions.

Bit 10, Leave Subroutine. This bit allows the microprogram to return from a subroutine to the level that was in effect prior to the last level switching operation. The instruction that has bit 10 on is still part of the subroutine and so is the next instruction. Only the instruction thereafter is the first one in the previous level.

Bit 11, Suppress TDR Bits 0...3. This bit provides the means to set the leftmost four bits of TDR to zero. This facility is used to delete the register address field (B1 or B2) of an operand address so as to retain the 12-bit displacement (D1 or D2). This allows the displacement (alone) to be added to the contents of a base register (general purpose register).

Bit 12, Force Carry. This bit allows a carry to be generated and inserted into ALU position 15 (the low order position).

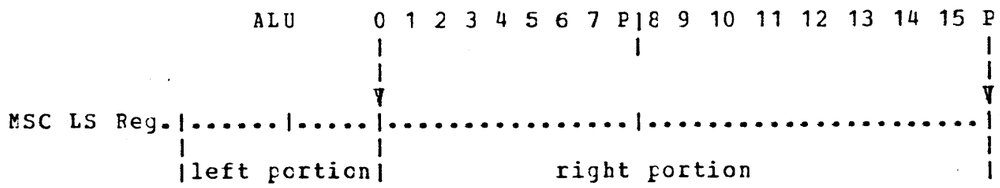
Bit 13, Reset Carry Latch. This bit resets the carry latch to the "no carry" state so as to prevent a carry from a previous ALU operation from entering ALU position 15. An operation that has bit 13 on is, in effect, always a stand-alone operation.

Bits 14 and 15, ALU Function. These bits represent a pattern capable of specifying four different arithmetic/logic operations, as follows:

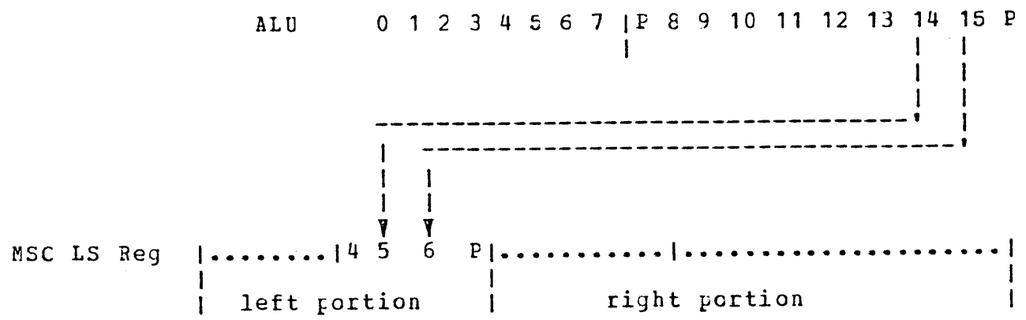
Bit 14	Bit 15	
0	0	= OR
0	1	= Add
1	0	= AND
1	1	= Exlusive OR

Bit 16, MSC Local Storage/Key Storage. This bit specifies the destination of the ALU result. When bit 16 is 0, the result is stored into either the left or right portion of an MSC LS register, as specified, by bit 23 (left/right). If bit 16 is 1, the ALU result is stored into key storage. The ALU result is a halfword (bits 0....15) and so is the right portion of every MSC LS register. However, the left portion of an MSC LS register as well as a key storage register are only 7 bits wide. Therefore, the following description explains which bits are stored in which positions depending on which destination has been specified.

Bit 16=0, Bit 23=0, means store operation into right portion of MSC LS register, as follows:

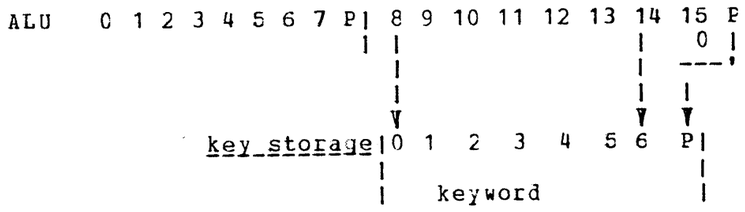


Bit 16=0, Bit 23=1, means store operation into left portion of MSC LS register. Actually, only two bits are stored because the physical size of the address is 18 bits. During the store operation, the MSC checks the binary value of the address and if the insertion of the two bits increases the value to greater than 256K, the address check bit (bit 4 in the left portion) is turned on. The parity in the left portion is generated by the MSC.



Bit 16=1, means store operation to key storage. In this case ALU bits 8....14 are stored, as shown. ALU bits 8...11 represent the key, while the rest of the bits have the following functions:

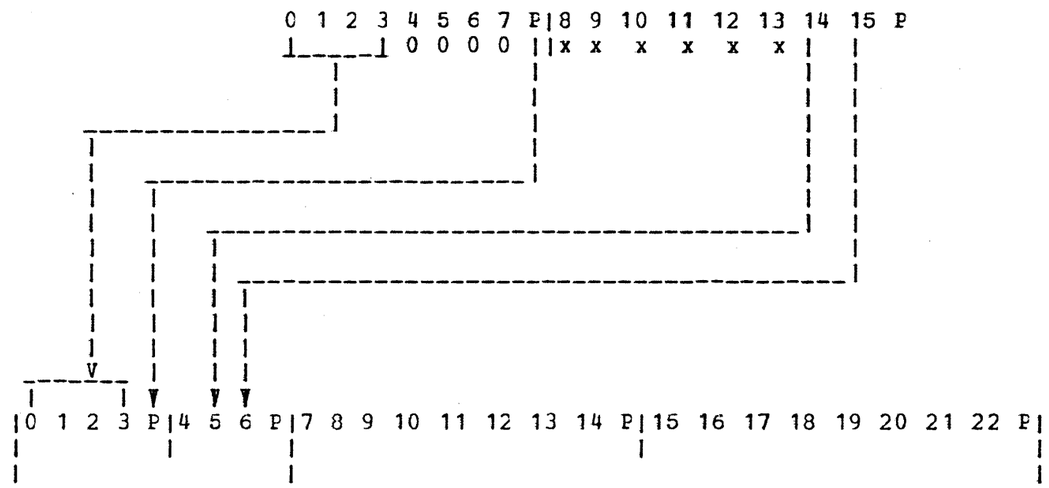
ALU bit 12 = reference bit
ALU bit 13 = change bit
ALU bit 14 = protection bit



Note: ALU bit 15 must be zero to ensure correct key parity.

Bit 17, Alter Key. This bit provides the means to alter the key in the selected MSC local storage register in addition to storing a new address.

However, when altering the key, the address storing that occurs concurrently must go into the left portion of the MSC LS register, i.e., bit 23 must be on when bit 17 is on. The following ALU bits are stored:



Note: ALU bits 4....7 must be zero to ensure correct parity for the key. ALU bits 8....13 are ignored.

Bit 18, Direct/Indirect. This bit determines the method of addressing the MSC local store and, in case of the special operation, the method of addressing MSC local storage, logic address storage, or real address storage.

If bit 18 is 0 (Direct). If direct addressing is specified, two address bits are forced by the MSC while four bits are provided by the instruction, as follows:

Origin: Forced, Forced, 20, 21, 22, 23
Value: 16, 8, 4, 2, 1, left/right

If bit 18 is 1 (Indirect). If indirect addressing is specified, two indirect addressing sources are available as specified by bit 20 which then has the logical meaning "TDR Select".

Bit 20=1 (not TDR select). The addressing source is composed of two bits out of LSAR0 and four bits out of the LSAR selected by instruction bits 22 and 23, as follows:

	LSAR0	+	Selected LSAR
Origin:	2,3,		0,1,2,3
Value:	16,8,		4,2,1,left/right

Bit 20=0 (TDR select). The addressing source is composed of specific bits in the TDR exclusively, as follows:

Origin: 6, 7, 0, 1, 2, 3 (TDR bits)
Value: 16, 8, 4, 2, 1, left/right

Bit 19, IPU LS Store. This bit is available only when direct addressing (bit 18=0) is specified. When bit 19 is on, ALU bits 0....15 are placed into IPU local storage and this occurs concurrently with the MSC local store or key storage transfer operation, The MSC LS address bits will then address also the IPU local storage, as follows:

Bit 22	Bit 21	Bit 20		IPU LS Register No (decimal)
1	1	1	=	71
1	1	0	=	70
1	0	1	=	69
1	0	0	=	68
0	1	1	=	67
0	1	0	=	66
0	0	1	=	65
0	0	0	=	64

Note: The IPU forces the missing address bits.

Bit 23, Left/Right. This bit specifies the portion of the selected MSC LS register into which data is stored. When indirect addressing is specified, the low order bit of the chained LSAR provides the same function.

Note 1: When "alter key" is specified, the portion must be specified as "left" to ensure correct key placement.

Transfer of Main Storage Size (for information only). The main storage size The main storage size is set by a store operation into right portion of MSC local store register. The contents of this register must be set into hardware latches by a control instruciton to become effective. The following table shows how the various main storage sizes reside in the right portion of an MSC local store register:

MSC LS Register

Bit Number	Meaning	
7	256K (when 1)	
8	128K	
9	64K	
10	32K	
11	8K	
12	4K	
13	-)	
14	-)	
15	-)	
16	-)	
17	-)	ignored
18	-)	
19	-)	
20	-)	
21	Allow Disk	
22	Allow IPU	

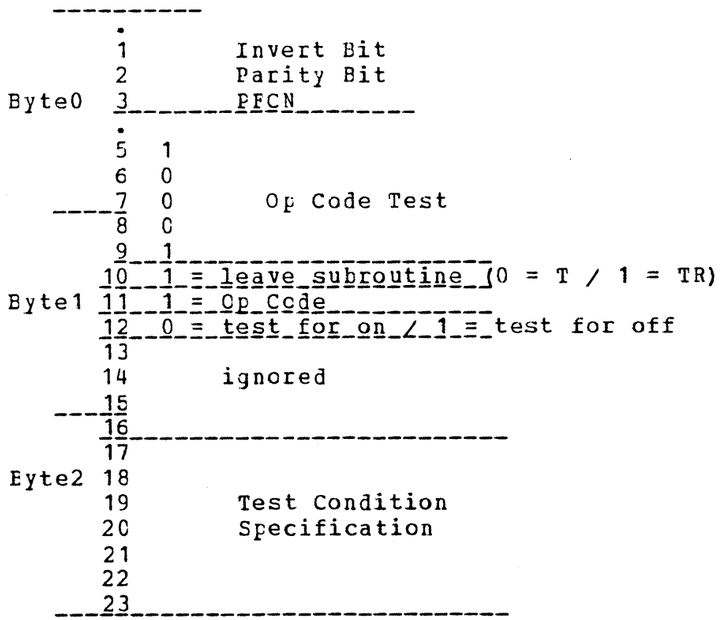
Note: "Allow Disk" is a bit that permits the Disk IOP to access the 2311 or 2314 buffer locations when the emulator is active. "Allow IPU" is a bit that permits the IPU to access the 2311 or 2314 emulator buffers. No other facility has access to the emulator buffers. Notice that up to eight individual 4K or 8K buffers are created at the upper end of main storage by setting a size smaller than the physical main storage size.

Group 8: Test Instructions

Primary Function: A test is performed on the presence or absence of a specific condition. If the test finds the specified condition, the 'Test-Fl' does not change its status. If the specified condition is not found, the 'Test-Fl' will be reset. The status of the 'Test-Fl' can be tested by a subsequent 'Branch on test'-instruction (Group 9: BT, BTS, BTM). The microprogram will branch only if the 'Test-Fl' is found on. The 'Branch on test'-instruction causes the 'Test-Fl' to be set. Therefore a second 'Branch on test'-instruction will be successful in any case.
Since the Test instruction can never set the 'Test-Fl', it is possible to AND several conditions by issuing several Test instructions in sequence before testing the status of of the 'Test-Fl'. If either one of the tested conditions does not exist, the 'Test-Fl' will be reset by the Test instruction and the subsequent 'Branch on test'-instruction is not successful.

Secondary Functions: None.

Layout of Test Instruction



Bit Function Description

Bit 1, Invert. Generated internally by hardware.

Bit 2, Parity Bit. Generated by assembler program.

Bit 3, PFCN - Parity Function Bit. Generated by assembler program.

Bits 5....9 and 11, Op Code. These bits represent a pattern that is unique for the test instruction.

Bit 10, Leave Subroutine. This bit provides the means to return from a subroutine. The instruction which has bit 10 on is still part of the subroutine and so is the next instruction. However, the instruction thereafter is the first one in the previous level.

Bit 12, Test for On/off. This bit provides the means to check for either the presence (on) or absence (off) of the specified condition (see table below).

Bits 17....23, Test Condition Specification. These bits can specify in binary notation 128 different conditions. The conditions listed below are presented as questions. It should, however, be noted that each such question can be stated in true or false form by setting Bit 12 (on/off) accordingly. The presently assigned test conditions are as follows:

Table of Test Condition Specifications

Instruction bits 17....23	12	Test Condition
000 0000	0	Any I/O Interrupt
	1	(Not) Any I/O Interrupt
000 0001	0	External Machine Check
	1	(Not) External Machine Check
000 0010	0	(Not) Previous Error
	1	Previous Error
000 0011	0	(Not) External Damage
	1	External Damage
000 0100	0	(Not) SVP Hardware Error
	1	SVP Hardware Error
000 0101	0	(Not) IOP Error
	1	IOP Error
000 0110	0	Program Interrupt
	1	(Not) Program Interrupt
000 0111	0	SVP Interrupt
	1	(Not) SVP Interrupt
000 10xx	0	IOP Response
	1	(Not) IOP Response
000 11xx	0	(Not) FP Overflow
	1	FP Overflow
001 0000	0	IOP Busy
	1	(Not) IOP Busy
001 0001	0	(Not) Page Carry
	1	Page Carry

Table of Test Condition Specifications (continued)

Instruction bits 17....23	12	Test Condition
001 0010	0	(Not) MS Address Stop
	1	MS Address Stop
001 0100	0	IOP not operational
	1	(Not) IOP not operational
001 10xx	0	(Not) TOD Security Switch
	1	TOD Security Switch
001 11xx	0	(Not) SVP Response
	1	SVP Response
010 0000	0	(Not) TDR bit 15
	1	TDR bit 15
010 0001	0	(Not) TDR bit 7
	1	TDR bit 7
010 0010	0	(Not) TDR bit 11
	1	TDR bit 11
010 0011	0	(Not) TDR bit 3
	1	TDR bit 3
010 0100	0	(Not) TDR bit 13
	1	TDR bit 13
010 0101	0	(Not) TDR bit 5
	1	TDR bit 5
010 0110	0	(Not) TDR bit 9
	1	TDR bit 9
010 0111	0	(Not) TDR bit 1
	1	TDR bit 1
010 1000	0	(Not) TDR bit 14
	1	TDR bit 14
010 1001	0	(Not) TDR bit 6
	1	TDR bit 6
010 1010	0	(Not) TDR bit 10
	1	TDR bit 10
010 1011	0	(Not) TDR bit 2
	1	TDR bit 2
010 1100	0	(Not) TDR bit 12
	1	TDR bit 12
010 1101	0	(Not) TDR bit 4
	1	TDR bit 4
010 1110	0	(Not) TDR bit 8
	1	TDR bit 8
010 1111	0	(Not) TDR bit 0
	1	TDR bit 0
011 xxxx	0	Exceptional Condit. 2
	1	(Not) Exceptional Condition 2
100 0000	0	Not LSAR 2 bit 3 and not LSAR 2 bit 0
	1	(Not) Not LSAR 2 bit 3 and not LSAR 2 bit 0
100 xx01	0	(Not) ALU zero accumulate
	1	ALU zero accumulate
100 0010	0	Not LSAR 3 bit 3 and not LSAR 3 bit 0
	1	(Not) Not LSAR 3 bit 3 and not LSAR 3 bit 0
100 xx11	0	(Not) ALU carry acc.
	1	ALU carry acc.

Table of Test Condition Specifications (continued)

Instruction bits 17....23	12	Test Condition
100 0100	0	(Not) Test Cond. Code, (No CC match)
	1	Test Condition Code, (CC match)
100 0110	0	(Not) ALU bit 0 accumulate
	1	ALU bit 0 accumulate
100 1000	0	(Not) LSAR 2 bit 3
	1	LSAR 2 bit 3
100 1010	0	(Not) LSAR 3 bit 3
	1	LSAR 3 bit 3
100 1100	0	(Not) LSAR 2 = LSAR 3
	1	LSAR 2 = LSAR 3
100 1110	0	(Not) C 8
	1	C 8
101 0000	0	(Not) Execute
	1	Execute
101 0001	0	(Not) Stop key
	1	Stop key
101 0010	0	(Not) Macro Instruction Step
	1	Macro Instruction Step
101 0011	0	(Not) Macro Address Stop
	1	Macro Address Stop
101 0100	0	(Not) IAR Boundary Check
	1	IAR Boundary Check
101 0101	0	Relocation Exception
	1	(Not) Relocation Exception
101 0110	0	(Not) MSC Check bit 2
	1	MSC Check bit 2
101 0111	0	(Not) MSC Check bit 1
	1	MSC Check bit 1
101 10xx	0	(Not) Program Event Recording
	1	Program Event Recording
101 11xx	0	(Not) Length Count Carry
	1	LC Carry
110 0100	0	(Not) Indicator bit 1
	1	Indicator bit 1
110 0101	0	(Not) Indicator bit 2
	1	Indicator bit 2
110 0110	0	(Not) Indicator bit 3
	1	Indicator bit 3
110 0111	0	(Not) Indicator bit 4
	1	Indicator bit 4
110 1000	0	Dec.Data CDR
	1	(Not) Dec. Data CDR
110 1001	0	TDR (14 and 15 = 0)
	1	(Not) TDR (14 and 15 = 0)
110 1010	0	Exceptional Condition 1
	1	(Not) Exceptional Condition 1
110 11xx	0	(Not) ALU Carry Latch
	1	ALU Carry Latch

Group 9: Branch on Test Instructions (Branch Type 1)

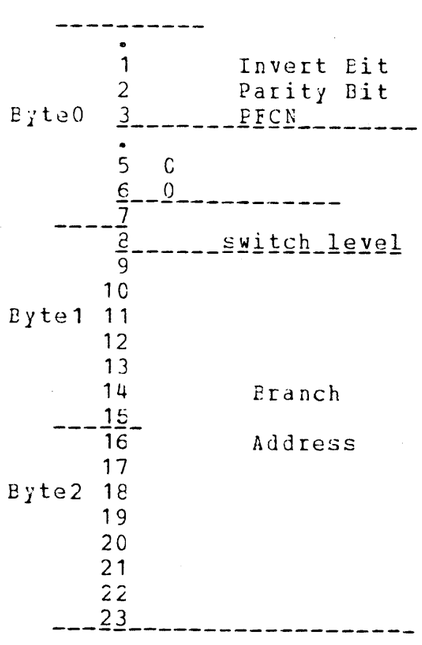
Primary Function. A check is performed on the state of the 'Test-Fl' that was reset or left unchanged by a previous test instruction (group 8). If the latch is found to be on, the next sequential instruction address located in the current IAR is ignored and, instead, the branch address contained in the "Branch on Test" instruction is used to read out the next microinstruction (branch).

If the latch is found to be off, the contents of the current IAR are used to read out the next microinstruction (no branch) and, in addition, the latch is turned on. The turned on latch provides the means to convert an unsuccessful branch to an unconditonal branch upon repetition.

Secondary Function

- Level switching can be specified or not. If specified, level switching occurs only if the branch is successful.

Common Layout of Branch on Test Instructions



Bit Function Description

Bit 1, Invert. Generated internally by hardware.

Bit 2, Parity Bit. Generated by assembler program.

Bit 3, PFCN - Parity Function Bit. Generated by assembler program.

Bits 7 and 8, Switch Level. These bits specify either of three functions as follows:

Bit 7	Bit 8	
0	0	= no level switching
0	1	= switch to subroutine
1	0	= switch to main routine

The level switching operations are defined as follows:

No Level Switching. The IPU microprogram can run in either one of three levels. Consequently, three types of microinstruction address registers are maintained as follows:

IAR 0 = main level
IAR 1 = subrcutine level
IAR 2 = trap routine level

The microprogram runs in a given level when the IAR of that level addresses the control store and subsequently receives the updated instruction address which then, again, addresses the control store. The IAR in charge is termed the "current IAR".

If no level switching is specified, and the branch is successful, then the contents of the current IAR are ignored and the branch address in the instruction reads out the next microinstruction . However, the branch address is updatedand placed into the current IAR. The program thus remains in the same level.

Switch to Subrcutine. The specification "switch to subroutine" is effective cnly if the branch instruction itself is located in the main routine. If the branch instruction is successful and switch to subroutine is specified, the current IAR (this would be IAR 0) is ignored and the branch address in the instruction reads out the next microinstruction . The address of the branch instruction itself is updated and returned to IAR 0 (main routine). However, the branch address is updated and placed into IAR 1 (subroutine). As of this moment, control has passed from IAR 0 to IAR 1 and IAR 1 is in charge of addressing the control storage.

Note: IAR 0 contains the address of the instruction that would have followed if the branch had not been successful. Any instruction in the subroutine level that has the "leave subroutine" bit (bit 10) on will return control to IAR 0, hence, continue in the main routine.

Switch to Main Routine. This specification is effective in any branch instruction that is succesful and not located in the main routine. Thus a successful branch instruction in the sub or trap level has the effect that the current IAR (IAR 1 or 2) is ignored and the branch address is used to fetch the next instruction. The address of the branch instruction itself is not updated, however the branch address is updated and placed into IAR 0 (main routine).

For Information. A branch instruction cannot go into the trap level. The trap level is forced by hardware events only. However, any instruction in

the trap level that has the "leave subroutine" bit (bit 10) on, returns control to the IAR that was in charge at the time when the trap occurred.

Special Note: If trapping occurred prior to the completion of a micro-instruction, this instruction is repeated upon returning from the trap routine.

Group 10: Conditional Branch Instructions

Primary Functions. A condition which is specified in the instruction is tested. If the specified condition is found, the address provided in the instruction is used to fetch the next microinstruction. If the specified condition is not found, the program continues with the instruction specified in the current IAR.

Secondary Functions

- Level switching is normally not performed, except where specifically stated.

Common Layout of Group 10 Instructions

```
   ----------
        .
        1        Invert Bit
        2        Parity Bit
Byte0   3        PFCN
        .
        5   0    0
        6   0 or 1    Op code
   -----7   1    |x
        8   1____|x    x = condition specification
        9   x____x
       10
Byte1  11
       12
       13
       14        Branch
   ----15
       16        Address
       17
Byte2  18
       19
       20
       21
       22
       23
   ----------------------
```

Bit Function Description

Bit 1, Invert. Generated internally by hardware.
Bit 2, Parity Bit. Generated by assembler program.
Bit 3, PFCN - Parity Function Bit. Generated by assembler program.

Bits 7...9, Condition Specification. These bits are used to specify eight different branch conditions. However, by using an unused pattern (bits 5,6,7, and 8 = 0011) of the Type I branch instructions (group 9, where bit 6 is a logical zero), the total number of branch conditions is raised to 10. The assignment is as follows:

Bit 6	Bit 7	Bit 8	Bit 9	Action and Condition
1	0	0	0	= Branch and switch to subroutine level if LSAR0 not zero.
1	0	1	0	= Branch and switch to subroutine level if LSAR0 is zero.
1	1	0	0	= Branch and switch to subroutine level if LSAR3 not zero.
1	1	1	0	= Branch and switch to subroutine level if TDR bits 8....15 do not contain decimal data.
1	0	0	1	= Branch in level if current ALU exit is zero.
1	0	1	1	= Branch in level if current ALU exit not zero.
1	1	0	1	= Branch in level if accumulated ALU result is zero. If last operation was executed with six correction enabled, then the condition is satisfied if ALU bits 8....15 are zero.
1	1	1	1	= Branch in level if TDR bit position 0 contains a logical 1.
0	1	1	0	= Branch in level if the last ALU operation with accumulate bit on produced a carry out of ALU position 0. If this was an operation with six correction enabled, then the condition is satisfied by a carry out of ALU position 8.
0	1	1	1	= Branch in level if MSC signals length count overdraw, and reset this signal.

Group 11: Shift Instructions

Primary Function. The contents of CDR are shifted by any amount from 0 to 15 bit positions either to the left or right and the shift result is returned either to CDR or TDR.

Secondary Functions

- The shift amount can be specified direct (in the instruction) or can be taken from one of the 4 LSARs .
- The shift amount can be specified either in true or complement form.
- The contents of TDR can be propagated into LSARs 1 and 2 or 2 and 3 or all LSARs.
- The invert switch can be set to true, invert, force ones or force zeros.
- The shift instruction can initiate a return from a subroutine.
- For mnemonic NOP(R) the bit pattern of the instruction word is similar to SRC(R), only instruction bits 16...23 must be zero. Bits 12..15 (LSAR setting and invert switch) keep their normal functions and may be one or zero.

Common Layout of Shift Instructions

```
   ----------
        .
        1        Invert Bit
        2        Parity Bit
Byte0   3_______PFCN________
        .
        5  1
        6  0
   _____7  0    Op Code, Shift
        8  0
        9  1
       10  1 = leave subroutine
Byte1  11  0 = Op Code
       12  1 = set LSAR 0 and 1
       13  1 = set LSAR 2 and 3
       14
   ____15______invert switch function
       16  1 = left/0=right
       17  0 = direct/1=indirect
Byte2  18  0 = true/1=complement
       19  1 = TDR/0=CDR
       20                |20
       21  direct        |21___ignored_________
       22 shift ammount  |22
   ____23_______________|23___LSAR address____
```

Bit Function Description

Bit 1, Invert. Generated internally by hardware.

Bit 2, Parity Bit. Generated by assembler program.

Bit 3, PFCN - Parity Function Bit. Generated by assembler program.

Bits 5....9 and 11, Op Code. These bits represent a pattern that is unique and common to all shift instructions.

Bit 10, Leave Subroutine. This bit provides the means to return from a subroutine. The instruction which has bit 10 on is still part of the subroutine and so is the next instruction. However, the instruction thereafter is the first one in the previous level.

Bit 12, Set LSAR 0 and 1. This bit causes TDR bits 0...3 to be propagated to LSAR 0 and TDR bits 4....7 to LSAR 1.

Bit 13, Set LSAR 2 and 3. This bit causes TDR bits 8...11 to be propagated to LSAR2 and TDR bits 12...15 to LSAR 3.

Bit 14 and 15, Invert Switch Function. These bits determine the function of the invert switch as follows:

Bit 14	Bit 15	
0	0	= invert
0	1	= true
1	0	= force ones
1	1	= force zeros

Bit 16, Left/Right. This bit determines the shift direction as toward left (high order) or right (low order).

Bit 17, Direct/Indirect. This bit defines the facility that is to provide the shift amount. If direct is specified, instruction bits 20...23 represent the shift amount. If indirect is specified, instruction bits 22 and 23 address an LSAR the contents of which represent the shift amount .

Note: The indirect specification of the shift amount can conflict with LSAR setting if TDR-data is propagated into that LSAR which provides the shift amount . Results are unpredictable.

Bit 18, True/Complement. This bit defines how the shift amount is to be interpreted. If true is specified (bit 18=0) the shift amount is used as is. If complement is specified (bit 18=1), every logical 1 in the shift amount is interpreted as logical zero, and vice versa (15's complement).

Bit 19, TDR/CDR. This bit defines the register into which the shift result is to be stored.

Bits 20...23, Shift Amount. These bits represent in binary notation the

number of bit positions by which the CDR contents are to be shifted, provided bit 17 is 0 (direct). Whether the shift amount is taken as a true or complement number is defined by bit 18.

Bits 22 and 23, LSAR Address. If bit 17 specifies indirect, bits 20 and 21 are ignored and bits 22 and 23 represent the address of the LSAR that is to provide the shift amount .

Group 12: Sense and Control Instructions

Sense

A byte is sensed from the addressed facility and placed either right or left adjusted into either TDR or CDR. The unused portion of TDR or CDR remains unchanged.

Control

Used to set a specific register, or to activate a functional line as defined by the control number.

Common Layout of Sense and Control Instructions

```
   ----------
         .
         1        Invert Bit
         2        Parity Bit
Byte0    3        PFCN
         .
         5   1
         6   0
   ------7   0    Cp Code
         8   1
         9   0
   ---------------------------------
        10       0=no function / 1=return to main routine
Byte1   11   1 = TDR/0 = CDR
        12   1 = set LSAR 0 and 1          Bits 11 through 15 are
        13   1 = left/0 = right
        14                                ignored in the control instruction
   -----15       invert switch function
        16       0 = sense / 1 = control
        17
Byte2   18
        19        Sense address
        20              or
        21        control number
        22
   -----23
```

Secondary Functions

- For sense instructions the invert switch can be set to true, invert, force ones or force zeros.
- Both instructions can initiate a return from the subroutine.

Bit Function Description

Bit 1, Invert. Generated internally by hardware.

Bit 2, Parity Bit. Generated by assembler program.

Bit 3, PFCN - Parity Function Bit. Generated by assembler program.

Bits 5....9, and 16, Op Code. These bits represent a pattern that is unique and common to all sense instructions.

Bit 10, Leave Subroutine. This bit allows the program to return from a subroutine to the level that was in effect prior to the last level switching operation. The instruction that has bit 10 on is still part of the subroutine and so is the next one. However, the instruction thereafter is the first one in the previous level.

Bit 11, TDR/CDR. This bit specifies the register into which the sensed data is placed.

Bit 12, Set LSAR 0 and 1. This bit causes TDR bits 0...3 to be propagated to LSAR 0 and TDR bits 4...7 to be propagated to LSAR 1.

Bit 13. 1=left/0=right.

Bits 14 and 15, Invert Switch Function. These bits specify the function of the invert switch as fcllows:

Bit 14	Bit 15	
0	0	= invert
0	1	= true
1	0	= force ones
1	1	= force zeros

Bits 17....23, Sense Address or Control Number. These bits represent binary numbers which address the facility to be sensed (for sense) or to be activated (for control). The following addresses are assigned:

Sense Table

Sense Addr. Instr. Bits 17....23	Sensed into... CDR (bit 11 = 0) TDR (bit 11 = 1) Left (13 = 1)	Right (13 = 0)	Sensed Data
010 0000	0...7	8...15	TOD Byte 5 (TOD Bits 40...47)
010 0001	0...7	8...15	" " 4 (" " 32...39)
010 0010	0...7	8...15	" " 3 (" " 24...31)
010 0011	0...7	8...15	" " 2 (" " 16...23)
010 0100	0...7	8...15	" " 1 (" " 8...15)
010 0101	0...7	8...15	" " 0 (" " 0...15)
010 1000	0...7	8...15	Difference Byte 5 (Bits 40...47)
010 1001	0...7	8...15	between " 4 (" 32...39)
010 1010	0...7	8...15	TOD-Clock " 3 (" 24...31)
010 1011	0...7	8...15	and " 2 (" 16...23)
010 1100	0...7	8...15	TOD- " 1 (" 8...15)
010 1101	0...7	8...15	Comperator " 0 (" 0....7)
011 0000	0...7	8...15	CPU Timer Byte 5 (CPU Timer 40...47)
011 0001	0...7	8...15	" " " 4 (" " 32...39)
011 0010	0...7	8...15	" " " 3 (" " 24...31)
011 0011	0...7	8...15	" " " 2 (" " 16...23)
011 0100	0...7	8...15	" " " 1 (" " 8...15)
011 0101	0...7	8...15	" " " 0 (" " 0....7)
011 1001	0...7	8...15	Accum. update carries from Loc. 80 counter bit 24 to bit 23
100 0000	0,1	8,9	Don't care
	2	10	(Not) Timer Check
	3	11	(Not) Loc. 80 Update Requ.
	4	12	(Not) Comp. Int.
	5	13	(Not) CPU Timer Int.
	6	14	(Not) Loc. 80 Timer Int.
	7	15	(Not) Key Int.
101 0000	0	8	ALU 8...15 zero
	1	9	(Not) Except. Cond.
	2...7	10...15	Don't care
110 1000	0	8	MSC LS Addr.0
	1	9	" " " 1
	2	10	LC Crossing
	3	11	No Associative Array Match
	4	12	Successful Branch
	5	13	Carry 0
	6	14	" 1
	7	15	Decrement
111 1000	0	8	IAR Cnt. 1
	1	9	" " 0
	2	10	CC 1
	3	11	CC 0
	4	12	(Not) Fixed Pt.
	5	13	Length Cnt. 2
	6	14	" " 1
	7	15	" " 0

Control Table

Control number (Instr. bits 17....23)	Function
010 0000	Set TOD byte 5 (bits 40...47) from CDR bits 8...15
010 0001	" " " 4 (" 32...39) " " " "
010 0010	" " " 3 (" 24...31) " " " "
010 0011	" " " 2 (" 16...23) " " " "
010 0100	" " " 1 (" 8...15) " " " "
010 0101	" " " 0 (" 0....7) " " " "
010 1000	Set Byte 5 (Bits 40...47)
010 1001	Difference " 4 (" 32...39)
010 1010	between " 3 (" 24...31)
010 1011	TOD-Clock " 2 (" 16...23)
010 1100	and TOD- " 1 (" 8...15)
010 1101	Comperator " 0 (" 0....7)
011 0000	Set CPU Timer byte 5 (bits 40...47) from CDR bits 8...15
011 0001	" " " " 4 (" 32...39) " " " "
011 0010	" " " " 3 (" 24...31) " " " "
011 0011	" " " " 2 (" 16...23) " " " "
011 0100	" " " " 1 (" 8...15) " " " "
011 0101	" " " " 0 (" 0....7) " " " "
011 1001	Set accum. update carries from Loc. 80 Counter bit 24 to bit 23
100 0000	Immediate Stop
100 0001	Set Indicate Bit 1 Latch
100 0010	" " " 2 "
100 0011	" " " 3 "
100 0100	Reset Trap Cond.
100 0101	Reset Stop Key Latch
100 0110	Set Cond. Code LSAR 0 (2,3)
100 0111	Reset Indicate Bit 1 Latch
100 1000	" " " 2 "
100 1001	" " " 3 "
100 1010	" " " 4 "
100 1011	Save (in case of trap condition)
100 1100	Set LSAR 2
100 1101	Set LSAR 3
100 1110	Set Indicate Bit 4 Latch
100 1111	Return Saved Info

Control Table (continued)

Control number (Instr. bits 17....23)	Function
101 0000	IAR Decrement Ctl
101 0001	Set Execute Latch
101 0010	Reset Execute Latch
101 0011	Set Progr. Event Rec. Latch
101 0100	Reset Progr. Event Rec. Latch
110 0000	Set Wait Latch
110 0001	Block I/O Interr. Requ. Latches
110 0010	Set Halt I/O or Halt DV Latch
110 0011	Reset IAR Counter
110 0100	Reset Wait Latch
110 0101	Reset IOP Error Latch
110 0110	Set Ext. Mach. Chk. Mask Latch
110 0111	Set FP Overflow Mask Latch
110 1000	Reset Previous Error Latch
110 1001	Reset Halt I/O or Halt DV Latch
110 1010	Reset SVP Int. Requ. Latch
110 1011	Ext. Bus Check Reset
110 1100	Open I/O Int. Requ. Latches
110 1101	Set Mask Latches (CDR 0..5,7,13)
110 1110	Set IOP Sel. Latches
110 1111	Reset SVP Response Latch
111 0000	Reset MS Addr. Stop Latch
111 0100	Set Cond. Code 0
111 0101	" " " 1
111 0110	" " " 2
111 0111	" " " 3
111 1000	Set Loc. 80 Timer Interrupt
111 1001	(Set) Reset Ext. Damage from CDR bit 15
111 1010	Set Ext. Masks: TOD Comp. Mask from CDR bit 8
	CPU Timer Mask " " " 9
	Loc. 80 Timer Mask " " " 12
	Int. Key Mask from " " " 13
111 1100	Rest Loc. 80 Update Requ.
111 1110	Reset Ext. Interr.: Loc. 80 Timer Int. from CDR bit 8
	Int.Key Int. " " " 9

Group 13: Table Look Up (Translate and Branch) Instruction

Function. This instruction is a branch to a specific address in the control storage.
The address of the next microinstruction is formed in various ways from:

- The immediate bits contained in this microinstruction,
- the contents of LSAR 0, and
- either TDR bits 0...3 or LSAR 0,1,2, or 3.

Instruction bits 20 through 23 define how the new address is to be composed. The four high order bits of CSAR are always set to 0001. The instruction located at the selected address is executed as next sequential microinstruction . The translate and branch instruction is primarily used at the beginning of the I-phase to branch to the execution routine required for processing a given System /360 or /370 op code.
If bit 11 is on (Test TLU), a special test has to be performed (Compare Cond. Code). If the result is equal, subroutine level is forced and the branch takes place. If the result is not equal (no Cond.-Code match), the TRB can be regarded as a No-Operation (no level switching,no branch).

Note: Besides TRB, the version TRBR exists which defines a return from subroutine.

Layout of TRB Instruction

```
          ----------
            .
            1        Invert Bit
            2        Parity Bit
Byte0       3_______PFCN________
            .
            5  1
            6  0
          __7  0      Op Code TRB(R)
            8  0
            9  0
           10  0 = TRB / 1 = TRBR
Byte1      11       0 = normal TLU / 1 = Test TLU
           12      ignored
           13
           14
          _15_      immediate
           16        bits
           17
Byte2      18
           19
           20  0 = TDR bits 0...3 / 1 = LSAR 0,1,2, or 3
           21  1 = LSAR 0 indirect / 0 = Instr. bits 16...19
           22
          _23_____LSAR address_______
```

Bit Function Description

Bit 1. Invert. Generated internally by hardware.

Bit 2, Parity Bit. Generated by assembler program

Bit 3, PFCN = Parity Function Bit. Generated by assembler program.

Bit 11: Normal TLU / Test TLU

Normal TLU (Bit 11 = 0): With the normal TLU-Operation only one important exception must be realized. If instruction bits 13, 14, 15 are 1 0 1, the entries to CSAR-bits 8 and 14 will be interchanged by the logical circuitry (e.g. instr. bit 17 goes to CSAR bit 14, while TDR bit 3 enters CSAR bit 8). However CSAR bits 7,9,10,11 and 14 will be set with the inverted bits (see example 2).

Test TLU (Bit 11 = 1): This version of the TLU is used to simulate the System/360 or /370 Branch Op-codes 07 and 47 (BCR and BC).Therefore the circuitry first compares the Condition-Code Latches with the Mask-field in LSAR 2. If the two fields do not match, the /360 or /370 Branch instruction is nct effective (no Branch). Thus the microprogram can go to the next sequential instruction without executing the TRB instruction (same as NOP).
In case of CC-match the machine will force subroutine level (change IAR) and test LSAR 3 for zero. If LSAR 3 is not zero, CSAR bit 11 is set to one. If LSAR 3 is zero, CSAR bit 11 is set to zero. In either case the other CSAR bits are set according to Instruction bits 13 through 23. CSAR bits 7,9,10 and 14 are set with the inverted bits from the immediate field, TDR bits 0...3 and/or LSAR's (see example 3).

Bits 20 and 21 determine four methods of address composition, as follows:

Instruction bit 20 21	Address of next microinstruction CSAR bits 0 1 2 3	4 5 6	7 8 9 10	11 12 13 14
0 0			Instr. bits 16 17 18 19	TDR bits 0 1 2 3
0 1	Always set to 0 0 0 1	Instruction bits 13 14 15	LSAR 0	TDR bits 0 1 2 3
1 0			Instr. bits 16 17 18 19	LSAR 0,1,2, or 3
1 1			LSAR 0	LSAR 0,1,2, or 3

Note: Bits entering CSAR positions 7,9,10,11 and 14 (underscored) are inverted before setting of CSAR.

Bits 22 and 23: If bit 20 = 1, these bits determine the address of the LSAR to be set into CSAR bits 11...14.

```
        Bit 22 Bit 23
           0      0    =  LSAR 0
           0      1    =  LSAR 1
           1      0    =  LSAR 2
           1      1    =  LSAR 3
```

Example 1: Normal TLU; bits 13,14,15 = not 1 0 1
bits 20...23 = 1 1 0 1

Bits in LSAR 0 and 1 = 0000 0111

Bits 7,9,10,11 and 14 inverted = 1 11 1 0

CSAR bits 7...14 (after TRB) = 1011 1110

Example 2: Normal TLU; bits 13,14,15 = 1 0 1
bits 20...23 = 1 1 0 1

Bits in LSAR 0 and 1 = 0000 0111

Second and last bit interchanged = 1 0

bits 7,9,10,11,14 inverted = 1 11 1 1

CSAR bits 7...14 (after TRB) = 1111 1111

Example 3: Test TLU (bit 11 = 1); bits 20...23 = 1 1 0 1

Bits in LSAR 0 and 1 = 0100 0111

LSAR 3 = not zero = 1

Bits 7,9,10 and 14 inverted = 1 11 0

CSAR bits 7...14 (after TRB) = 1111 1110

Section 2: IOP Microprogram Codes

IOP Instruction Group Determination

Group	Description	Object Code in Op-Register 3rd Hex. Char.	Object Code in Op-Register 5th Hex. Char.	Bit C2	Bit C3	Bit Y0
1	Branch Instructions	0...3	x	0	0	x
		4...7	0...7	0	1	0
2	Data-Storage Instructions	4...7 and	8...F	0	1	1
3	Move Instructions	8...B	x	1	0	x
4	Logical Instructions	C...F	x	1	1	x

IOP Microinstructions by Mnemonics

Mnemonic	Description	Group
ADD(U)	Add register to register, reset previous carry (and change IAR's).	4
ADDC(U)	Add register to register, use previous carry (and change IAR's).	4
ADDE(U)	Add register to register, use external carry (and change IAR's).	4
ADD1(U)	Add register to register, use forced carry (and change IAR's).	4
ADDI	Add immediate data to register, reset previous carry.	4
AND(U)	AND register to register (and change IAR's).	4
ANDI	AND immediate data with register into register	4
B(U)	Branch unconditionally (and change IAR's).	1
BC(U)	Branch on condition defined by mask (and change IAR's).	1
BCN(U)	Branch if 'Carry-FL'= On and Cond. Code = not zero (and change IAR's).	1
BCNR(U)	Branch on register if 'Carry-FL'= On and Cond. Code = not zero (change IAR's if condition is not met).	1
BCR(U)	Branch on register on condition defined by mask (change IAR's if condition is met).	1
BCY(U)	Branch if 'Carry-FL'=On (change IAR's if condition is met).	1
BCYR(U)	Branch on register if 'Carry-FL'=On (change IAR's if condition is met)	1
BNC(U)	Branch if 'Carry-FL'=Off (change IAR's if condition is met).	1
BNCR(U)	Branch on register if 'Carry-FL'=Off (change IAR's if condition is met).	1
BNZ(U)	Branch if Cond. Code= not zero (change IAR's if condition is met).	1
BNZR(U)	Branch on register if Cond. Code=not zero (change IAR's if condition is met).	1
BR(U)	Branch on register unconditionally (and change IAR's).	1
BZ(U)	Branch if condition code = zero (change IAR's if condition is met).	1
BZN(U)	Branch if condition code = zero, or if 'Carry-FL'=Off (change IAR's if either condition is met).	1
BZNR(U)	Branch on register if Cond.Code = zero, or if 'Carry-FL'=Off, (and change IAR's if either condition is met).	1
BZR(U)	Branch on register if Cond. Code = zero (change IAR's if condition is met).	1
EOR(U)	Exclusively OR register with register into register and set condition code (and change IAR's).	4
EORI	Exclusively OR immediate data with register into register and set condition code.	4
LBI	Load immediately byte into register.	3
LDEC	Load byte from data storage into register and decrement data storage address.	2
LINC	Load byte from data storage into register and increment data storage address.	2
LLKR	Load register from SVP-Link-Reg.	3
MV(U)	Move byte from register to register (change IAR's).	3
MVX(U)	Move byte from register to register with digit crossing, (change IAR's).	3
NOP	No operation.	1
OR(U)	OR register to register, (change IAR's).	4
ORI	OR immediate data with register.	4

Mnemonic	Description	Group
SABI	Store immediate byte into ALS-B.	3
SABR	Store byte from register into ALS-B.	3
SADI	Store immediate byte into ALS-D.	3
SADR	Store byte from register into ALS-D.	3
SDEC	Store byte from register into data storage and decrement data storage address.	2
SINC	Store byte from register into data storage and increment data storage address.	2
SLKI	Set immediate byte into SVP-Link-Reg.	3
SLKR	Set byte from register into SVP-Link-Reg.	3
SMODE	Set Mode buffer.	3
SZI	Set immediate byte into ZLS.	3
SZR	Set byte from register into ZLS.	3
TADD (TADU)	Reset previous carry, add register plus register into D-reg. and set new condition code (change IAR's).	4
TADDC (TADCU)	Add register plus register plus previous carry into D-reg, set new condition code (change IAR's).	4
TADDE (TADEU)	Add register plus register plus external carry into D-reg., set new condition code (change IAR's).	4
TADD1 (TAD1U)	Add register plus register plus forced carry into D-reg., set new condition code (change IAR's).	4
TADDI	Reset previous carry, add register plus immediate data into D-reg., set new condition code.	4
TAND(U)	AND two registers into D-reg. and set condition code (change IAR's).	4
TANDI	AND register with immediate data into D-Reg. and set condition code.	4
TBOFF	Test bit and branch if off.	1
TBON	Test bit and branch if on.	1
TEOR(U)	Exclusively OR two registers into D-reg. and set condition code (change IAR's).	4
TEORI	Exclusively OR register with immediate data into D-reg. and set condition code.	4
TIBOF	Test Bit and Branch if off, same as TBOFF Bit can be addressed indirectly.	1
TIBON	Test Bit and Branch if on, same as TBON Bit can be addressed indirectly.	1
TOR(U)	OR two registers into D-reg. and set condition code (change IAR's).	4
TORI	OR register with immediate data into D-reg. and set condition code.	4

Explanation of IOP Microinstruction Groups

Group 1: IOP Branch Instructions

Primary Function: The microprogram branches to another instruction anywhere in the control storage instead of continuing with the next sequential instruction. The branch may be unconditional or depending on a certain condition. In the latter case the microprogram proceeds with the instruction if the condition is not met.
The branch condition is specified either directly by the mnemonic or in conjunction with the mnemonics BC and BCR by a separate parameter (mask bits in hex.).

Secondary Functions

- Program level switching is possible in case of a successful branch. (See under bit function description for bit Y0.)
- With the mnemonic NOP the microprogram performs no operation but it proceeds with the next microinstruction.

Layout of IOP Group 1 Instructions

Mnemonic	C (4 5 6 7 2 3)	R (2 3 4 5 6 7)	Y (0 1 2 3 4 5 6 7)
	Number of Hex Character in Assembler List		
	2nd | 3rd	4th	5th | 6th
B(U)	0 0 |		Branch Displacement Address
BR(U)	0 1 |		0 | Addr. of WR with CS-Disp
BNZ(U)	0 0 | 0 1	Block-part of CS-Branch-Address	S u f f i x | Branch Displacement Address
BCY(U)	0 0 | 1 0		
BCN(U)	0 0 | 1 1		
BZ(U)	1 0 | 0 1		
BNC(U)	1 0 | 1 0		
BZN(U)	1 0 | 1 1		
BNZR(U)	0 0 | 0 1		Bit Pos. | Address of W-Reg. containing CS-Displacement
BCYR(U)	0 0 | 1 0		
BCNR(U)	0 0 | 1 1		
BZR(U)	1 0 | 0 1		
BNCR(U)	1 0 | 1 0		
BZNR(U)	1 0 | 1 1		
BC(U)	M1 0 | M2 M3		Branch Displacement Address | See Note 1
BCR(U)	M1 1 | M2 M3		0 | Addr. of WR with CS-Disp | See Note 1
NOP	1 0 | 0 0 0	0 0 Don't care	0 | Don't care
TBON	0 | Bit-	0 1 WR tested	0 | Branch Displacement Address | See Note 2
TBOFF	1 | Pos.		

Note 1: For Mask bits M1, M2, M3 see next table.

Note 2: With 'Bit test'-Instructions bit position 3 cannot be tested on its own. If instruction bits C5...C7 are 0 1 1 (hex 3), this means for :

TBON: Branch if any bit of the selected byte is on.

TBOFF: Branch if all bits off the selected byte are off.

Table of parameters used with mnemonics BC(U) and BCR(U)

Parameter	Same as mnemonic	Branch if:	Instr. Bits C4 C5 C6 C7 M1 M2 M3 (Mask Bits)
x'1'	BNZ(U)	Condition Code =	0 0 0 1
	BNZR(U)	not zero	0 1 0 1
x'2'	BCY(U)	Carry-Fl = On	0 0 1 0
	BCYR(U)		0 1 1 0
x'3'	BCN(U)	Carry-Fl = On and	0 0 1 1
	BCNR(U)	CC = not zero	0 1 1 1
x'9'	BZ(U)	Condition Code =	1 0 0 1
	BZR(U)	zero	1 1 0 1
x'A'	BNC(U)	Carry-Fl = Off	1 0 1 0
	BNCR(U)		1 1 1 0
x'B'	BZN(U)	Cond. Code = zero	1 0 1 1
	BZNR(U)	or Carry-Fl = Off	1 1 1 1

Bit function description

C-field This field represents the Op-code including the branch condition. For mnemonics TBON and TBOFF bits C5,C6 and C7 define the bit to be tested.

R-field: With Branch instructions this field contains the block part of the branch address (6 high order bits of the 13 bits of the next instruction).
With TBON and TBOFF the R-field is used to define the register containing the bit to be tested.
With mnemonic NOP this field may be disregarded.

Y-field: Y0 = Suffix U-bit. This bit is on for all suffix-U mnemonics (last character of mnemonic = U). It causes the microprogram to change IAR's (main routine IAR to subroutine IAR and vice versa) in case of a successful branch. In such a case the address of the next sequential instruction will be saved in the old IAR and the branch address derived from the R- and Y-field of the branch instruction will be loaded into the new IAR before control is switched to the new IAR.

Y1...Y7: This field contains either the branch displacement (7 low order bits of the 13 bit branch address) directly, or (with branch on register instructions) it defines the register containing the displacement. The displacement is loaded into ALS-D bit 1 through 7 of the controlling IAR, while the block part of the address (R-field) enters ALS-B bit 2 through 7.
With mnemonics TBON and TBOFF the block part of the instruction address remains unchanged.
With mnemonic NOP the whole Y-field may be disregarded.

Group 2: IOP Data Storage Instructions

Primary Function: To either store a byte from a register into data storage or to load a byte from data storage into a register. Multibyte transfer is possible if the multiple byte bit (ALS-B bit 0) has been set by a previous instruction. In such a case storage- and register-address will be incremented by one after each byte transfer. The operation is repeated until a double word boundary (8 bytes) in the register area is reached. In other words the operation is ended after a register with three loworder one-bits in its address has been loaded or stored.

Secondary Functions

- It is possible to increment or decrement the data-storage address contained in the register defined by the R-field. It must be emphasized that this type of increment/decrement has nothing to do with the incrementing in multibyte transfer. It is performed after the byte transfer, therefore, it affects the program only when this instruction is executed the next time.

Layout of IOP Group 2 Instructions

	C	R	Y	
	4 5 6 7 2 3	2 3 4 5 6 7	0 1 2 3 4 5 6 7	
Mnemonic	Number of Hex Character in Assembler List			
	2nd	3rd	4th	5th / 6th
SINC	0 Incr.		Address of	0
SDEC	1 or		Work-Reg.	0 Address of
	Decr.	0 1	containing	1 - work-Reg.
LINC	0 Am-		Data-Store	1 (Data-Reg.)
LDEC	1 ount		Address	1

Bit function description

C-field Bits C4, C2, C3 together with bits Y0 and Y1 represent the Op-Code.
Bits C5, C6 and C7 define the increment/decrement amount in the following way:

Instruction bits C5 C6 C7	Increment by:	Decrement by:
0 0 0	0	8
0 0 1	1	7
0 1 0	2	6
0 1 1	3	5
1 0 0	4	4
1 0 1	5	3
1 1 0	6	2
1 1 1	7	1

The increment/decrement part of the operation can be considered as an add or subtract immediate operation which is executed after the load/store part. It is used to modify the data storage address before this instruction is executed again.

R-field: Bits R2 through R7 define the working register from which the data storage address will be taken. Only the displacement (7 low order bits) is contained in this register. The block portion of the storage address is taken from ALS-B bit 2...7. The contents of this register are incremented or decremented according to the value of bits C5, C6 and C7.

Y-field: Bits Y0 and Y1 together with bits C4, C2 and C3 represent the Op-code. Bits Y2 through Y7 define the working register which will receive data in case of a load instruction or from which data will be taken in case of a store operation. If the multiple byte bit in ALS-B is on, the load/ store operation starts at this working register.

Group 3: IOP Move Instructions

Primary Function: To move a byte of information from one location to another.

Secondary Functions

- With mnemonics MV or MVX level switching (main routine to subroutine or vice versa) is possible. In these cases the mnemonic is MVU or MVXU respectively and instruction bit Y0 is on.
- If the mnemonic is MVX or MVXU, the two digits (hex characters) will be crossed.
- For special Store and Load operations see under bit function description of R- and Y-field.

Layout of IOP Group 3 Instructions

Mnemonic	C (4 5 6 7 2 3) 2nd		3rd	R (2 3 4 5 6 7) 4th	Y (0 1 2 3 4 5 6 7) 5th	6th
	Number of Hex Character in Assembler List					
SABI		0	0	ALS-B Addr		
SADI		0	1	ALS-D Addr		
	0 0 1				Immediate Data	
SZI		1	0	ZLS Addr.		
SLKI		1	1	Cntl-Funct		
SMODE		0	0	P0P1P2M0M1	1 0	Don't care
SABR		0	0	ALS-B Addr	0 0	Address
SADR	0 1 1	0	1 0 1	ALS-D Addr	0 0	of
SZR		1	0	ZLS Addr.	0 0	'From'-
SLKR		1	1	Cntl-Funct	0 0	Register
LLKR	1 0 1	1		see Note	0 0	'To'Reg-Addr
LBI	1 0 0	0			Immediate Data	
MV(U)		0		Address	U B i t	Address
MVX(U)	1 1 0	1		of 'To'-Reg.	0 0	of 'From'-Reg.

Note: For details see description of mnemonic LLKR.

Bit function description

C-field: These 6 instruction bits represent the Op-code. Unless the Y-field contains immediate data, instruction bits Y0 and Y1 belong to the Op-code as well.

R- and Y-field: The meaning of these two fields is dependent on the Op-code:

LBI, MV(U), MVX(U): With these mnemonics the R-field defines the receiving register. The Y-field contains either immediate data or two bits of the Op-code and the 'From register' (see layout). If bit Y0 is on (suffix 'U' bit), this causes the microprogram to change from MIAR to SIAR (main routine to subroutine) and vice versa. Instr. bit C7 being on causes digit crossing (mnemonic MVX(U). In this case bits 0...3 of the 'From-Reg.' go to bits 4...7 of the 'To-Reg.' while bits 4...7 go to bit 0...3.

SABI/SABR: The purpose of these instructions is to store information into Address Local Storage B.

R2 bit: is always 0.

R3...R7: These bits contain the address of the ALS-B where the information is to be stored.

Y0...Y7: These instruction bits contain either the information (immediate data) if the mnemonic is SABI or the address of the work register from where the information is to be fetched if the mnemonic is SABR. In the latter case bits Y0 amd Y1 belong to the Op-code and must be 00.

SADI/SADR: The purpose of these two instructions is to store information intc Address Local Storage D.

R2 bit: Is always 1.

R3...R7: These bits contain the address of the ALS-D where the information is to be stored.

Y0...Y7: These instruction bits contain either the information (immediate data) if the mnemonic is SADI, or the address of the work register from where the information is to be fetched if the mnemonic is SADR. In the latter case bits Y0 and Y1 are part of the Op-code and must be CC.

SZI/SZR: The purpose of these two instructions is to store information into the Zone Local Storage.

R-field:

R2 is always zero.

R3...R7: These 5 bits contain the address of the ZLS location where the information is to be stored.

Y-field: For mnemonic SZI these bits contain the information (immediate data) to be stored into ZLS.
If the mnemonic is SZR, bit Y0 and Y1 must be 00, while bits Y2 through Y7 contain the address of the work register from where the information is to be fetched.

SLKI/SLKR: These instructions serve for twc purposes:

- Transfer of data to the SVP.
- Control functions.

R-field:

Bit R2: is always 1.

Bits R3 through R7 = Control Function Bits: These bits are used to determine the control functions which may be performed besides the data transfer to the SVP.

Bit R3: If this bit is on, the contents of the Y-field (if SLKI) or the contents of the work register determined by the Y-field bits 2 through 7 (if SLKR) will be transferred to the X-register. If bit R3 is off, no data will be transferred.

Bit R4: If this bit is on, the PCR-FL (Program Controlled Request from ICP to SVP) will be set.

Bit R5: If this bit is on, the SVP-Request-FL (from SVP to IOP) will be reset.

Bit R6: Don't care.

Bit R7: If this bit is on, it causes the 'Prevent I/O-FL' to be reset.

Note: If R-field bits 3,4,5 and 7 are all off, the SIKI/ SLKR operation performs no function and thus it can be regarded as an NOP instruction.

LLKR: This instruction is used to transfer data from the X-register into a work register.

R-field:

Bit R2: Don't care.

Bits R3 through R6: Always zero.

Bit R7: If this bit is off, the work register is loaded with the contents of the X-register.
If bit R7 is on, the work register is loaded as follows:
X-reg bits 0...5 into work register bits 0...5
'SVP-requ.-FL' on: set 1 into work register bit 6
'PCR-FL'on: set 1 into work reg. bit 7.

Y-field: bits 0 and 1 are always 00. Bits 2...7 contain the address of the receiving work register.

SMODE: This instruction allows the setting (changing) of any mode buffer by any microprogram.

R-field:

Bit 2: Always zero.

Bits 3...5: These three bits address the mode buffer to be loaded.

Bits 6 and 7: The contents of these two bits are set into the mode buffer.

R6 = M0
R7 = M1

Y-field: Bits 0 and 1 = always 1 0. Bits 2 through 7 = don't care.

Group 4: Logical IOP Instructions

Primary Function: The contents of the 'From'-reg. and the 'To'-reg. are locially combined by the ALU (Added, ANDed, ORed or Exclusively ORed). Unless it is a test type instruction, the result is stored into the 'To' work register defined by the R-field.

Secondary Functions

- The condition code is set depending on the ALU output to indicate whether the ALU output (D-reg.) was zero or not and to indicate a carry or no carry out of the high order position of the ALU.
- With test operations (mnemonic Txx..) the result is not stored into the 'To' work register, because the purpose of these instructions is only the setting of the condition code.
- With suffix 'U' instructions MIAR and SIAR will be interchanged (level switching).
- With Add-instructions it is possible to
  - Reset the previous carry
  - Use the previous carry
  - Force a carry (one)
  - Use an external carry from the front end.

Layout of IOP Group 4 Instructions

Mnemonic	C 4 5 6 7 2 3	R 2 3 4 5 6 7	Y 0 1 2 3 4 5 6 7
	Number of Hex Character in Assembler List: 2nd / 3rd / 4th / 5th / 6th		
ANDI	0 0		
CRI	0 1		
	1		Immediate
EORI	1 0		
ADDI	1 1		
	0		
TANDI	0 0		
TCRI	0 1		
	0		Data
TEORI	1 0		
TADDI	1 1		
	-		
AND	0 0		
CR	0 1		1
EOR	1 0		
	---	Address	0 -
ADD	0 0		
ADDC	0 1		0
ADDE	1 0		
ADD1	1 1		
	1	1 1 of	- - Address
ANDU	0 0		
ORU	0 1		1
ECRU	1 0		
	---	'To'-	1 - of
ADDU	0 0		
ADDCU	0 1		0
ADDEU	1 0		
ADD1U	1 1	Register	'From'-
	1		-
TAND	0 0		
TCR	0 1		1
TEOR	1 0		Register
	---		0 -
TADD	0 0		
TADDC	0 1		0
TADDE	1 0		
TADD1	1 1		
	0		-
TANDU	0 0		
TORU	0 1		1
TEORU	1 0		
	---		1 -
TADU	0 0		
TADCU	0 1		0
TADEU	1 0		
TAD1U	1 1		

Bit_function_description

C-field: This field represents the Op-code. Unless the instruction is of the RI-format where the Y-field contains immediate data, bits Y0 and Y1 also belong to the Op-code.

Bit_C4: If this bit is on, the result is not transferred from the D-register to the 'To'-register (test type instruction).

Bit_C5: Off = RI-format On = RR-format.

Bits_C6_and_C7: With the four different add operations, these two bits are used for carry control.

R-field: The content of this field defines the 'To'-register. Except for test type instructions (mnemonic Tx...), the 'To'-register contains one of the two operands before and the result after the operation. For test type instructions the content of the 'To'-register (operand) remains unchanged.

Y-field:

With RI-format instructions (bit C5 = off) this field contains one byte of immediate data.

With RR-format instructions (bit C5 = on) bits Y2 through Y7 represent the 'From'-register address and bits Y0 and Y1 are part of the Op-code. Bit Y0 is the suffix-'U'-bit. If the suffix 'U'-bit is on, IAR's (MIAR and SIAR) will be interchanged after the operation is executed. Thus the program will switch from subroutine level to main routine level or vice versa after storing the address of the next sequential instruction in the 'old' IAR.

Section 3: SVP Microprogram Codes

SVP Op Codes by Bit Pattern

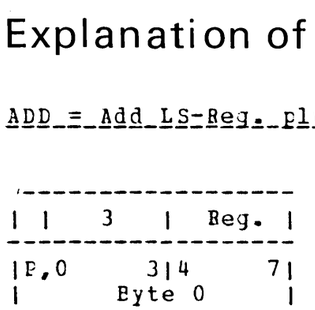
Byte 0	Mnemonic
0 x	CHECK
1 x	OR
2 x	XOR
3 x	ADD
4 x	AND
5 x	BZR
6 x	LBI
7 x	LBR
8 x	STROB
9 x	FR
A x	STR
B x	SF
C x	SST
D 0	ANDI
D 1	CRI
D 2	XORI
D 3	ADDI
D 4	SLS
D 5	LBAP
D 6	LDAC
D 7	BZ
D 8	B
D 9	BNZ
D A	STOP
D B	NOP
D C	Branch to stop (no mnemonic)
D D	STBX
D E	STBA
D F	CTB
E x	BR
F x	CHECK

Summary of SVP Mnemonics

Mnemonic	Description
ADD	Add LS-Reg. plus Accu into Accu
ADDI	Add Accu plus Immediate data byte into Accu
AND	AND LS-Reg. with Accu into Accu
ANDI	AND Accu with Immediate data byte into Accu
B	Branch unconditionally
BNZ	Branch if (Not) ALU zero
BR	Branch unconditionally to address contained in register
BZ	Branch if ALU zero
BZR	Branch to address contained in register if ALU zero
CHECK	Op Code check
CTB	Add LS-Reg. plus constant into LS-Reg., exclusively OR result with mask into Accu, branch if ALU zero
FR	Fetch one byte from LS-Reg. into Accu
LBAP	Load Bus and Parity bit from LS-Reg. into BAR (BAR's) or BDR (BDR's)
LBI	Load Immediate data byte into LS-Reg.
LBR	Load Accu into Bus-Reg.
LDAC	Load Immediate data byte into Accu
NOP	No operation
OR	Logically OR LS-Reg. with Accu into Accu
ORI	Logically OR Accu with Immediate data byte into Accu
SF	Fetch one byte from storage into Accu
SLS	Switch LS-Zone
SST	Store one byte from Accu into storage
STBA	Sense, AND with mask into Accu, Branch if ALU-out = zero
STBX	Sense, exclusively OR with mask into Accu, Branch if ALU zero
STOP	
STR	Store one byte from Accu into LS-Reg.
STROB	Sense from Bus into Accu, or activate CTRL Strobe Bus 0 and/or 1
XOR	Exclusively OR LS-Reg. with Accu into Accu
XORI	Exclusively OR Accu with Immediate data byte into Accu

Explanation of SVP Mnemonics

ADD = Add LS-Reg. plus Accu into Accu

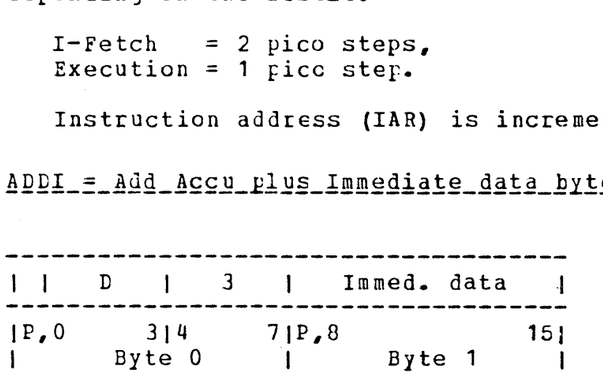


The contents of a local storage register addressed by instruction bits 4...7 are added to the contents of the Accu. The result is stored into the Accu. The LS-Zone is selected by the LSZR. A carry from a previous add operation is added intc bit position 7 of the ALU. A carry out of position 0 causes the 'Carry Fl' to be set. The 'ALU zero Fl' is set or reset depending on the result.

I-Fetch = 2 pico steps,
Execution = 1 picc step.

Instruction address (IAR) is incremented by 1.

ADDI = Add Accu plus Immediate data byte into Accu

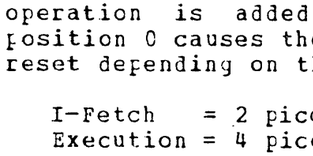


The Immediate data byte (instruction byte 1) is added to the contents of the Accu. The result is stored into the Accu. A carry from a previous Add operation is added into bit position 7 of the ALU. A carry out of ALU position 0 causes the 'Carry Fl' to be set. The 'ALU zero Fl' is set or reset depending on the result.

I-Fetch = 2 pico steps,
Execution = 4 pico steps

Instruction address (IAR) is incremented by 2.

AND = AND LS-Reg. with Accu into Accu

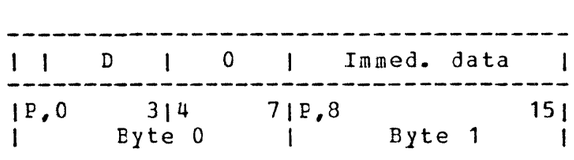


The contents of a local storage register addressed by instruction bits 4...7 are anded with the contents of the Accu. The result is stored into the Accu. The LS-Zone is selected depending on the contents of LSZR. The 'Carry-Fl' is reset before and cannot be set by this operation. The 'ALU zero Fl' is set or reset depending on the result.

I-Fetch = 2 pico steps,
Execution = 1 pico step.

Instruction address (IAR) is incremented by 1.

ANDI = AND Accu with Immediate data byte into Accu

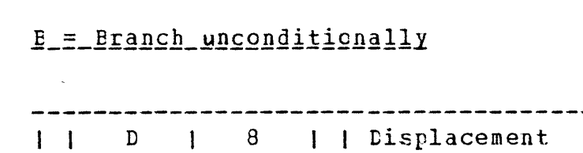


The contents of the Accu are ANDed with the Immediate data byte. The result is stored into the Accu. The 'Carry Fl' is reset before and cannot be set by this operation. The 'ALU zero Fl' is set or reset depending on the result.

I-Fetch = 2 pico steps,
Execution = 4 pico steps

Instruction address (IAR) is incremented by 2.

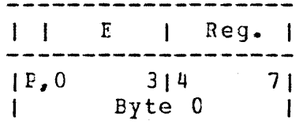
B = Branch unconditionally



Instruction bits 8 through 15 (byte 1) are set into the low order byte (odd numbered LS-Reg.) of the current IAR. This provides the means to branch within a 256-byte block.

I-Fetch = 2 pico steps,
Execution = 4 pico steps.

BR = Branch unconditionally to address contained in register



This instruction can be performed in two ways:

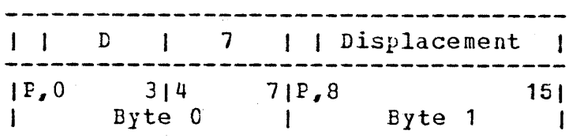
- Instruction bit 4 = one: The low order byte (displacement) of the current IAR is changed to the value contained in the LS-Register defined by instruction bits 4 through 7. Only a branch within the current 256-byte block can be performed in this manner.

- Instruction bit 4 = zero: Another pair of LS-Registers becomes IAR in order to branch to another 256-byte block. LS-Reg. pairs 0-1, 2-3, 4-5, or 6-7 can be used as IAR. The pair is defined by instruction bits 5,6 and 7. However the value of bit 7 is ineffective and always considered to be zero, since the high order byte of the instruction address must be in an even numbered LS-Register (0,2,4, or 6). The LS-zone is defined by the contents of LSZR.

I-Fetch = 2 pico steps,
Execution = 2 pico steps.

The current (old) Instruction address (IAR) is incremented by 1 if instruction bit 4 is a logical zero.

BZ = Branch if ALU zero

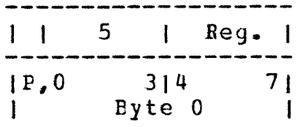


If the 'ALU zero Fl' is on instruction bits 8 through 15 (byte 1) are set into the low order byte (odd numbered LS-Reg.) of the current IAR. This provides the means to branch within a 256-byte block.

I-Fetch = 2 pico steps,
Execution = 4 pico steps.

Instruction address (IAR) is incremented by 2 if the branch does not take place. (ALU zero = on.)

BZR = Branch to address contained in register if ALU zero.



The microprogram branches only if the 'ALU zero Fl' is on. If the branch takes place, it will be performed in one of two ways depending on the value of instruction bit 4.

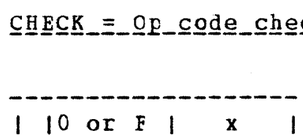
- Instruction bit 4 = one: The low order byte of the current IAR is changed to the value contained in the LS-Register defined by instruction bits 4 through 7. Only a branch within the current 256-byte block can be performed in this manner.

- Instruction bit 4 = zero: Another pair of LS-Registers becomes IAR in order to branch to another 256-byte block. LS-Reg. pairs 0-1, 2-3, 4-5, or 6-7 can be used as IAR. The pair is defined by instruction bits 5,6 and 7. However the value of bit 7 is ineffective and always considered to be zero, since the high order byte of the instruction address must be in an even numbered LS-Register (0,2,4, or 6). The LS-zone is defined by the contents of LSZR.

I-Fetch = 2 pico steps,
Execution = 2 pico steps.

The current (old) Instruction address (IAR) is incremented by 1 if the branch does not take place, or if the entire IAR is changed (instr. bit 4 is a logical 0).

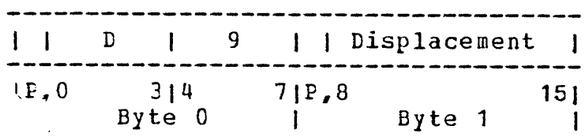
CHECK = Op code check



Any Op code starting with either 0 or F in the first four bits is considered to be invalid. If such a non-valid Op code is detected in the Storage Data Register, the SVP stops with the check light at the keyboard turned on. Restart is possible only via IMPL key.

Instruction address (IAR) is not updated.

BNZ = Branch if (Not) ALU zero

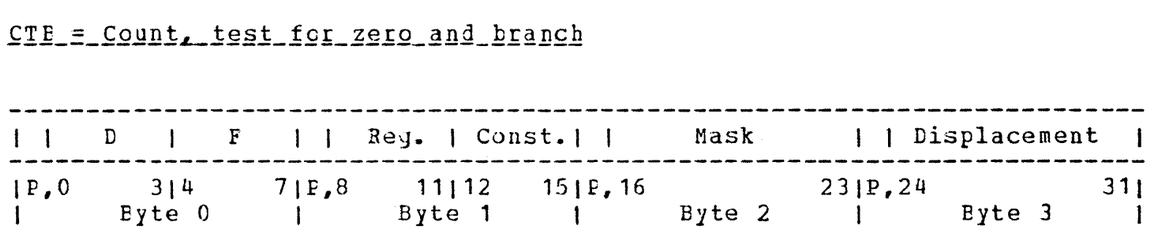


If the 'ALU zero Fl' is off instruction bits 8 through 15 (byte 1) are set into the low order byte (odd numbered LS-Reg.) of the current IAR. This provides the means to branch within a 256-byte block.

I-Fetch = 2 pico steps,
Execution = 4 pico steps.

Instruction address (IAR) is incremented by 2 if the branch does not take place. (ALU zero = on.)

CTE = Count, test for zero and branch

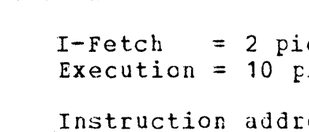


This instruction provides the means to add a constant to the contents of an LS-reg. and to branch within a 256-byte block if the result matches a mask. The contents of the LS-reg. addressed by instruction bits 8 through 11 is added to the constant from instuction bits 12 through 15. The result is stored into the same LS-reg. from which the first operand was taken.
Thereafter the result is exclusively ORed with the mask (instruction bits 16 through 23) and the new result remains in the accumulator. If this new result of the Exclusive OR operation is zero, the displacement from instruction bits 24 through 31 is placed into the low order byte of the current IAR thus causing a branch within this 256-byte block.

I-Fetch = 2 pico steps,
Execution = 10 pico steps.

Instruction address (IAR) is incremented by 4 if the branch is not taken.

'R = Fetch one byte from LS-Reg. into Accu

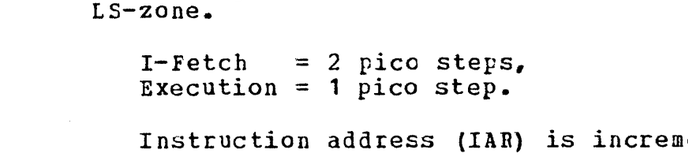


A byte is fetched from any LS-register into the Accu. The LS-register is defined by instruction bits 4 through 7. The contents of LSZR define the LS-zone.

I-Fetch = 2 pico steps,
Execution = 1 pico step.

Instruction address (IAR) is incremented by 1.

LBAP = Load Bus and Parity bit from LS-Reg. into BAR(s) or BDR(s)



This instruction provides the means to place any value into the data or address register of either one or both bus systems and to set either odd or even parity with this value onto the address bus. Instruction bits 8,9 and 10 select a pair of LS-regs. (even/odd numbered), bit 11 is ignored.
The odd numbered LS-reg. supplies the information that is placed into the Bus Data Reg. or the Bus Address Reg. selected by instruction bits 12,13 and 15.
The low order bit (bit 7) of the even numbered LS-reg. supplies the parity bit. If a Bus Address Reg. is specified, this bit 7 overwrites the parity bit that is normaly generated for the address bus.
The contents of LSZR defines the LS-zone. Instruction bits 12 and 13 are set into the A-Reg. from where they are decoded.
Bus registers are selected by instruction bits 11 through 15 as follows (bit 14 has no effect):

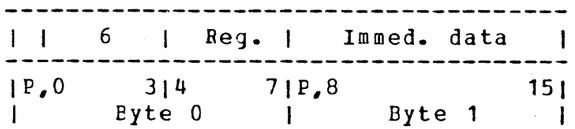
Bits 12 through 15 in hex:

0, 1, 2, or 3	= No operation
4 or 6	= Bus Address Reg. 1
5 or 7	= Bus Data Reg. 1
8 or A	= Bus Address Reg. 0
9 or B	= Bus Data Reg. 0
C or E	= Bus Address Reg. 0 and 1
D or F	= Bus Data Reg. 0 and 1

I-Fetch = 2 pico steps,
Execution = 5 pico steps.

Instruction address (IAR) is incremented by 2.

LBI = Load Immediate data byte into LS-register

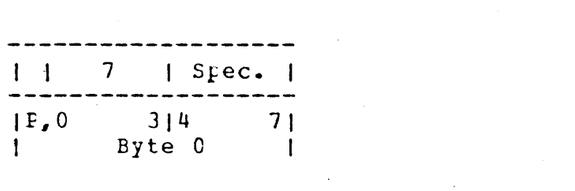


The immediate byte provided by instruction byte 2 is placed into the LS-register addressed by instruction bits 4 through 7.

I-Fetch = 2 pico steps,
Execution = 3 pico steps.

Instruction address (IAR) is incremented by 2.

LBR = Load Accu into Bus Register



The contents of the accumulator (including the parity bit) are placed into the bus register specified by instruction bits 4 through 7. Either the address register(s) or the data register(s) of either or both bus systems may be specified as follows:

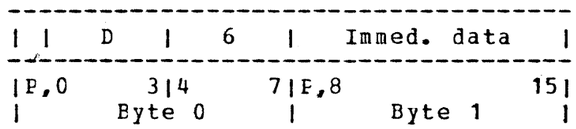
Instr. bits 4...7 in hex

0, 1, 2, or 3	= No Operation
4 or 6	= Bus Address Reg. 1
5 or 7	= Bus Data Reg. 1
8 or A	= Bus Address Reg. 0
9 or B	= Bus Data Reg. 0
C or E	= Bus Address Regs. 0 and 1
D or F	= Bus Data Regs. 0 and 1

I-Fetch = 2 pico steps,
Execution = 1 pico step.

Instruction address (IAR) is incremented by 1.

LDAC = Load immediate data byte into Accu

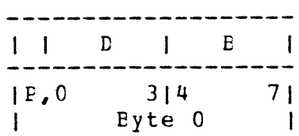


The immediate data byte provided by instruction bits 8 through 15 is placed into the accumulator.

I-Fetch = 2 pico steps,
Execution = 4 pico steps.

Instruction address (IAR) is incremented by 2.

NOP = No Operation

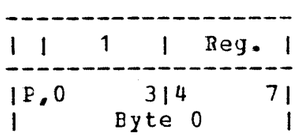


This Op-Code causes no operation. The microprogram continues with the next operation.

I-Fetch = 2 pico steps,
Execution = 2 pico steps.

Instruction address (IAR) is incremented by 1.

OR = Logically OR LS-Reg. with Accu into Accu

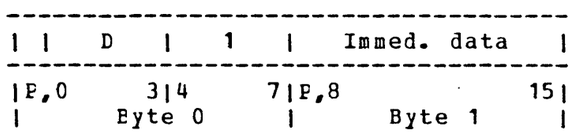


The contents of a local storage register addressed by instruction bits 4...7 are ORed with the contents of the Accu. The result is stored into the Accu. The LS-Zone is selected depending on the contents of LSZR. The 'Carry-Fl' is reset before and cannot be set by this operation. The 'ALU zero Fl' is set or reset depending on the result.

I-Fetch = 2 pico steps,
Execution = 1 pico step.

Instruction address (IAR) is incremented by 1.

CRI = Logically OR Accu with Immediate data byte into Accu

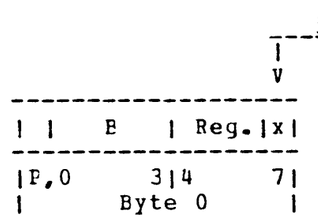


The contents of the Accu are ORed with the Immediate data byte (instruction byte 1). The result is stored into the Accu. The 'Carry Fl' is reset before and cannot be set by this operation. The 'ALU zero Fl' is set or reset depending on the result.

I-Fetch = 2 pico steps,
Execution = 4 pico steps

Instruction address (IAR) is incremented by 2.

SF = Fetch one byte from storage into Accu.

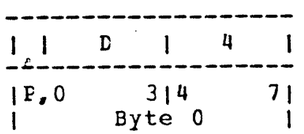


A single byte is fetched from the SVP storage and stored into the Accu. The 16-bit storage address is obtained from an adjacent pair (even+odd numbered) of LS registers. The even numbered LS register, containing the high order byte of the storage address, is defined by instruction bits 4,5 and 6. Instruction bit 7 is ignored. The contents of LSZR define the LS-zone. The storage address (in the LS-reg. pair) is automatically incremented by 1 via the ALU.

I-Fetch = 2 pico steps,
Execution = 3 pico steps.

Instruction address (IAR) is incremented by 1.

SLS = Switch Local Storage Zone



This instruction provides the means to select a new local storage zone (a new set of 16 registers) and, because each zone contains its own IAR pairs, also a new IAR.
Prior to issuing the SLS instruction, the microprogrammer must have loaded two specific registers of the current zone with the following information:
Register 14 (hex E) in current zone must contain the binary number of the new zone (0, 1, 2, or 3) in bits 2 and 3, and the IAR that is to have control in the new zone in bits 4 through 7. Any number from 0 through 7 can be specified as IAR in the new zone.
Register 15 (hex F) in current zone should contain the number of the current zone in bits 2 and 3, and the number of the current IAR or any IAR that is to be used when switch back to the current zone is desired in bits 4,5 and 6. However, register 15 need not necessarily contain the current zone, another zone may be specified if the switch back is to be to another zone.
The loading of register 14 and 15 is a prerequisite for the SLS instruction because the following actions occur when SLS is issued:

- The current IAR is updated by plus 1
- Bits 2 and 3 of LS-register 14 (E) in the current zone are loaded into LSZR. Bits 4,5 and 6 are loaded into the IAR Select Register
- The new zone is selected via LSZR
- The IAR Select Reg. points to the LS-Reg. pair in the new LS-zone that serves as IAR from now on
- The contents of register 14 (E) of the old zone are transferred to register 15 (F) of the new zone
- The contents of register 15 (F) of the old zone are transferred to register 14 (E) of the new zone

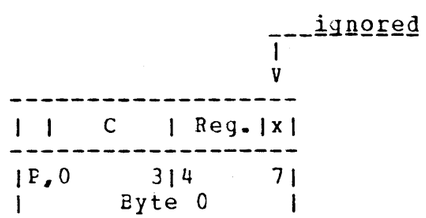
After this cross-over of old to new registers, register 14 (E) of the new

(now current) zone is the old zone recall register; while register 15 (F) of the new (now current) zone specifies the new zone for eventual recall. Thus by repeating SLS instructions, alternating zones can be selected. All register addresses in instructions refer to registers in the current zone cnly.

I-Fetch = 2 pico steps,
Execution = 9 pico steps.

Instruction address (IAR) is incremented by 1.

SST = Store one byte from Accu into storage

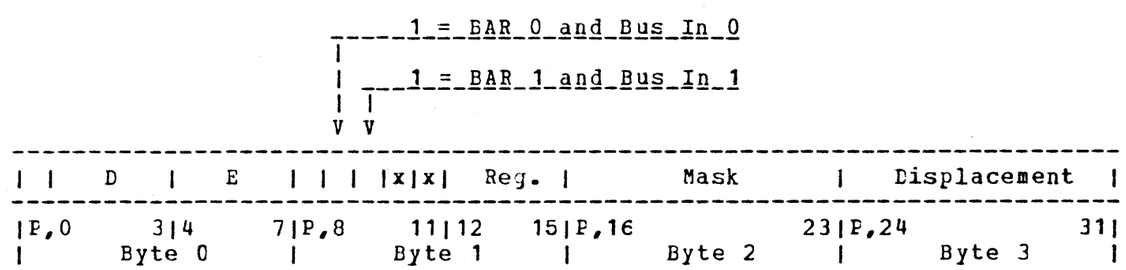


The contents of the accumulator are stored into the SVP storage. The 16-bit storage address is obtained from an adjacent pair (even+odd numbered) of LS registers. The even numbered LS register, containing the high order byte of the storage address, is defined by instruction bits 4,5 and 6. Instruction bit 7 is ignored. The contents of LSZR define the LS-zone. The storage address (in the LS-reg. pair) is automatically incremented by 1 via the ALU.

I-Fetch = 2 pico steps,
Execution = 3 picc steps.

Instruction address (IAR) is incremented by 1.

STBA = Sense, AND with mask into Accu, Branch if ALU zero.

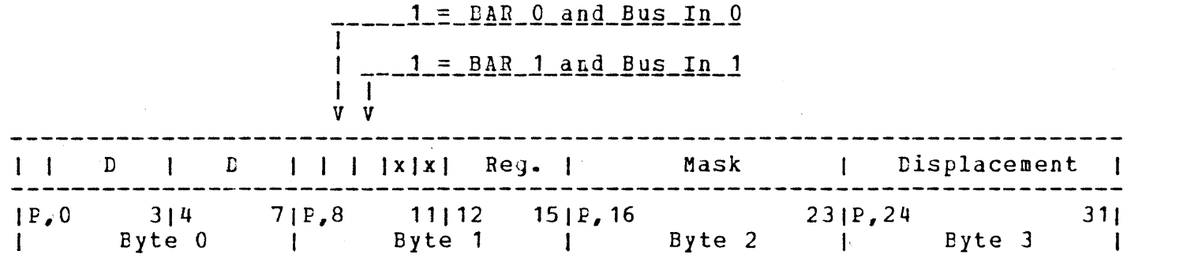


This instruction provides the means to address an external facility (outside SVP), to fetch the contents of this facility, and to logically AND this data with a mask in order to derive a branch decision in case of a zero result.
Instruction bits 12 through 15 specify an LS-Register, the contents of which are placed into either one of the two Bus-Address-Registers, whichever is specified by instruction bits 8 and 9.
The address placed onto the Bus causes the corresponding facility to set its data onto the inbound side of the SVP data ring bus. The data appears in the External In Reg. 0 or 1. From there it is logically ANDed with the mask (instruction byte 2) and the result is placed into the accumulator.
If the result is zero, the displacement (instruction byte 3) is placed into the low order byte of the current IAR, allowing a branch within a 256-byte block.

I-Fetch = 2 pico steps,
Execution = 9 pico steps if the branch does not take place,
11 pico steps if the branch takes place.

Instruction address (IAR) is incremented by 4 if the branch does not take place.

STBX = Sense, exclusively OR with mask into Accu, Branch if ALU zero.

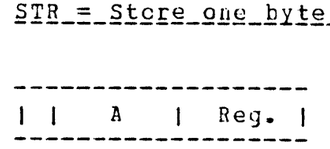


This instruction provides the means to address an external facility (outside SVP), to fetch the contents of this facility,and to exclusively OR (compare) this data with a mask in order to derive a branch decision in case of a match.
Instruction bits 12 through 15 specify an LS-Register, the contents of which are placed into either one of the two Bus-Address-Registers, whichever is specified by instruction bits 8 and 9.
The address placed onto the Bus causes the corresponding facility to set its data onto the inbound side of the SVP data ring bus. The data appears in the External In Reg. 0 or 1. From there it is exclusively ORed with the mask (instruction byte 2) and the result is placed into the accumulator.
If the result is zero, the displacement (instruction byte 3) is placed into the low order byte of the current IAR, allowing a branch within a 256-byte block.

I-Fetch = 2 pico steps,
Execution = 9 pico steps if the branch does not take place,
11 pico steps if the branch takes place.

Instruction address (IAR) is incremented by 4 if the branch does not take place.

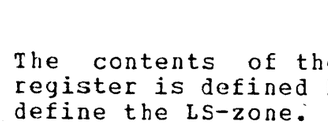
STOP = Halt Service Processor



The SVP stops after the fourth cycle (last execute cycle). After that the SVP can be started only via the IMPL key. The Stop instruction is used for diagnostic purposes.

I-Fetch = 2 pico steps,
Execution = 2 pico steps.

STR = Store one byte from Accu into LS-register

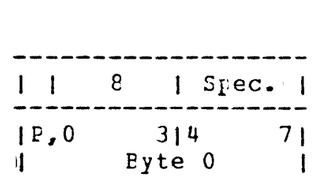


The contents of the accumulator are stored into any LS-register. The LS-register is defined by instruction bits 4 through 7. The contents of LSZR define the LS-zone.

I-Fetch = 2 pico steps,
Execution = 1 pico step.

Instruction address (IAR) is incremented by 1.

STROB = Sense from Bus 0 and/or 1 into Accu, or activate 'CTRL Strobe Bus 0 and/or 1'



This instruction provides the means to send out data previously set into one or both Bus Data Registers or to admit data from one or both Bus Input Registers into the accumulator, depending on the specification bits 4 through 7 of the instruction (see cases below). For both types of operation the appropriate Bus Address Reg. must have been loaded prior to the STROB operation. For the send operation (control) the loading of the Bus Data Register(s) is an additional prerequisite.
The STROB operation then controls:

- for sense (instruction bit 7 = off): The gating of data from the 'External In Bus(es) into the accumulator.
- for control (instruction bit 7 = on): The generation of 'CTRL Strobe Bus 0/1' which control the gating trom the Bus out to the external or internal unit.

When data is sensed from both buses simultaneously, the two bytes are ORed by the ALU and the result is placed into the acumulator.
The bus and the action (sense or control) are specified by instruction bits 4,5 and 7 as follows (bit 6 has no affect):

Bit 4...7 in hex.

4 or 6 = Sense 'External In Bus 1' into Accu

5 or 7 = Activate 'CTRL Strobe Bus 1'

8 or A = Sense 'External In Bus 0' into Accu

9 or B = Activate 'CTRL Strobe Bus 0'

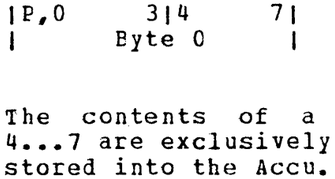
C or E = Sense 'External In Bus 0 and 1' (ORed) into Accu

D or F = Activate 'CTRL Strobe Bus 0 and 1'

I-Fetch = 2 pico steps,
Execution = 1 pico step.

Instruction address (IAR) is incremented by 1.

XOR = Exclusively OR LS-Reg. with Accu into Accu

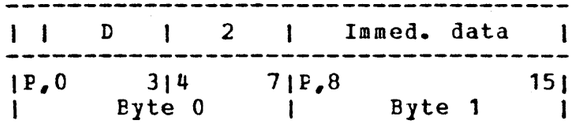


The contents of a local storage register addressed by instruction bits 4...7 are exclusively ORed with the contents of the Accu. The result is stored into the Accu. The LS-Zone is selected depending on the contents of LSZR. The 'Carry-Fl' is reset before and cannot be set by this operation. The 'ALU zero Fl' is set or reset depending on the result.

I-Fetch = 2 pico steps,
Execution = 1 pico step.

Instruction address (IAR) is incremented by 1.

XORI = Exclusively OR Accu with Immediate data byte into Accu



The contents of the Accu are exclusively ORed with the Immediate data byte (instruction byte 1). The result is stored into the Accu. The 'Carry Fl' is reset before and cannot be set by this operation. The 'ALU zero Fl' is set or reset depending on the result.

I-Fetch = 2 pico steps,
Execution = 4 pico steps.

Instruction address (IAR) is incremented by 2.

3125 Processing Unit
Microinstructions

Order No. SY33-1058-1

READER'S COMMENT FORM

Your views about this publication may help improve its usefulness; this form will be sent to the author's department for appropriate action. Using this form to request system assistance or additional publications will delay response, however. *For more direct handling of such requests, please contact your IBM representative or the IBM Branch Office serving your locality.*

Possible topics for comment are:

Clarity Accuracy Completeness Organization Index Figures Examples Legibility

What is your occupation? _____

Number of latest Technical Newsletter (if any) concerning this publication: _____

Please indicate in the space below if you wish a reply.

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

Cut or Fold Along Line

3125 Processing Unit
Microinstructions

Order No. SY33-1058-1

READER'S COMMENT FORM

Your views about this publication may help improve its usefulness; this form will be sent to the author's department for appropriate action. Using this form to request system assistance or additional publications will delay response, however. *For more direct handling of such requests, please contact your IBM representative or the IBM Branch Office serving your locality.*

Possible topics for comment are:

Clarity Accuracy Completeness Organization Index Figures Examples Legibility

What is your occupation? _____

Number of latest Technical Newsletter (if any) concerning this publication: _____

Please indicate in the space below if you wish a reply.

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

SY33-1058-1

Your comments, please . . .

This manual is part of a library that serves as a reference source for customer engineers. Your comments on the other side of this form will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Fold

Fold

First Class
Permit 40
Armonk
New York

Business Reply Mail
No postage stamp necessary if mailed in the U.S.A.

Postage will be paid by:

International Business Machines Corporation
Department 813B
1133 Westchester Avenue
White Plains, New York 10604

Fold

Fold

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

Cut or Fold Along Line

SY33-1058-1

Your comments, please . . .

This manual is part of a library that serves as a reference source for customer engineers. Your comments on the other side of this form will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Fold

Fold

First Class
Permit 40
Armonk
New York

Business Reply Mail
No postage stamp necessary if mailed in the U.S.A.

Postage will be paid by:

International Business Machines Corporation
Department 813B
1133 Westchester Avenue
White Plains, New York 10604

Fold

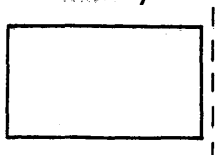
Fold

IBM

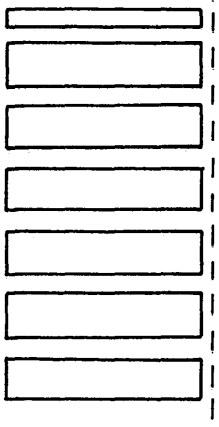
International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

System
Maintenance
Library



System



— — — cut here — —

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)