

**K M P D
ENGINEERING
PERFORMANCE
SPECIFICATION**

UNCLASSIFIED

SAC - 11

Draft - 2

**PROGRAM INSTRUCTION SPECIFICATIONS
FOR THE STRATEGIC AIR COMMAND DATA
PROCESSING SYSTEM**

by

JOHN REILLY

L. G. Sellers

Corrections

added Feb 3, '59

Approved by:

Approved by:

December 8, 1958
MILITARY PRODUCTS DIVISION

INTERNATIONAL BUSINESS MACHINES CORPORATION

KINGSTON, NEW YORK

The work reported in this document was performed under a government contract. Information contained herein is of a proprietary nature. ALL INFORMATION CONTAINED HEREIN SHALL BE KEPT IN CONFIDENCE. No information shall be divulged to persons other than IBM employees authorized by the nature of their duties to receive such information or individuals or organizations who are authorized in writing by the Department of Engineering or its appointees to receive such information. GOVERNMENT RELEASE MUST BE OBTAINED THROUGH THE IBM PATENT DEPARTMENT BEFORE THIS INFORMATION MAY BE USED FOR COMMERCIAL APPLICATIONS.

UNCLASSIFIED

CONTENTS

<u>Heading</u>		<u>Page</u>
Section 1	Introduction	1
Section 2	Addressing	1
2.1	Direct Addressing	1
2.2	Immediate Addressing	1
2.3	Indirect Addressing	2
2.4	Indexing	2
2.5	Addressable Registers	2
Section 3	Configurations	3
3.1	Full Word	3
3.2	Dual Word	3
3.3	Bytes	4
Section 4	Instructions	5
4.1	Load Class	5
4.2	Arithmetic Class	11
4.3	Floating Point	16
4.4	Store Class	19
4.5	Shift Class	23
4.6	Reset Class	27
4.7	Branch Class	29
4.8	Logical Class	36
4.9	Convert Class	41
4.10	Miscellaneous Class	43
4.11)	High Speed I/O Operation	47
4.12)	Low Speed I/O Operation	51

2.4.1 Direct Addressing
 2.4.2 Route Addressing
 4.11 I/O operations (combined)

pencil
 corrections added
 Feb 3, 1958

PREFACE

Abstract:

**This document describes the basic instruction list for the
Strategic Air Command Data Processing System.**

1.0 INTRODUCTION

The SACCS computer word is a binary 48 bit word with negative numbers expressed in one's complement form. For arithmetic processes, the word can be treated as a single sign-bit-plus-47-bit word, or as a pair of sign-bit-plus-23-bit "half-words."

For logical or arithmetic purposes, selected sets of 6 consecutive bits (called "bytes") can be operated upon. There are eight of these bytes to a word. Some logical operations can treat less than 6 bits at a time. The

instruction word takes several forms, but all forms include 4 bits for index

^{seven} selection and eight bits to specify the basic instruction. *Only those instructions (BXH, BXE, BSX, etc.) using the decrement field (DF) require less than 7 bits. (They require 6.)*

2.0 ADDRESSING

Initially 32,768 or 2^{15} registers of core memory shall be provided in the SACCS machine, ~~requiring 16 bits for addressing.~~ Since 18 bits have been allocated for addressing, room has been provided to expand memory.

Special modifiers or address codes will be used to address the internal registers.

2.1 Direct Addressing

The "direct" method of addressing is to use the right-most 18 bits of an instruction to specify the address of the core memory (or internal), register which contains the operand, next instruction, etc.

2.2 Immediate Addressing

"Immediate" or "real-data," addressing is a method in which the right-most 24 bits act as the operand.

2.3 Indirect Addressing

"Indirect" addressing is a method in which the right-most 18 bits of the word specify the address of a word containing the operand address, or branch address. Multi-level "indirect" addressing shall be permissible.

(Order is managed here)

2.4 Indexing

In all of the above addressing schemes, it is possible to change these addresses by a number contained in an 18 bit register called an "index register." There are 4 (expandable to 13), of these index registers, and the right accumulator and program counter may also be used in this manner. Four bits of the instruction word are used to specify the index register to be used in this process. One bit of the instruction word is used to specify whether the mode of indexing will be direct or indirect.

2.4.1 Direct Indexing

In this mode of indexing, the contents of the specified index register are added to the right-most 18 bits of the instruction word. The resulting address shall hereafter be referred to as an "effective address." With indirect addressing, multiple indexing is possible. That is, an index register can be specified and used in the instruction word, and the same index register (or another index register) can be used in the addressed word which contains the operand address (or branch address).

2.4.2 Indirect Indexing

left out

In this mode of indexing, the index selection bits (IX_0) contained in the instruction word (I_0) specify the index register whose contents specify the address of a second word I_1 . The address portion of I_1 (A_1), is added to the address portion (A_0) of the original instruction word, and the indirect index tag of I_1 is examined. If this tag is set, the index selection bits of I_1 specify the index register whose contents specify a memory address for a third word I_2 .

The process described above will continue until the indirect index tag of a word I_n is found to be zero. At this time an address $A_0 + A_1 + A_2 + \dots + A_n$ plus the contents of IX_n is formed. This address will specify the address of an operand, or the address of a word containing the operand address depending upon the value of the indirect address tag of the word I_n . Once the process of indirect indexing has been completed (i. e., the I_n th word), no further reference to the indirect indexing tag shall be made for the duration of the instruction.

2.5 Addressable Registers

Many "internal" registers of the central computer and peripheral unit processor shall be addressable. These shall include the following:

- (1) Test memory (*control panel, switch registers, logical reg. plugboard, toggle switch, live*)
- (2) Index registers (4)
- (3) Accumulator
- (4) B register
- (5) Program register

(6) ~~Logical register~~

(7) Clock register (X) (real time)

(8) Alarm registers (~~master alarm, specific-alarm, alarm-mask~~)

(9) I/O counters (word, address, ^{High speed} angular-position)

(10) ~~DI selection register~~

The first ^{four} six of these can be read, set, or used to increment the BAR

(Branch to Address plus Register) ^(BAR) instruction ~~It will be possible to~~

~~branch to test memory.~~ The clock and ^{I/O} alarm registers can ^{only} be read, or

~~set, and the I/O counters can be only read.~~ The read & set status of the

alarm regs, is specified either in referne A-1 (SAC-51) or referne A-2 (SAC-52). The

3.0 CONFIGURATIONS

Various configurations are possible with the SACCS word. These include

full-word, dual-(or half-) word, and byte configurations.

3.1 Full Word

The full word shall consist of a sign-bit (0 = +, 1 = -) and 47 magnitude bits in one's complement form (i. e., the negative of a number shall be obtained by complementing all 48 bits). In Multiplication and Division, the full-word product and dividend shall occupy 96 bits, the most significant half having its sign as the sign-bit of the left accumulator and its magnitude in the remaining 47 bits of the accumulators, and the least significant half having an identical sign in the sign bit of the left B register and its magnitude in the remaining 47 bits of the B register.

3.2 Dual Words

Dual words shall denote pairs of half-words. Half-words shall consist of a sign bit and 23 magnitude bits in one's complement form, similar to the full-word form. Again, in Multiplication and Division, the half-word product and dividend shall occupy 48 bits of the left-accumulator/

left-B-register, or right-accumulator/right-B-register, having its sign in the sign bits of the appropriate accumulator and B register, and its magnitude in the remaining 46 bits of these half-registers. As with the full word, the most significant half is in the accumulator.

3.3 Bytes

A memory word can be considered to be composed of eight 6-bit "bytes." These can, by byte-configuration control, be brought into, or stored from, the arithmetic element, displaced by any number of byte positions from 0 to 7. Up to eight bytes may be loaded into, or stored from, bytes within the accumulator or B-register word. Byte displacement is on a cyclic basis. For example, displacement by 4 byte positions sends byte 0 to (or from) byte 4, and 1 to 5, but sends byte 4 to byte 0. Byte-configuration control takes place between a register called the "Exchange" register and the "A" register. Transfer paths are set up from the former to the latter. Loading and adding involve the transfer of the memory word to the Exchange register, and the active bytes along the proper path to the A register, and thence directly to the proper register in the arithmetic element. Inactive bytes may be set in the A register. Storing involves the placing of the memory word directly in the A register, and modifying it there via the transfer paths from the Exchange register. The A register is then restored to memory. There are some special byte configurations, used for "twin" operations.

These are configurations in which byte paths which go to one A register half-word also go to the other. In the "twin" mode with no byte displacement, transfer paths are set up as described above except that the left Exchange-register bytes are connected to both A registers.

omitted
?

4.0 INSTRUCTIONS

The SACCS instructions shall be divided into ten classes: Load, Arithmetic, Store, Shift, Branch, Convert, Logical, Reset, Miscellaneous, and In/Out.

The word layout, modifier description, and a general description of the class are given with each class of instructions. The list of each class will contain the mnemonic codes of each instruction with a brief description of the instruction action, a word layout, and modifier description, if different from those of the instruction class.

4.1 Load Class Instructions

OP	R	P	M	T	A	RI	I	I _x	Addr				
5-6	8	9-11	12-14	15	16-23	24	25	26-29	30-47				
Op	P	M	A	T	R	X ₁	I	I _x	Addr.				
S-7	8	10	11	13	14	21	22	23	24	25	26-29	30	47

All instructions in this class are indexable.

Modifiers

For all load class instructions, the Exchange and A registers are initially cleared. The addressed memory word is then loaded into the Exchange registers (half-word addressable machine registers are loaded into the right exchange registers; the six-bit auxiliary logical register is loaded into the right-most byte of the exchange register). Configuration control is applied in the Exchange register - to - A register transfer according

to the following scheme:

A. Byte Activity - This modifier is an eight bit mask and is applied to either or both A registers. It specifies which bytes are loaded from exchange register bytes. Bits ~~14~~¹⁶, ~~15~~¹⁷, ~~21~~²³ correspond to the first, second, eighth A register bytes. For each bit:

0 - Corresponding A register bytes are loaded from the Exchange register.

1 - Corresponding A register bytes not loaded.

P Byte Displacement - This three bit tag specifies byte positioning between the exchange and A register bytes. The binary code specifies the number of byte positions each Exchange register byte is displaced to the right as it is transferred to the A register. (The eight bytes of a full register are considered to form a ring - that is, bytes are displaced from byte position 7 to byte position 0). Byte displacement in the twin mode is essentially the same as above.

M Mode Selection - This modifier specifies the arithmetic mode in which the instruction operation takes place. The binary codes are interpreted as follows:

000-~~011~~ Full Word - For this mode of operation, there is one arithmetic element. The combined left and right accumulators (A registers, B registers, etc.) are considered to form a single register containing a signed quantity with 47 bits of magnitude.

100 Twin Mode - The instruction is executed similar to the dual mode. In this mode, the left exchange register is transferred to both the left & right A reg. The byte activity (modified A) will determine the A reg. bytes that are set. The P modifier is not referenced.

It should be noted that byte activity (A mod.) must be identical for each half word to arrive at a true twin operation.

101 - Reverse Twin Mode - This mode is analogous to the Twin mode with the exception that the right exchange reg. will be transferred to both the left & right A registers.

011

~~100~~ Dual Mode - The instruction is executed independently in both arithmetic elements.

001

~~101~~ Left Mode - The instruction is executed only in the left arithmetic element.

010

~~110~~ Right Mode - The instruction is executed only in the right arithmetic element.

100

~~111~~ Twin Mode - A load class instruction will be executed in the same manner as in the Dual Mode, except that at the completion of the normal dual mode operation, a +4 byte displacement will be made in addition to the transfers already performed.

For example, if a load class instruction is given, specifying twin mode, an A modifier of ^{1101 1101} 11011110, and P modifier of 010, transfers would occur according to the following diagram:

	Byte	0	1	2	3	4	5	6	7
Exchange Register		1		X			1		

A Register

	Byte	0	1	2	3	4	5	6	7
				X	1			X	1

This is arrived at in the following manner:

The A modifier indicates that bytes 2 and 7 of the A register will be active. The P modifier indicates that the information comes from bytes 0 and 5 of the Exchange register. Finally, the +4 byte displacement takes place, transferring (non-destructively) byte 2 to byte 6 and byte 7 to byte 3.

~~Spurious results may be obtained if corresponding bytes in the left and right half word (0 and 4, 2 and 6, etc.) are selected, or if a P modifier is chosen such that information would be transferred from one half-word to the other.~~

T- Signed Data Tag

After the byte-transfer from Exchange to A registers have been made, this tag is examined. If it is set, then for each (active) A register:

The computer looks at the first bit of the left-most byte that received data from an Exchange register. If this bit is a "1", then all bytes that did not receive Exchange register data are set to "1". That is, all six bits in each of these bytes are set. If the first bit of the left-most byte is a "0", or if the T tag is not set, no action is taken upon the A registers.

R- Real Data Tag

Real data tag indicates whether the address portion of the word is to be used as real data or a constant:

A "0" indicates the address portion refers to a memory location.

A "1" indicates the address portion is to be used as a constant.

When this tag is set the right half of the instruction word (bits 29-47) will be transferred to both the left & right exchange registers. A "1" indicates the right half of the instr. word is to be used as a constant, not just the address portion

I- Indirect Address Tag

When a "1" bit is present in this bit position the addressing mode will be indirect. If the bit in this position is "0" the addressing mode is direct.

~~X~~ Indirect Index Tag

~~X~~ When a "1" bit is present in this position the indexing mode will be indirect. If the bit in this position is "0" the indexing mode is direct.

DI. Double Index Tag - When this bit is set "1" the mode of indexing shall be "double". If the bit is not set "0" the indexing mode is direct.

LDM Load Magnitude. The contents of the acc. are replaced by the abs. value of the contents of the Mem. location - after configuration into the A reg. - specified by the address portion of the instr. It is actually the abs. value of the resultant A reg(s), after configuration control is completed, that is loaded into the accumulator. Consequently the T modifier could have effect in determining the abs. value of the A reg.

Example: LDM (burl) A 110010, P=001

Byte	0	1	2	3	4	5	6	7	
Exch. Rd	A	B	X	Y	C	D	E	F	
A Reg.				X	Y				

when T=1, the sign of X is 1, Y is 0

LMC. The contents of the accumulator are replaced by the ones complement of the absolute value of the contents of the memory location - after configuration into the A reg. - specified by the address portion of the instruction. Again, it is actually the ones complement of the absolute value of the resultant A reg(s) - after configuration control is completed - that is loaded into the accumulator. This instr. is analogous to LDM.

LOAD CLASS INSTRUCTIONS

LAC Load Accumulators

The contents of the accumulator are replaced by the contents of the memory location specified by the address portion of the instruction.

LBR Load B Registers

The contents of the B register are replaced by the contents of the memory location specified by the address portion of the instruction.

LBC Load B Registers Complement

The contents of the B register are replaced by the complement of the contents of the memory location specified by the address portion of the instruction.

LDM Load Magnitude

The contents of the accumulator are replaced by the absolute value of the contents of the memory location specified by the address portion of the instruction.

LMC Load Magnitude Complement

The contents of the accumulator are replaced by the one's complement of the absolute value of the contents of the memory location specified by the address portion of the instruction.

LCP Load Complement

The contents of the accumulator are replaced by the one's complement of the contents of the memory location specified by the address portion of the instruction.

LDL Load Logical Register

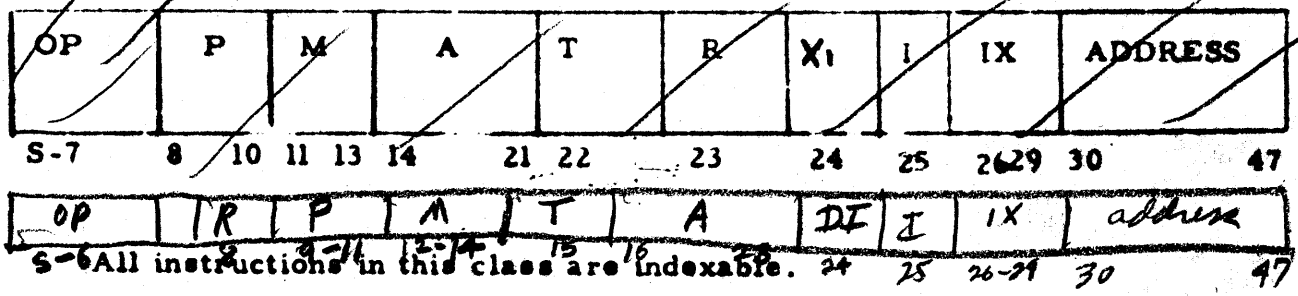
The contents of the logical register are replaced by the contents of the memory location specified by the address portion of the instruction.

LDI Load Double Index Selection Register

OP	R	B ₁	DI	I	IX	address
0-6	8	9-11	24	25	26-29	30-47

B₁ specifies the mem. byte to be loaded into the DI selection register. only the right 4 bits will be loaded

4.2 Arithmetic Class Instructions



Modifiers

The above format is used with all Arithmetic Class Instructions.

The modifiers are identical to those used with Load Class Instructions and yield the same results. That is, they serve to determine the configuration of data which arrives at the A register(s), and the arithmetic elements in which instruction operation takes place.

ARITHMETIC INSTRUCTIONS

ADD Add

This instruction algebraically adds the contents of the specified memory location to the contents of the accumulator and replaces the contents of the accumulator with the resulting sum.

SUB Subtract

This instruction algebraically subtracts the contents of the specified memory location from the contents of the accumulator and replaces the contents of the accumulator with the resulting difference.

ADM Add Magnitude

This instruction algebraically adds the magnitude of the contents of the specified memory location to the contents of the accumulator and replaces the contents of the accumulator with the resulting sum. The sign of the contents of the memory location is ignored.

SBM Subtract Magnitude

This instruction algebraically subtracts the magnitude of the contents of the specified memory location from the contents of the accumulator and replaces the contents of the accumulator with the resulting difference. The instruction is performed by first complementing the contents of the specified memory location, if positive, and then proceeding as in addition.

DIM

Difference Magnitude

The absolute value of the contents of the specified memory location are subtracted from the absolute value of the contents of the accumulator. The instruction proceeds as does the ADD instruction except that, first the specified memory location is complemented, if positive; the accumulator is complemented, if negative.

MUL

Multiply

Dual, Left or Right:

The contents of the left accumulator are multiplied by the contents of the left memory register and/or similarly for the right accumulator. At the conclusion of the multiply: the accumulator bits 1-23 contain the 23 most significant bits of the product, the B register bits 1-23 contain the 23 least significant bits of the product, the accumulator sign and B register sign contain the algebraic sign of the product.

Full Word:

The full accumulator and the full contents of the specified memory location, treated as signed, 47-bit magnitudes, are multiplied. At the conclusion of the multiply: the accumulator bits 1-47 contain the 47 most significant bits of the product, The B register bits 1-47 contain the 47 least significant bits of the product, the accumulator (left) sign and B register (left) sign contain the algebraic sign of the product.

MPR **Multiply and Round**

This instruction executes a multiply followed by a round.

DVD **Divide**

Full Word:

The 94-bit dividend is placed in full accumulator bits 1-47 and B-register bits 1-47. The left accumulator sign bit and left B-register sign bit contain the algebraic sign of the dividend. The divisor is treated as a signed quantity with 47 bits of magnitude. The sign plus 47 bits of the quotient appear in the accumulators; the sign plus 47 bits of the remainder appear in the B-registers.

Dual, Left or Right: (Combined)

The contents of the left Accumulator/left B register are divided by the contents of the left memory register and/or similarly for the right Accumulator/B register and right memory register. Before executing a Divide instruction, the dividend must be placed in the combined accumulator/B register. It takes the same form as the product of a Multiply instruction. That is, the dividend sign is loaded into both accumulator and B-Register sign position; the 23 most significant bits and the 23 least significant bits of the dividend are loaded, respectively, into accumulator bits 1-23 and B-register bits 1-23. The divisor, which is taken from the specified memory location, is a signed quantity with 23 bits of magnitude. At the conclusion of the divide operation, the accumulator contains the

sign and 23 bit quotient. The B register contains the sign and 23 bit remainder. The sign of the remainder is identical to that of the quotient.

NOTE: Prior to division, the ACC-B Reg. sign will be compared. If unlike, the B reg. will be complemented to conform to the sign of the accumulator.

A test for illegal division will be made prior to beginning divide (illegal is $| \text{divisor} | \neq | \text{dividend} |$). A signal to the interrupt system will be generated if illegal conditions occur.

The instr. word layout and modifier description is given below:

op		UN		DI	I	IX	addr.
3-6		8		24	25	26-29	30 - 47

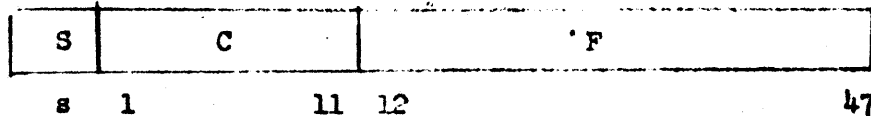
UN un-normalized. applies only to FAC, FSB, FMP.

"0" - the instr. shall be normalized prior to completion

"1" - No attempt shall be made to normalize at the completion of the instruction

4.3 Floating Point Operations

A floating point number shall be represented by a quantity $2^n \times$ a fractional quantity (F). The number shall be stored in a register as shown below.



S - The sign of F is in the S position of the word.

F - The magnitude of F is in bit positions 12-47. A negative fraction is expressed in ones complement form. A floating point binary number is said to be normalized when $1/2 \leq |F| < 1$.

C - The characteristic of a number is stored in positions 1-11. The characteristic of the fraction is formed by adding 102^4 to the exponent n. The characteristic of a negative number shall be expressed in one's complement form.

The modified bits (of the left half word) described in previous sections shall have no meaning in floating point instructions. Unless, otherwise noted, all floating point instructions are normalized prior to completion. All instructions in this class are indexable.

FLT - Float

This instruction will operate on non-floating words within the accumulator. The fraction will be shifted until it occupies the last 36 bit positions of the accumulator with the remaining bits in bit positions 12 thru 22 of the B Register. The right most 11 bits of the contents of the effective address shall be deposited in bits 1 thru 11 of the accumulator. The resultant floating point number shall then be normalized and the characteristic suitably

FRN - Floating Round

If position 11 of the B Register contains a "1", the magnitude of the fraction in the accumulator is increased by adding a "1" into position 47. The contents of the B Register are unchanged. For the special case where the contents of the accumulator 12-47 are all ones and a carry is propagated from position 12 into position 11 of the accumulator, a "1" is inserted into position 12 of the accumulator.

FAD Floating Add

The contents of the specified memory location are algebraically added to the contents of the accumulator according to the rules of floating point arithmetic and the resulting sum replaces the contents of the accumulator and the B register.

FAM Floating Add Magnitude

Same as floating add except the magnitude of the contents of the specified memory location is used.

FSB Floating Subtract

Same as floating add except the negative of the contents of the specified memory location is used.

FSM Floating Subtract Magnitude

Same as floating add except the negative of the magnitude of the specified memory location is used

FMP Floating Multiply

The contents of the specified memory location are multiplied by the contents of the accumulator. The most significant part of the product appears in the accumulator, and the least significant part appears in the B register. The product of two normalized floating point numbers is in normalized form. ^{if $\frac{1}{2} \leq |P| < 1$} If either the multiplier or multiplicand is not normalized, the product may or may not be in normalized form.

FDV Floating Divide

The contents of the accumulator are divided by the contents of the specified memory location. The quotient appears in the Accumulator and the remainder appears in the B Register. A test for illegal division

will be made prior to divide (a successful divide check implies that $0 < Q_F < 2$). A signal to the interrupt system will be generated if illegal conditions occur.

Note: An operate instruction will be provided to enable the program to set the B Register to ± 0 to conform to the sign of the accumulator.

~~UFA Un-normalized Floating Add~~

~~This instruction is identical to FAG except that no attempt will be made to normalize at the completion of the add.~~

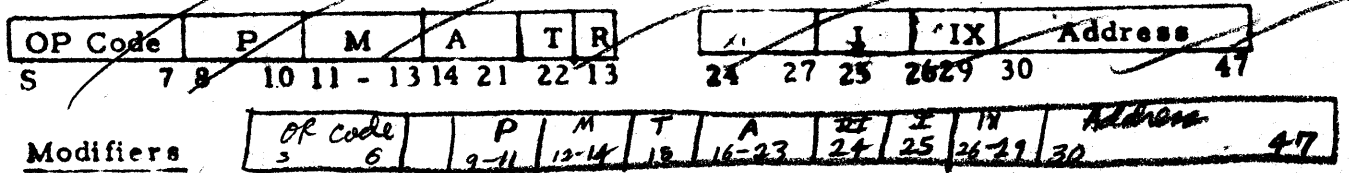
~~UFS Un-normalized Floating Subtract~~

~~This instruction is identical to FSB except that no attempt will be made to normalize at the completion of the operation.~~

~~UFM Un-normalized Floating Multiply~~

~~This instruction is identical to FMP except that no attempt will be made to normalize at the completion of the operation.~~

4.4 Store Class Instructions



T This modifier is used as in the Load and Arithmetic instructions for a signed data tag; however, it applies only to the ADR and ATR instructions.

A, P These modifiers are analogous to the A and P modifiers used with Load and Arithmetic instructions. They are presented here as they apply to the simple store instructions (SAC, ^{STL}STB). Their use with the SPR, AOR, ^{SOR} and ATR instructions is defined separately with each of these instructions.

At the beginning of a store class instruction, the contents of the addressed memory location are loaded into the exchange register and transferred to the A register. During this portion of the instruction execution, only those bytes specified not active (i.e., containing a "1" bit in the A modifier) are transferred, and the P modifier is not referenced. The contents of the accumulator are then transferred to the exchange register. The A modifiers now specify which A register bytes are to receive information (i.e., active bytes are specified by 0's). The P modifier specifies the byte displacement as previously described. The contents of the A registers are then transferred to memory. The A register may be considered as an assembly register for the word to be returned to memory.

Thus, on store class instructions the A field specifies those memory bytes which are to receive information. Where the instruction LAC, A - 11111011, P - 010, M - 000 brings memory byte 3 into accumulator byte 5, the instruction SAC, A - 11111011, P - 010, M - 000 stores accumulator byte 3 into memory byte 5.

Real Data Tag - has no meaning

M mode - here meaning when applied to AOR, SOR, ATR instrs. Twin reverse twin mode shall not be used

page 70 missing?
SAC, STB, SPR

AOR Add One to Register

All values of the M modifier, except for twin, shall be legal for this instruction and shall be interpreted as previously described. The A registers are loaded with memory bytes such that the active byte(s) in the exchange register (specified by "0" in the A field) is transferred to the A register. The P modifier during this portion of the instruction specifies the number of byte positions each exchange register byte is displaced to the right as it is transferred to the A registers. A one is added to bit 23 of the left/or right adder (dual mode), or to bit 47 (full word); the results appear in each accumulator (dual) or in the combined accumulator (full words). The accumulator(s) are then stored into memory in a manner analogous to the SAC instruction except that during this portion of the instruction the P modifiers specify the number of byte positions each exchange register byte is displaced to the left as it is transferred to the A register.

Either or both half word accumulators or the combined 48-bit accumulator are changed by this instruction. The T modifier, if used, will act as previously described; however, the inactive memory bytes within the specified memory location will remain undisturbed after completion of the instruction.

SOR Subtract One from Register

This instruction is analogous to AOR except that a one is subtracted from the active byte(s).

ATR

Add to Register

Modifiers are interpreted as with the AOR instruction. The A registers are loaded with memory bytes as with the LAC instruction. The A registers and accumulator(s) are then added. The M modifiers will govern the adder connections (i. e., dual, dual left, dual right, or full). The accumulator(s) are then stored into memory as with the AOR instruction.

ECH

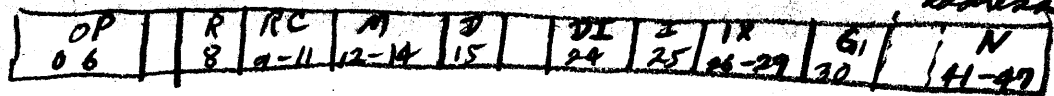
Exchange

This instruction exchanges selected memory and accumulator byte(s) without disturbing inactive bytes in either register. The instruction ECH, A--11111101, P--1 will exchange memory byte 6 with accumulator byte 7.

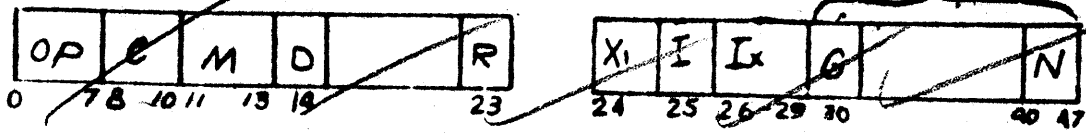
STL

Store Logical Register

The contents of the addressed memory locations are transferred to the A registers. The contents of the logical register are transferred to the Exchange registers. Selected exchange register bytes replace A register bytes without disturbing non-active byte position. The A registers are then transferred to the addressed memory location.



4.5 Shift Class Instructions



All Shift Class instructions are indexable

Modifiers:

R: Real Data. This bit applies to SFT and CYC only. If this bit is set, the shift count (number of bits to shift) is taken from the effective address. If it is not set, the effective address specifies a memory word whose effective address in turn specifies the shift count.

I: Indirect Address Tag. When a "1" bit is present in this position, the addressing mode will be indirect. If the bit in this position is "0" the addressing mode is direct.

DI: Double index tag

~~X₁ Indirect Index Tag. When a "1" bit is present in this position, the indexing mode will be indirect. If the bit in this position is "0" the indexing mode is direct.~~

M: Mode. This modifier specifies in which arithmetic elements shifting will take place. "Full" also specifies a connective or connectives. The left element bit 23 is connected to the right element sign bit, and shifting takes place through the right sign bit. The codes are as follows:

000	000-011	Full
011	100	Dual Mode
001	101	Left Mode
010	110	Right Mode

RC ~~A~~: Connectives. These bits specify connectives or variations for both cycle and shift instructions.

000	Accumulator/B register
001	Accumulator(s) only
010	B-Register(s) only
100 011	Invert

The last code, Invert specifies that the accumulator(s) are cycled in one direction (as specified by G, below) and the B register(s) in the opposite direction. This code is legal only for CYC. The sign bit of the active accumulator(s) (as specified by M) is connected to the sign bit of the active B register(s), and the least significant bit of the active accumulator(s) is connected to the least significant bit of the active B register(s).

D: Round. If this bit is set, rounding is performed; that is, if the most significant magnitude bit of the appropriate B register(s) differs from the sign of the B register(s), the absolute value of the contents of the accumulator(s) is increased by one.

The B register(s) remains unchanged. This modifier affects only NOR and SFT.

G: Direction of Shift (Cycles). This modifier which is used only in SFT and CYC, specifies in which direction the shifting or cycling is to take place. It is the most significant bit of the effective address. The codes are as follows:

1 ~~0~~ Left

0 ~~1~~ Right

N: Number of Shifts (cycles). This modifier (the last seven bits of the effective address) specifies the shift count. When G (the most significant bit of the effective address) is set the N bits specify the ones complement of the number of shifts to be performed.

Instructions

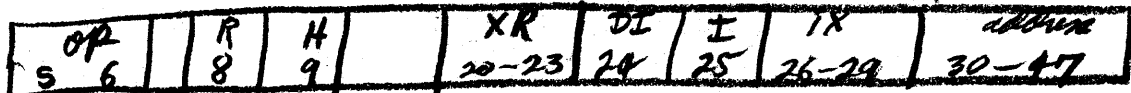
SFT: Shift. The designated 24, 48 or 96 bit number is shifted to the left or right, as specified. Sign bits (see modifiers) are bypassed. Bits shifted out one end of the word are lost; bit positions which are vacated are filled with the value of the sign bit.

CYC: Cycle. This is similar to SFT, except that sign bits are not bypassed and vacated bit positions at one end of an element are

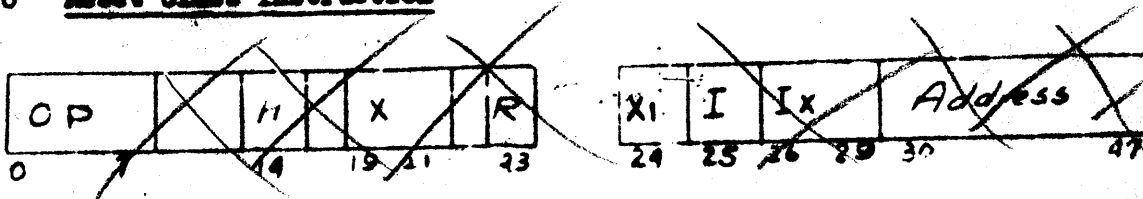
filled with the bits which are cycled out of the other end.

NOR: Normalize. For this instruction, the C modifier must have the value 00, and the R and G modifiers are meaningless. The contents of the specified accumulator or accumulator B-register configuration are shifted left until bit one of the left (or right, if M = 100 or 110) accumulator is different from the left (right) sign bit. That is, the shifting continues until the absolute value of the number being shifted is $\geq 1/2$ (the binary point being immediately to the right of the sign bit). The number of shifts is stored in the right-half-word of the memory register specified by the effective address.

If the D modifier is set, rounding is performed, provided the C modifier is = 000. If rounding causes overflow, ^{an overflow} ~~the com-~~puter shall correct this condition. ~~alarm shall be generated,~~ The vacated bit positions shall be filled with the value of the sign bit.



4.6 Reset Class Instruction



All instructions in this class are indexable

OP, R, IX, ~~X~~^{DI}, I, Addr. have the meaning previously specified.

H Half word modifier - the modifier specifies the half word of the operand to be used in the execution of the instruction. The modifier shall have the following meaning:

0 - Right half memory word is used

1 - Left half memory word is used.

The H modifier shall have no meaning when the R modifier is set

X Index Register Selection - This modifier refers to the index register which is modified or stored.

LIX Load Index Register

If the R modifier is set, the contents of the address portion of the instruction is loaded into the index register specified by XR. *real data base. The addressing will be done in the normal manner.* If ~~R~~ is not set, ~~the contents of the right-most 18 bits of the half word indicated by the H modifier of the specified memory location is loaded into the index register specified by X.~~

ALX Add to Index Register

Same as Load Index Register except that the selected information is added to the contents of the specified index register instead of replacing it.

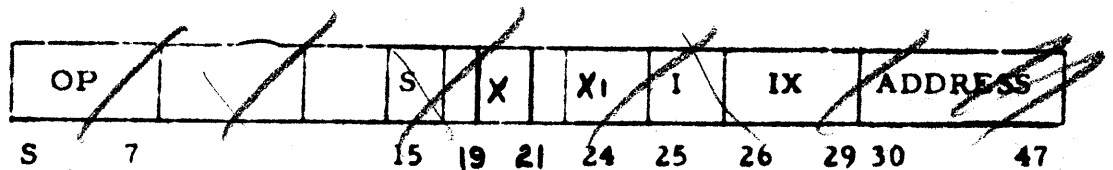
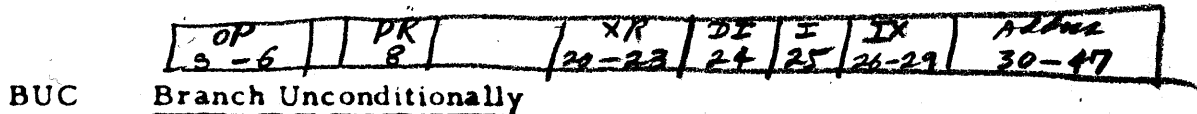
SIX Store Index Register

The contents of the index register specified by modifier "X" is deposited in the right-most 18 bits (half word specified by H) of the memory location specified by the effective address of the instruction. The R modifier is not interpreted.

4.7 Branch Class Instructions

Branch Class

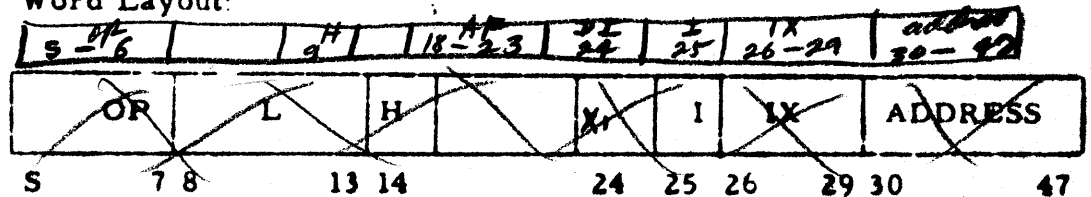
All branch-class instructions are indexable unless otherwise specified. All, except BUC, always set the Program Register from the Program Counter and the Program Counter from the effective address, provided the branch is "active."



Branching is performed unconditionally, but the S modifier of the instructions, if set to "one," indicates that the Program register is not set from the Program Counter. The index register specified in the X modifier is set from the Program Counter (specifying index register "zero" nullifies this action).

BAR Branch to Address Plus Register

Word Layout:

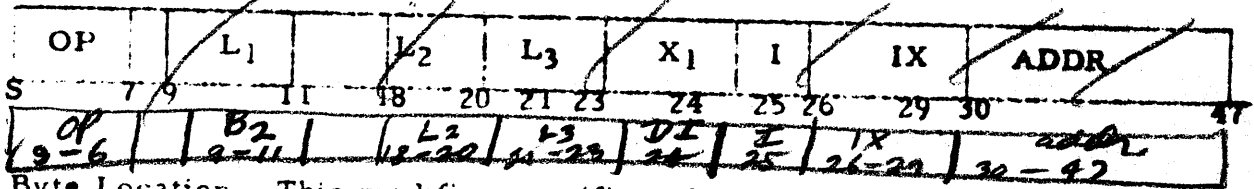


Branching is unconditional, and the Program register is set from the Program Counter. The branch address (address to which the Program Counter is set) is calculated by adding to the effective address the contents of the right most 18 bits of the internal register whose address relative to 777600_8 is

given by the L modifier. That is, the address of the internal register is calculated by adding (777 600)₈ to the contents of the L modifier. If the internal register is a full 48 bit register, the H modifier specifies the half word used.

H codes are 1 - left, 0 - right.

BAL Branch to Address Plus Logical Register



~~X~~
B₂

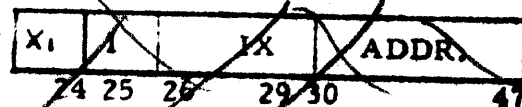
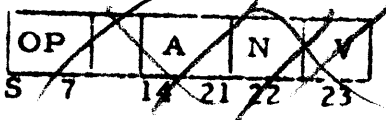
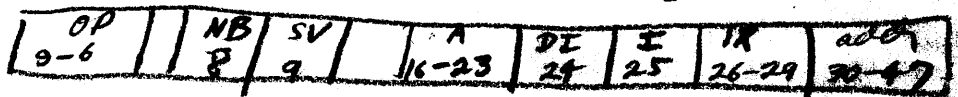
Byte Location - This modifier specifies which byte of the logical register is to be utilized by the instruction.

L₂ First Bit - This modifier specifies the active first bit of the auxiliary logical register. 0 - 5 are legal values, 6 and 7 are illegal.

L₃ Number of Shifts - This modifier specifies the number of bit positions the auxiliary logical register is shifted to the right. *The value of L₂ + L₃ must fall between 1 ≤ L₂ + L₃ ≤ 6. If the sum equals any other value, the result is an unconditional branch to the effective address.* Branching is unconditional, and the contents of the Program Counter are

stored in the Program register. The branch address is calculated by adding to the instructions effective address to the selected bits of the Logical register.

BSG Branch on Sign



Modifiers

A: Active - This modifier specifies the active bytes of the accumulator.

NB, Branch on condition not met - This bit is used for conditional branches, and, when set, causes the branch to take place only when the condition for branching is not met.

SV: Variation - This bit specifies the condition for branching.

The codes are as follows:

0 - Condition met if all active signs are plus.

1 - Condition met if all active signs are negative.

This instruction and its variations (**NB**^{SV} and **A**) provide the facility to branch if any, or all, of any set of accumulator bytes are positive, or negative. (That is, if left most bit of byte is "0" or "1" respectively). If the branch is active, the Program register is set from the Program Counter, and the Program Counter is set from the instructions effective address. Otherwise, the next instruction is taken in sequence.

BOZ

OP	NB	Z	A	PI	I	IX	ADDR
5-6	8	10-11	16-23	24	25	26-29	30-47

OP	Z	A	N	V	X1	I	IX	ADDRESS				
5	7	12-13	14	21	22	23	24	25	26	29	20	47

Modifiers

Z - Zero Definition - Zero, as used in the V variation is defined by Z as follows:

00 "Zero" means "+0" only

01 "Zero" means "-0" only

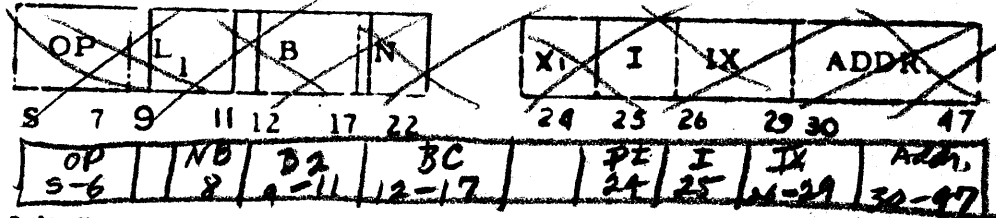
10-11 "Zero" means "+0" (all active bytes having same sign)

NB

A, **N**, **V** have the meanings as described in the BSG instruction.

This instruction and its variations (N, V, A and Z) provide the facility to branch if any, or all, of any set of accumulator bytes are "zero" or "non-zero," where "zero" is defined by the Z modifier. If the branch is performed, the program register is set from the program counter and the program counter is then set from the instructions effective address. Otherwise, the next instruction is taken in sequence.

BLC Branch on Logical Compare



B2 ~~X~~ Byte Location - This modifier specifies which byte of the logical register is to be utilized by the instruction.

BC Byte compare

~~B~~ ~~Byte Mask~~ - This modifier contains the bit configuration to which the selected byte of the logical register will be compared.

NB ~~X~~ Branch on Condition not met -

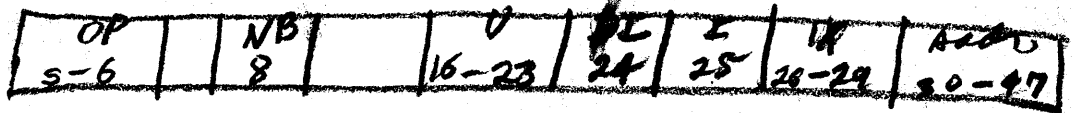
The contents of the 6-bit B modifier are compared bit-for-bit with the contents of the selected byte of the Logical register.

If they are identical, and the N modifier is not set, or if they differ, and N is set, then the branch is performed and the

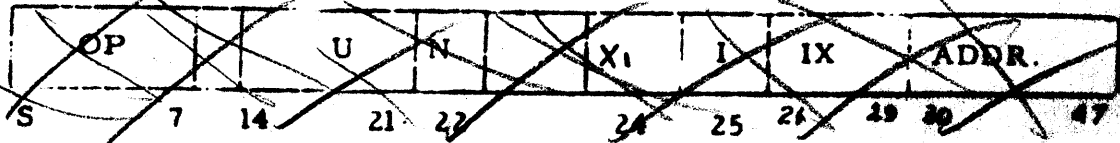
Program Counter and Program Register are set as in other

branches. Otherwise, the next instruction is taken in sequence.

The following branch instrs. utilize the 18-bit decrement field (DF modifier) during their execution. The index reg. specified by the IX modifier (bits 26-29) will be the index register which will be compared to and/or modified by the decrement. Consequently direct indexing will not be possible. Since the double index reg. can be added to the address portion of the instruction, double indexing will be possible. When the indirect addressing method is used, direct indexing will not be possible with the first memory reference, but thereafter will proceed in the normal manner.



BSN Branch on Sense Unit



Modifiers

N - Branch if condition is not met.

U - Code assigned to sense unit. If N is not set, branch if the sense unit is "on"; if N is set, branch if the unit is "off." If the branch is performed, set the Program Counter and Program register as in other branch-class instructions.

~~BBP~~ Branch on B Register Plus

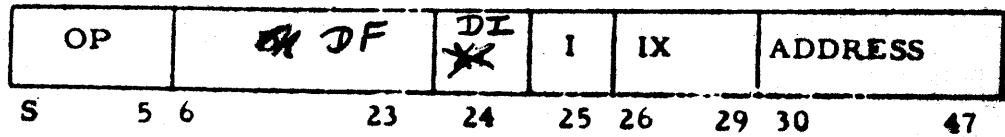
Modifiers

N - Branch if condition is not met.

If the B Register is positive, the computer takes its next instruction from the memory location specified by the effective address of the instruction and proceeds from there.

If the sign of the B Register is negative, the computer proceeds to the next instruction in sequence.

BXH Branch on Index High



This instruction is non-indexable.

~~IX~~ **DF** - Decrement - This field represents an 18-bit number.

If the number in the specified index register is greater than the decrement, ^{DF}~~IX~~, the computer shall branch to the address specified

by the address portion of the instruction. Otherwise, the computer takes the next instruction in sequence. In the interpretation of this instruction "+0" shall be considered greater than "-0".

BXL Branch on Index Low

If the number in the specified index register is less than the decrement ~~DI~~^{DF}, the computer shall branch to the address specified by the address portion of the instruction. Otherwise the computer takes its next instruction in sequence. In the interpretation of this instruction "+0" shall be considered greater than "-0". No indexing is possible with this instruction.

BXE Branch on Index Equal

If the number in the specified index register is equal to the decrement ~~DI~~^{DF}, the computer shall branch to the address specified by the address portion of the instruction. Otherwise, the computer takes the next instruction in sequence. In the interpretation of this instruction "+0" shall be considered greater than "-0". No indexing shall be possible with this instruction.

BSX Branch and Set Index

This instruction replaces the contents of the specified index register with the value of the decrement ~~DI~~^{DF}, and unconditionally branches to the location specified by the address portion of the instruction. No indexing shall be possible with this instruction.

BIX Branch and Increment Index

This instruction replaces the contents of the specified index register with the algebraic sum of the decrement and the original value of the index register, and unconditionally branches to the location specified by the address part of the instruction. No indexing shall be possible with this instruction.

BPX Branch on Positive Index

If the specified index register is positive, the index register contents are decreased by the value of the $\frac{DF}{D_1}$ modifier and the computer branches to the location specified by the address portion of the instruction. If the specified index register is negative the computer proceeds to the next instruction and the contents of the specified index register remain undisturbed. No indexing is possible with this instruction.

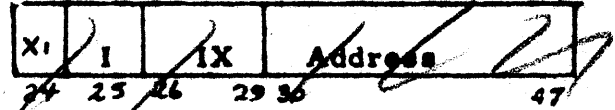
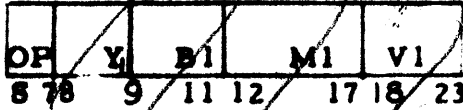
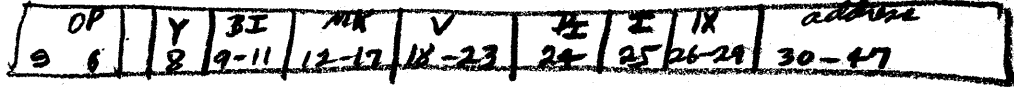
BNX Branch on Negative Index

If the specified index register is negative, the index register contents are increased by the value of the $\frac{DF}{D_1}$ modifier and the computer branches to the location specified by the address portion of the instruction. If the index register is positive, the computer proceeds to the next instruction and the contents of the specified index register remain undisturbed. No indexing is possible with this instruction.

4.8 Logical Class

These instructions deal with bytes or portions of bytes and involve the use of the logical register (48 bits) or the auxiliary logical register (6 bits).

INS Insert



This instruction is indexable

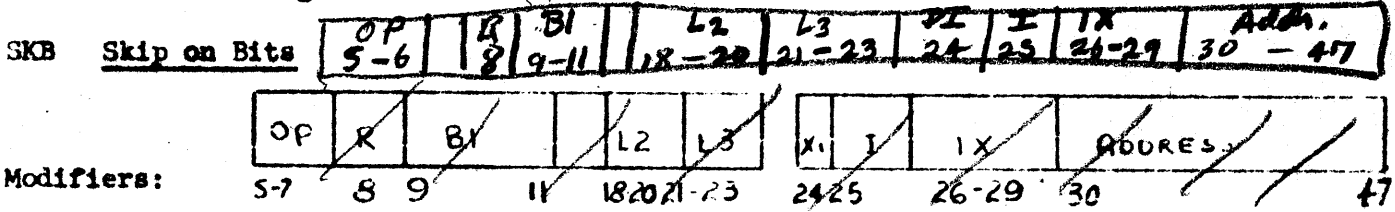
Modifiers

- V₁ Value** The contents of this modifier are inserted into the specified memory byte.
- B₁ Byte #** This modifier specifies the memory byte which is to receive information.
- Y₁ Mask Selector** This modifier selects the mask which is applied to V₁. The meaning of this modifier is as follows:
- 0 - Specifies that bits 12-17 of M1 shall be used as a mask.
 - 1 - Specifies that bits ¹²⁻¹⁴~~15-17~~ of M1 shall select the byte of the logical register which serves as a mask.

MK Mask This modifier serves as a mask or byte indicator subject to the value of the modifier Y₁. The contents of the ^Y~~MK~~ modifier are deposited in the specified memory byte. Modifiers will determine whether the M1 modifier or a specified logical register byte will serve as a mask.

COM Complement Bits

Only modifiers B1, and V1 are used in this instruction and shall have the meaning described above. For the specified memory byte, bits corresponding to 1's in the V1 modifiers are complemented. All other bits on the memory word remain unchanged.



B1: Byte Numbers

This modifier specifies the memory byte to be transferred to the auxiliary logical register.

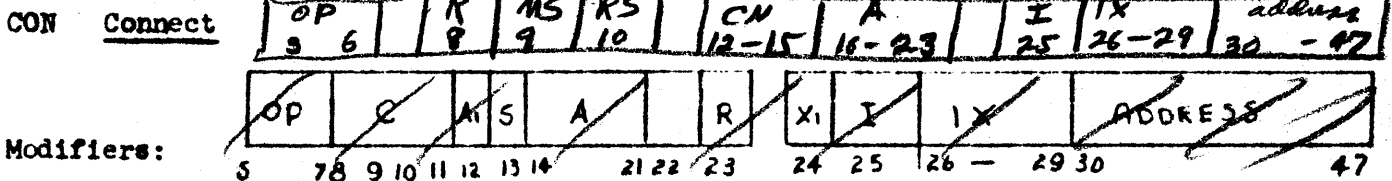
L2: First Bit

This modifier specifies the active first bit of the auxiliary logical register. ¹⁻⁶ ~~1-5~~ are legal values, ⁰ ~~6~~ and 7 are illegal.

L3: Number of Shifts - This modifier specifies the number of bit positions

the auxiliary logical register is shifted right. *0-5 are legal 6-7 are illegal*
The values of L2 + L3 must fall between 1 ≤ L2 + L3 ≤ 6. (If not, will use 0's for middle byte.)

R: Real Data - If set, the right half word will be used as real data. The specified byte of the memory word is transferred to the auxiliary logical register where modifiers L2 and L3 determine the bits of this byte to be added to the program counters. If L2 = ² ~~2~~ and L3 = ⁴ ~~2~~, bits ³ ~~2~~ through ⁴ ~~2~~ of the auxiliary logical register will be added to the program counter.



MS Mask Selector - This bit if set, specifies that the B register shall be used as a mask. When this bit is not set, the instruction shall act on active bytes specified by the A modifier.

A Activity - This modifier specifies the active bytes upon which the instruction operates. This modifier has no meaning when the ^{MS} ~~MS~~ modifier is set.

RS Results Stored:

0 - Accumulator

1 - Memory

R Real Data - This modifier, if set, specifies that the right half word be used as data. *+ transferred to both it, + left exchange reg., an RS mod. of 1 shall be considered illegal whenever the R modifier contains a 1.*

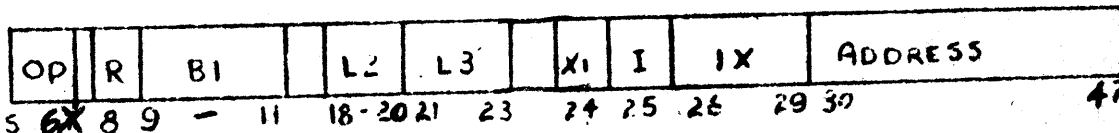
CN Connective - This four bit modifier specifies the logical operation.

Each bit of the C modifier corresponds to a combination of a bit in memory and its counterpart in the accumulator. Whenever this combination appears, corresponding bit of the resultant is set to the value of the associated bit of the C modifier. The combinations are as follows:

If Memory bit is:	and Accumulator bit is:	The Associated bit in CMA bit:
0	0	<i>15</i> 12
0	1	<i>16</i> 13
1	0	<i>17</i> 14
1	1	<i>18</i> 15

To obtain the logical operation "exclusive or" the C modifier would be set to the value 0110 while to obtain the logical operation "and" the C modifier would be set to 0001.

LDS Load and Shift



Modifiers:

R Real Data Tag

B1 Byte Number

L2 First bit of word

L3 Number of bits

Modifiers are as previously described.

Modifiers L2 & L3 have the same restrictions as defined by SKB instr.

This instruction process the memory byte specified by the B1 modifier.

This byte is transferred to the auxiliary logical register where it is

shifted and masked (shifting and masking determined by L2, L3). - 38 -

resultant byte is then loaded into the low order byte of the accumulator. Inactive bytes of the accumulator are cleared.

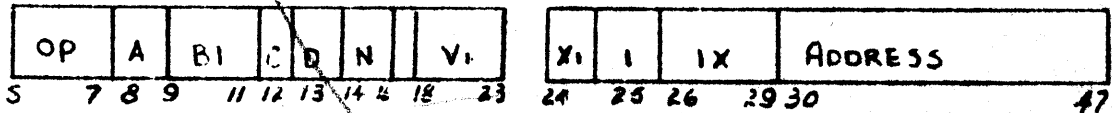
STS Store and Shift

The low order byte of the accumulator is transferred to the auxiliary logical register where it is shifted and masked as in LDS. The resulting active bits are then deposited into the specified byte of the memory word. The L3 modifier in this instruction shall specify the number of shifts to the left.

Modifiers L2 & L3 must meet the following $0 < L2 + L3 < 11$. Illegal values will yield spurious results.

SBS Serial Bit Sense

Instruction word layout



Modifiers:

- A Store item numbers if bit = 1
- BI Byte number This modifier specifies the Byte in the logical register which is to be used with the instruction.
- C Compare Mode
 - 0 - Logical Compare
 - 1 - Arithmetic Compare
- D Direction
 - 0 - Left to Right
 - 1 - Right to Left
- V1 Value - The content of the last N bits of this modifier (N being specified by the N modifier) replaces in the accumulator those items which check.
- N Number of shifts - This modifier specifies the number of shifts and the item length. The values 0 and 7 are illegal.

Omitted ?

B Register Word Layout



C1 Number of comparisons - This modifier specifies the number of comparisons which are made

A1 Base Address to Store item number - Count number of successful comparisons.

The memory word specified by the effective address is brought into the accumulator. Comparisons are made between N bit items of the accumulator and the last N bits of the logical register. After each check, the left B register is increased by one, and the accumulator is shifted N places. After each successful check, the right B register is increased by one and if the A modifier is set, the resultant address is used to specify the memory register into which the following information is placed:

Left half word: Contents of left B register

Right half word: Location within the word of item compared

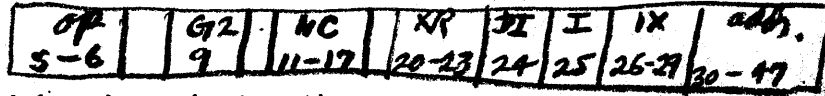
The contents of the last N bits of the instruction left half word replace, in the accumulator, these items which check.

After all the bits of the accumulator have been checked, the instruction is complete. If no successful comparisons have occurred, one instruction is skipped; if any have occurred, the next instruction is taken in sequence.

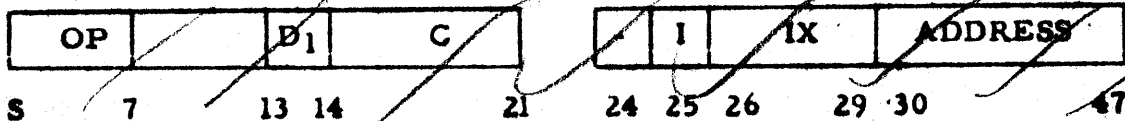
omitted

4.9 Convert Class

The convert operations enable a program to have very rapid access to information stored as tables in core storage. A single convert instruction can perform a series of table look-up operations by making multiple references to core storage.



The following format will be used for these instructions:



Modifiers:

NC - Count This modifier specifies the number of converts to be made.

~~IX~~ **XR** - Index Register Selection - This modifier specifies the index register to be loaded with the right-most 18 bits of the final core memory reference.

G2 ~~X~~ - Direction When this bit is a "0", conversion shall progress from left (byte 0) to right. When this bit is a "1", conversion shall progress from right (byte 7) to left.

CRA - Convert by Replacement from the Accumulator

This instruction treats the contents of the accumulator as eight 6-bit bytes and replaces these bytes by bytes from core storage. The following discussion assumes that the ~~D₁~~ **G2** modifier is cleared.

Byte 0 of the accumulator is used to increment the address portion of the instruction. Byte 0 of the contents of the "incremented" address replaces byte 0 of the accumulator, and the right-most 18 bits of the contents of the "incremented" memory register are added to byte 1 of the accumulator to form a second memory address. Byte 0 of the contents of the second memory address replace Byte 1 of the accumulator, and Byte 2 of the accumulator increments the right-most 18 bits of the contents of the memory register to form a third memory address. The process of replacement continues for the number of times specified by the C modifier. The specified index register is set to the value of the right-most 18 bits of the contents of the final memory reference.

When the ^{GZ}~~D~~ modifier is set, conversion shall begin with byte 7 and progress to the left.

CAB - Convert by Addition from the B-Register

This instruction treats the contents of the B-register as eight six-bit quantities which are used to form memory addresses as previously described. Words selected from storage by the references are added to the contents of the accumulator. No overflow alarm shall be generated as a result of the instruction.

The specified index register is set to the value of the right-most 18 bits of the contents of the final memory reference. *Modifier*

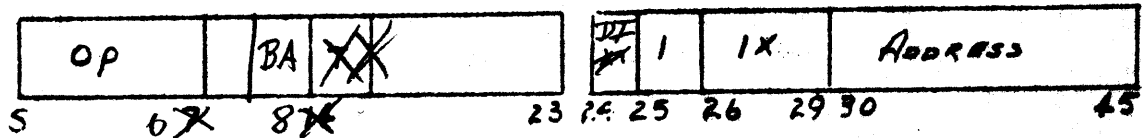
GZ has no meaning in this instr. Conversion shall progress from left to right. After a byte has been used for a table reference, the contents of the B reg is cycled left until byte L1 replaces L0 etc.

any reference to an internal register shall be considered illegal

4. 10 Miscellaneous Class Instructions

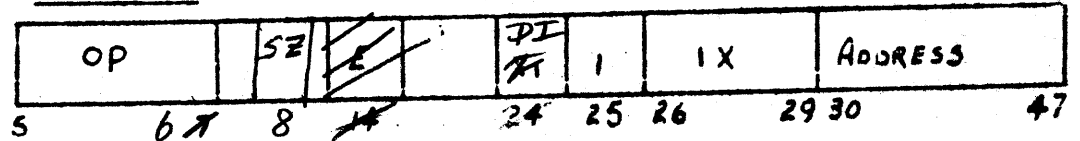
This class includes instructions which differ from each other, and from other similar instructions in other classes. For example, FST (Full-word Store) is one machine cycle shorter than all other store-class instructions. The word layouts and modifiers are described for each instruction in the miscellaneous class.

HLT Halt



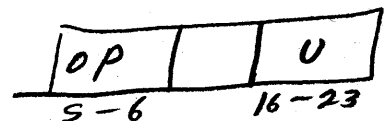
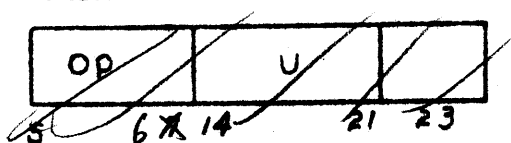
The computer is stopped when all I/O operations presently in process are complete. When the ^{BA}~~X~~ modifier is set, activating the Program Continue Control located on the Operator's Console shall cause the program to start at the effective address generated by the halt instruction, rather than the next sequential location. *This instr. is indexable.*

FST Full Store



The contents of the full accumulator are placed in the memory word specified by the effective address of the instruction. When the ^{SZ}~~E~~ modifier is set, the instruction shall store zeros in specified address. *Is indexable.*

PER OPERATE



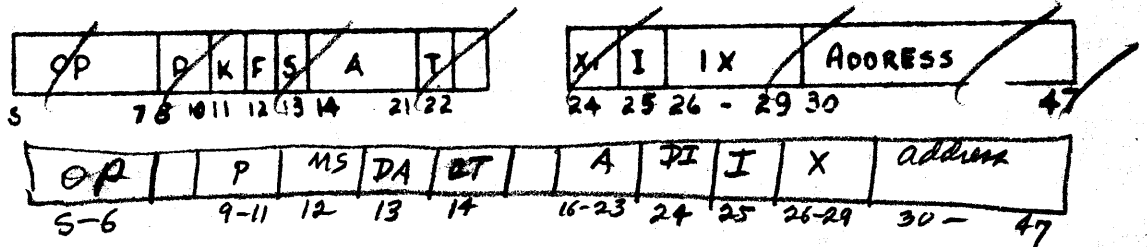
Right half word has no meaning

U - Operate-unit code

The Operate instruction is used to actuate certain electro-mechanical devices (e. g., condition lights, area discriminator, etc.). Also, through the PER instruction, microprogramming can be performed.

That is, certain commands, such as "clear A register" and "transfer A-register contents to selected Memory Buffer Register," can be performed as Operate instruction. *Further details are not available at this time*

CAS Compare Accumulator with Storage



Modifiers

MS Mask Selector - When set this modifier specifies that the B register shall be used as a mask. *only for a memory word. The accumulator must be masked by a prior instr. to obtain a valid comparison* When this modifier is not set, the instruction shall operate on bytes specified by the A modifier. Modifiers P, and A are meaningless when MS : 1.

P, A Displacement and Activity - These modifiers shall be interpreted as in the load class

DA Difference in Accumulator - If this bit is set, the difference, arithmetic or logical depending on the *CT* modifier, is left in the accumulator.

The logical difference means the "ex-

When the DA tag is 0, the instr. will not change the active bits or bytes of the accumulator (inactive bits & bytes are destroyed.) When the DA tag is set, the following conditions exist:

1. Logical Compare


- (a) MS=1, all inactive bits set to zero upon completion,
- (b) MS=0 " " " " " " " " " "

2. Arithmetic Compare

- (a) MS=1 before subtraction the inactive bits of the memory word will be set to 0, a full wd. subtraction is then made, & the results left in the acc.
- (b) MS=0 all inactive bytes in both words will be set to pos. or neg. 0's depending on the sign of the left most active byte. a full wd. subtr. is then made

clusive or," where 0's are left only where corresponding bits compare. The "arithmetic difference" is the difference resulting from a subtraction of the operand from the contents of the accumulator.

- 1 Logical Compare - If the active bits of the accumulator and memory word specified correspond bit for bit, the program counter is stepped an additional time. If the bits do not compare, the program counter is stepped normally. The bits to be compared are determined by the modifiers.

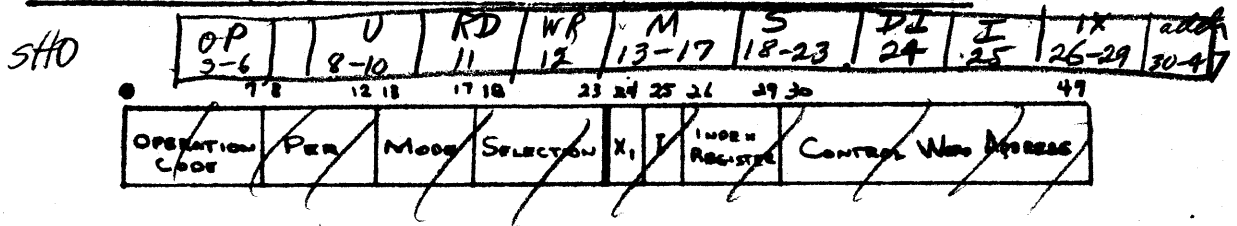
 - 0 Arithmetic Compare - The accumulator is compared with the memory word. If the accumulator is greater than the memory word, the program counter is stepped in the normal manner. If the accumulator is equal to the memory word, the program counter is stepped one additional time. If the accumulator is less than the memory word, the program counter is stepped two additional times.
- 

~~If selective comparison is to take place (i. e. B register mask or byte comparison) and the P modifier is set, inactive bits shall be cleared before comparison takes place.~~

XEC Execute

This instruction shall cause the computer to perform the instruction located in the memory address specified by the effective address without altering the program counter. Upon the completion of this instruction, the next instruction is taken in sequence, unless the executed instruction is an active branch instruction, *SICB*, or *CAS instr*,

4.11 HIGH SPEED I/O OPERATION (SHO INSTRUCTION FORMAT)



~~7 (operation code) shall select the (rd) instruction to initiate~~
 The instrⁿ shall enable the selection & operation of I/O high speed devices under control of the I/O control unit.

~~12 (per) shall be bits designated for specific operations~~

V-operate 001
 A 0001 (Lock Address Counter) shall inhibit the stepping of the I/O address counter

010 (Inhibit Alarm Disconnect) shall be used for drums to inhibit a disconnect caused by an alarm condition

100 Inhibit Clearing, IO Reg - when drums are selected, causes an inhibit field stop position
~~(read or write) shall signify whether I/O transfer shall be~~

~~read or write operation~~

~~17 (Mode) shall be used in drum operation only. A detailed description of the drum modes begins in section 6 of the Drum Specification SAC-...~~

Code	Operation
0000	Addressable
0001	Addressable-interleave by 2 (skip every other register)
0010	Addressable-interleave by 4 (skip 3 registers)
1000	Identity Drum Crase
0100	Block transfer (transfer of entire block between drum and memory starting at the first possible address)

~~21 (selection code) shall select the high speed I/O device~~

000001	Data Drum			
000010	aux. Mem. Drum 1			
000011	"	"	"	2
000100	"	"	"	3
001001	"	"	"	4
001010	"	"	"	5

RD - Read Mode 1 in bit 11 shall cause a read op.
 WR - Write Mode 1 in bit 12 " " " write " "
 S - selection This shall select the high speed I/O device

<u>Code</u>	<u>Selection</u>
000010	Drum
111000	I/O Register

(INDIRECT INDEXING) - A "1" SHALL ALLOW FOR AN

INDIRECT INDEXING OPERATION.

24 - (Indirect Addressing) a "1" shall allow for indirect addressing.

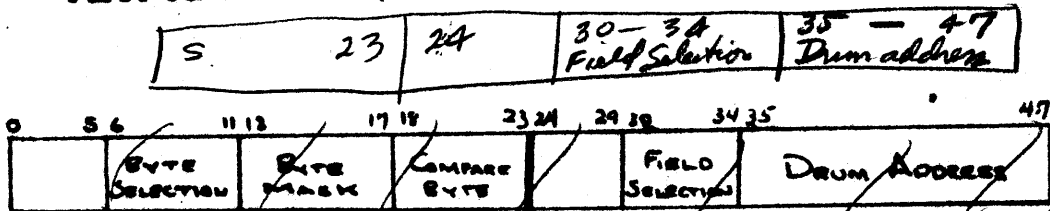
30-34 (Index Register) shall allow for selection of a particular index register.

35-47 (Control Word Address) shall contain the memory location in which the control word is stored.

Control Words

The drum operation depends upon two control words to perform the instruction word function while all other high speed channels use only one word containing the I/O word count and I/O address count (refer to figure 2-2)

First Control Word (used only by drums)



6: 11 (Byte Selection) shall indicate which of the 8 bytes in the I/O buffer is to be compared with the Compare Byte in L18-L23 of the same control word.

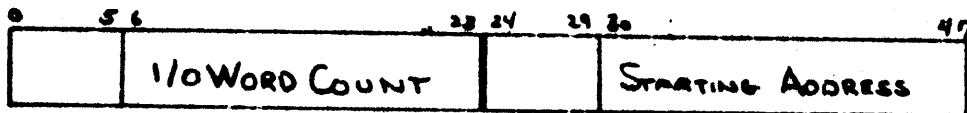
11-12 (Byte Mask) shall indicate that a bit containing a one corresponds to that bit in the selected byte that is not to be used for comparison.

18-23 (Compare byte) shall be used to compare with the I/O Buffer selected byte.

30-34 (Field Selection) shall be used to select the drum field for a drum I/O operation. ~~For Field Selection Codes Refer To~~

35-47 (Drum address) shall signify the starting address of word transfer on the selected drum field.

Second Control Word of Drums (First Control Word of other High-Speed Devices)

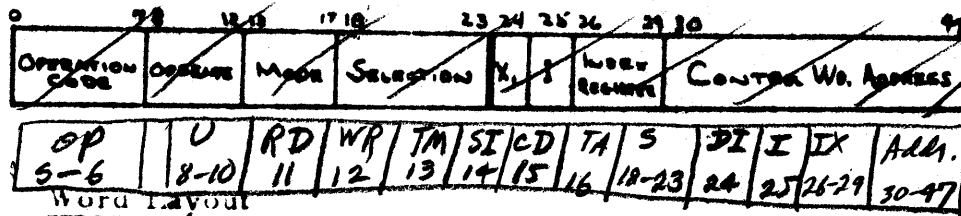


6-23 (I/O Word Count) shall contain the number of words to be read or written.

30-47 (Starting Address) shall contain the memory address of the first word to be written or read.

4.12 Low Speed I/O OPERATION (SLO INSTRUCTION FORMAT)

Low speed operation shall be initiated by a SLO ("Select Low-Speed")



0-7 (Operation Code) shall select the SLO instruction to initiate a low speed operation.

U 8-12 (Operate Codes) shall be designated for special operations of low speed operations.

	<u>Code</u>	<u>Function</u>
001	0001	Backspace
010	0010	Rewind
011	0011	Write end of file
100	0100	Write end of tape
001-010	00101 to 00110 <i>for card punch</i>	Punch operate codes (2)
001-010	01000 to 10001 <i>for Printer</i>	Printer operate codes (10)

~~11-17~~ (Mode) shall determine the mode of operation for the I/O operation.

RD } A- ~~13~~ (read = 0, write = 1) shall signify whether
WR } operation is in a read or write mode.

~~14~~ (status 0, identity 1) is used for tapes to
PUT TAPES INTO AN IDENTITY OPERATION WHEN
DESIRED.

TM shall distinguish the matrix between octal or alphanumeric operation from the I/O Typewriter

I=0 Typewriter

- 0 out of
- 1 alphanumeric

SI used to put tape into an identity operation when done

- 0 Status
- 1 identity

CD Binary or BC Hollerith

- 0 Binary
- 1 BC Hollerith

TA Tape identity address

- 1 Store address
- 0 Inhibit store address

S Selection

~~C- 15 (Low Speed Tape = 0, High Speed = 1) shall distinguish the instruction between A Low Or High SPEED read-write TAPE OPERATION.~~

~~16 (binary = 0, binary coded Hollerith = 1) shall select binary or binary coded Hollerith operations.~~

~~E- 17 (on line operation = 0, off line operation = 1) shall allow tape to printer off line operation or card reader to tape off line operation.~~

~~18-23 (Selection Codes) shall select the I/O device to be used in the I/O transfer.~~

Code (Octal)	Selection
1-6 001000-001111	Tape Adapter 1
7-12 010000-010111	Tape Adapter 2
13-18 011000-011111	Tape Adapter 3
19-24 100000-100111	Tape Adapter 4
25-30 101000-101111	Tape Adapter 5
31-36 110000-110111	Tape Adapter 6
60 110000	Traffic Control Center
61 000010	Printer
62 000011	Punch
63 000001	Card Reader
64 110001	Teletype Channel
70-79 101000-101111	I/O Typewriters 1-8

~~24 - (Indirect Address) a "1" shall allow for A, to correct to DENTIVE~~

OPERATION

Modifier's DI, I, IX & address have same meaning as previously specified

25 - (Indirect Address) a "1" shall allow for indirect addressing.

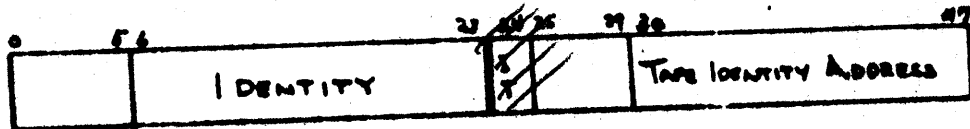
26-29 (Index Register) shall allow for selection of a particular index register

30-47 (Control Word Address) shall contain the memory location in which the control word is stored

Control Words

Two control words are used only in tape operation. The other low speed I/O devices only make use of the second control word.

First Control Word (used only by tapes)

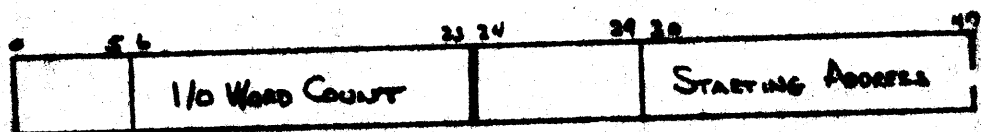


6 - 23 (Identity) shall indicate the identity to be written on tapes or compared with on reading from tapes.

~~24 - (tape identity address transfer) - - If RS is a "1" the tape identity shall be transferred to address specified in R4 - R23.~~

30 - 47 (identity address) shall indicate the memory address where the identity of the record read from tapes should be stored during a status read operation.

Second Control Word (First Control Word of Other Low Speed Devices)



6 - 23 (I/O word count) shall contain the number of words to be read or written

30 - 47 (starting address) shall contain the memory location of the first word to be written or read.