INSTRUCTION SET

FOR

1080

Fabri-Tek Instruments, Inc.

5225 Verona Road

Madison, Wisconsin

February 1971

INSTRUCTION SET FOR 1080 PROCESSOR

I. INTRODUCTION

The instruction set for the 1080 consists of two groups of instructions plus the Input/Output (I/O) instructions. Group I contains those instructions which are not combinable and which may - and in some cases must - be followed by an operand. Group II contains instructions which are combinable and which do not use an operand (see Fabri-Tek Instruments FASS Programming Manual for information concerning the operand).

Listed below are the instructions in their respective groups. Each instruction is given in mnemonic form with a short description followed by the word format indicating the bits set. (An "x" indicates the bit can be either a "1" or a "0".) A shorthand description of the instruction is also given.

The shorthand description of the instruction includes several characters and symbols which are defined as follows:

"C(__)" indicates the contents of __. For example, C(AC) refers to the contents of the accumulator. Similarly, C(M) designates the contents, or data, found on the memory buffer while DB refers to the data on the data bus. (The data bus is not a storage register but rather the data lines exiting from the arithmetic unit.)

"→" indicates a value that is placed or moved to an indicated location. For example, C(AC) → DB means the contents of the accumulator are placed on the data bus.

"." indicates a logical AND operation.

"$\overline{M}$" or "$\overline{AC}$" indicates the 1's complement or logical negation of the contents of the memory or accumulator.

"-M" or "-AC" indicates the 2's complement or arithmetic negation of the contents of the memory or accumulator.

For example,

$$C(AC) + C(\overline{M}) \rightarrow DB$$

means that the contents of the accumulator are added to the complement of the contents of a memory location (specified by an operand) and the result is placed on the data bus.

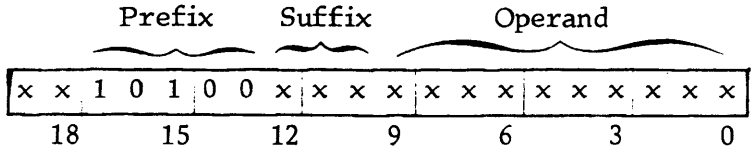
II. GROUP I

This set of instructions is divided into six subsets. The first four subsets consist of the prefix of an instruction while the fifth subset contains the suffix which must follow the prefix. The sixth subset contains jump (JMP and JMS) instructions which use no suffix. The Group I instructions are not combinable.

A. Two Operand Prefixes

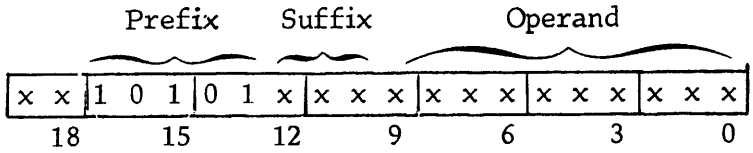
This subset of instructions operates on the contents of the accumulator and a memory location that is specified by an operand and the result is placed in the data bus. These instructions must be followed by a suffix and operand.

A + M Accumulator + Memory



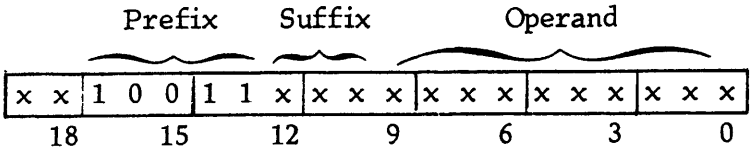
$$C(AC) + C(M) \rightarrow DB$$

AMP Accumulator + Memory + 1



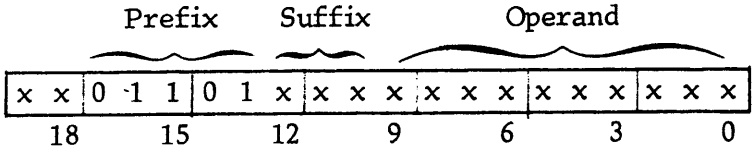
$C(AC) + C(M) + 1 \rightarrow DB$

A-M Accumulator - Memory



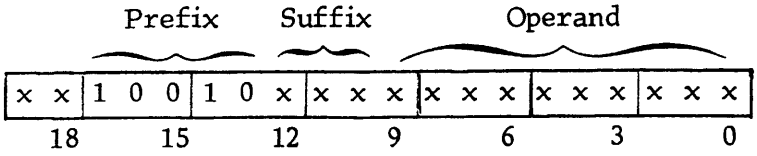
$C(AC) - C(M) \rightarrow DB$

M-A Memory - Accumulator



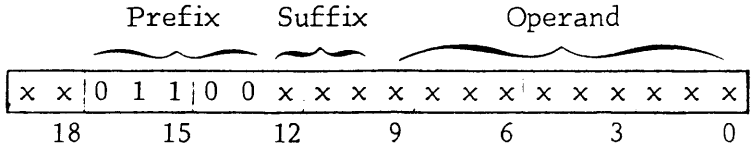
$C(M) - C(AC) \rightarrow DB$

ACM Accumulator + Complement of Memory



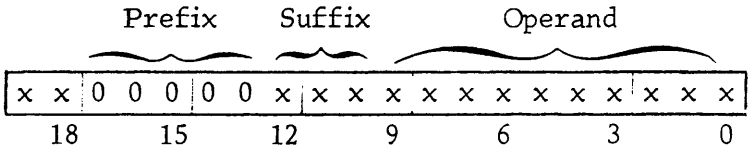
$C(AC) + C(\overline{M}) \rightarrow DB$

CAM Complement of Accumulator + Memory



$C(\overline{AC}) + C(M) \rightarrow DB$

AND Logical "AND" of Accumulator and Memory



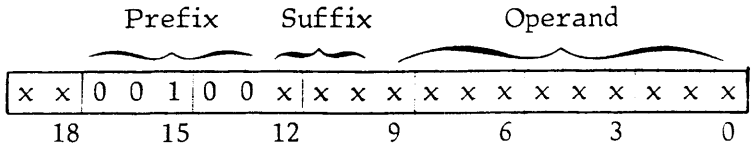
$C(AC) \cdot C(M) \rightarrow DB$

(A Boolean "AND" operation is performed on the C(AC) and C(M) and the result is placed on the data bus.)

B. Memory Operand Prefixes

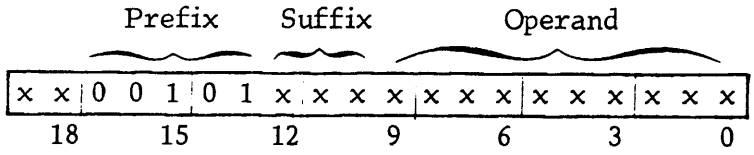
This subset of instructions operates on the contents of a memory location that is specified by an operand and the result is placed on the data bus. These instructions must be followed by a suffix and by an operand.

MEM Memory



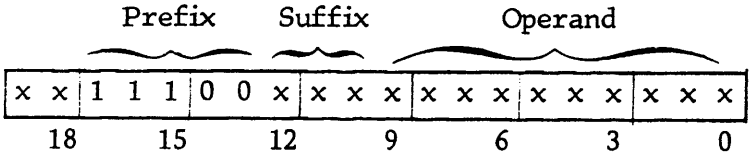
$C(M) \rightarrow DB$

MPO Memory + 1



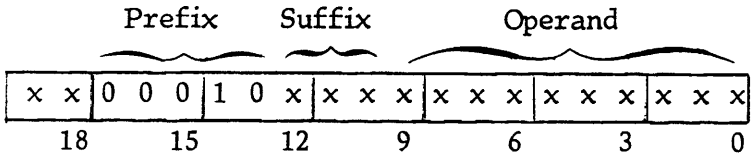
C(M) + 1 → DB

MMO Memory - 1



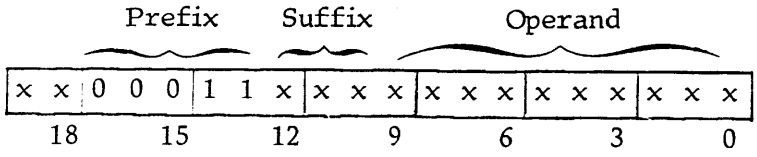
C(M) - 1 → DB

MCP Complement of Memory



$C(\overline{M}) \rightarrow DB$

MNG Negative of Memory (2's Complement)

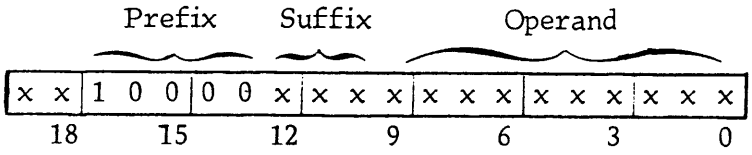


C(-M) → DB

C. Accumulator Operand Prefixes

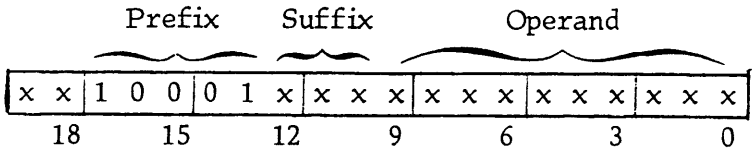
This subset of instructions operates on the contents of the accumulator and the result is placed on the data bus. These instructions must be followed by a suffix. An operand may be required, but this depends on the suffix.

ACC Accumulator



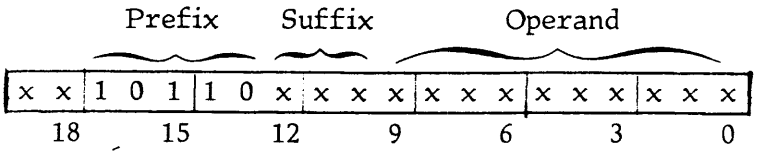
C(AC) → DB

APO Accumulator + 1



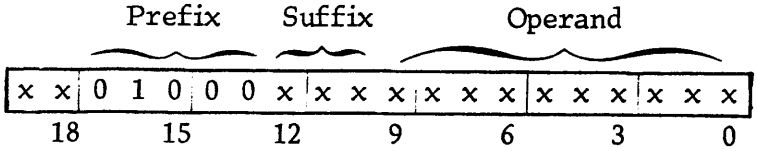
C(AC) + 1 → DB

AMO Accumulator - 1



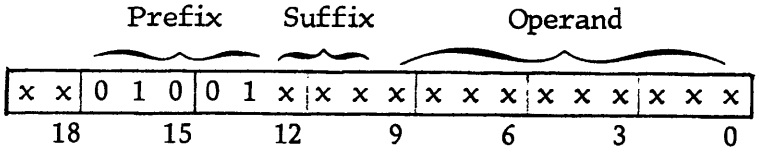
C(AC) - 1 → DB

ACP Complement of Accumulator



$C(\overline{AC})$ — DB

ANG Negative of Accumulator (2's Complement)

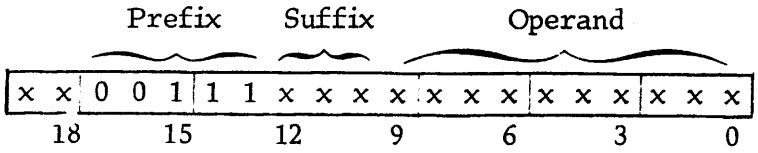


C(-AC) — DB

D. Constant Operand Prefixes

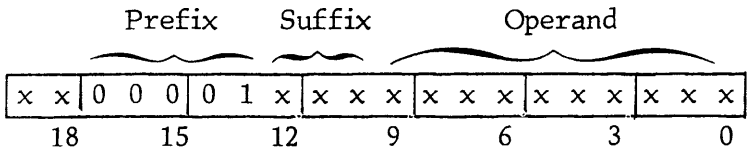
This subset of instructions places a constant on the data bus. These instructions must be followed by a suffix. An operand may be required but this depends on the suffix.

ZER Place Zero on Data Bus



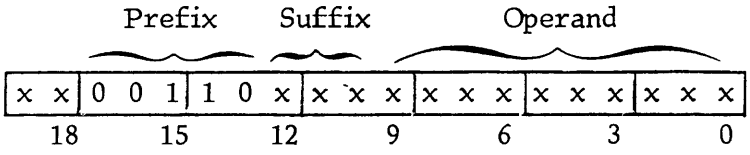
ø — DB

ONE Place One on Data Bus,



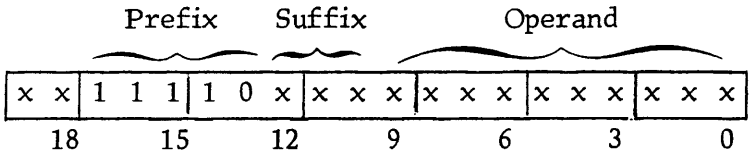
1 — DB

MON Place Minus One on Data Bus



-1 — DB

MTO Place Minus Two on Data Bus

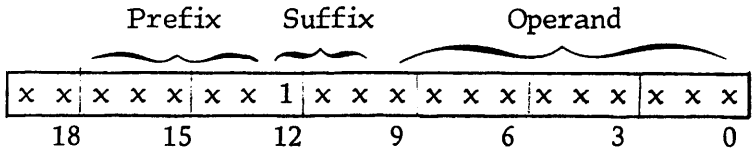


-2 — DB

E. The Suffixes

All of the preceeding instructions require a suffix for completion of the instruction (in some cases an operand is also necessary). The purpose of the suffix is to move or to test data on the data bus.

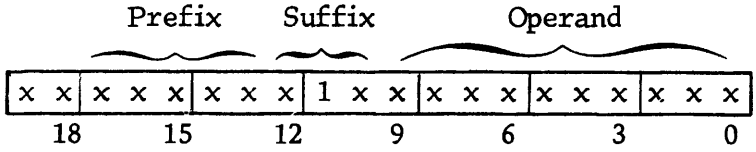
A Move Data Bus to Accumulator



DB — C(AC)

(Any operand following this suffix will be ignored.)

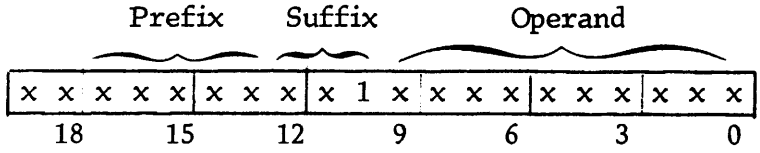
M Move Data Bus to Memory



DB — C(M)

(This suffix must be followed by an operand specifying the memory location to be addressed.)

Z Skip if Data Bus is Zero



Skip next instruction if DB = ∅.

The three suffixes can be combined in any way. The instruction

MPOAMZ POINT

is valid. It states that the contents of the address labeled POINT will be incremented and the new value placed in the accumulator, placed back in POINT and tested for zero. If zero, it will skip the next instruction.

It should also be noted that the suffixes need not appear in any order.

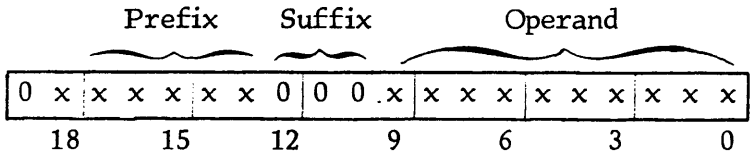
MPOZAM POINT

is the same instruction as the previous example.

F. Jump Instructions

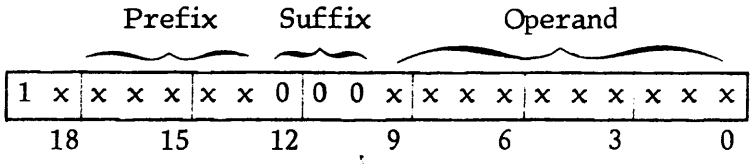
There are two different types of jump or branching instructions. Each must be followed by an operand.

JMP Jump to a Memory Location



Unconditional jump (branch). The operand designates the memory location at which the program is to continue.

JMS Jump to a Subroutine



The operand designates the initial address of the subroutine. The address succeeding the JMS instruction will be placed at the initial address of the subroutine to serve as a "mark." At the end of the subroutine the program returns to the "marked" address.

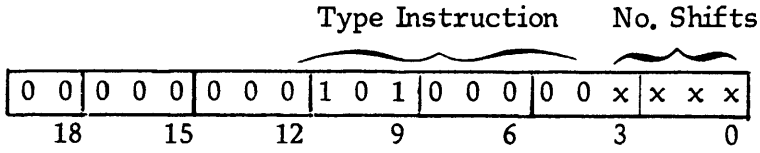
III. GROUP II INSTRUCTIONS

A. Shift Instructions

Shift instructions operate on the contents of the accumulator. They are followed by an octal number less than 20_8 specifying the number of shifts to be made. (From 0 to 15_{10} shifts can be made.) The link is not included in any of the shift operations. Arithmetic shifts allow bits to be "dropped off the end" whereas logical shifts cause bits to be shifted in a "circular manner."

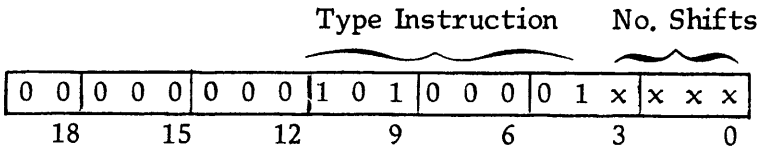
These instructions are not combinable with any others.

LASH Left Arithmetic Shift



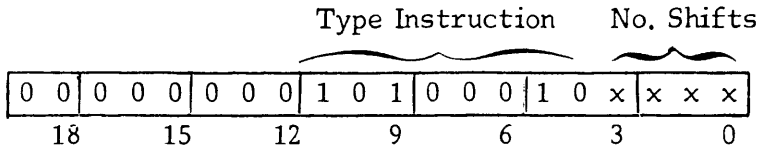
C(AC) are shifted left as specified by the octal number following the instruction. Zeros will be shifted into bit Ø.

RASH Right Arithmetic Shift



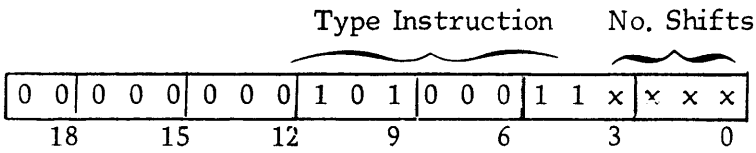
C(AC) are shifted right as specified by the octal number following the instruction. Zeros will be shifted into bit 19 if bit 19 originally had a zero in it (positive number) and 1's will be shifted into bit 19 if bit 19 originally had a one in it (negative number).

LLSH Left Logical Shift



C(AC) are shifted left as specified by the octal number following the instruction. The shift is "end around," thus the data leaving bit 19 enters bit Ø.

RLSH Right Logical Shift

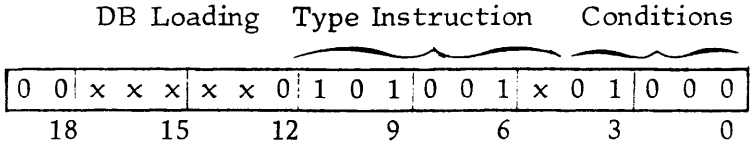


C(AC) are shifted right as specified by the octal number following the instruction. The shift is "end around," thus the data leaving bit Ø enters bit 19.

B. Test (Branching) Instructions

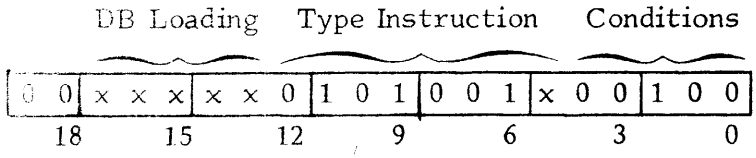
1. There are five conditions which may cause a branch to occur. Any, or all, of these instructions can be combined.

ACØ Test Bit Ø of Accumulator for a "1"



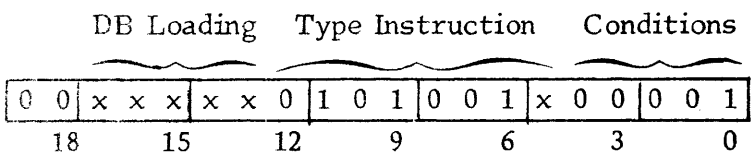
$C(AC_0) \stackrel{?}{=} 1$

AC19 Test Bit 19 of Accumulator for a "1"



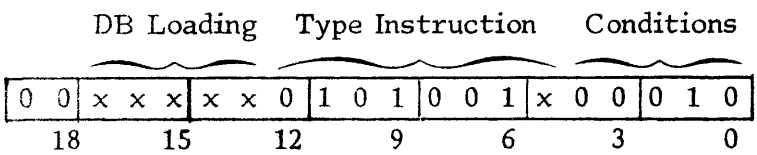
$$C(AC_{19}) \stackrel{?}{=} 1$$

L Test Link for a "1"



$$C(L) \stackrel{?}{=} 1$$

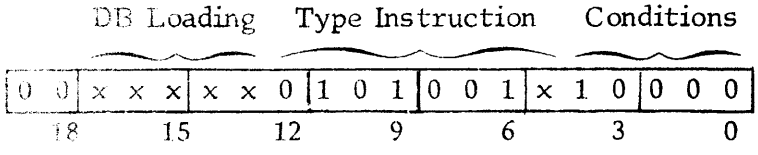
COUT Test Carry Output of Data Bus for a "1"



$$DB_{20} \stackrel{?}{=} 1$$

The carry output is the overflow bit of the data bus. This instruction is similar to "L" above. The state of COUT is determined by the first five Data Bus instructions (see Section C below).

ZDB Test Data Bus for a "0"

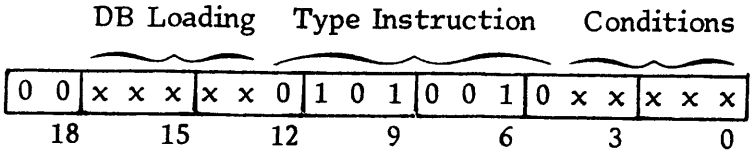


$$DB \stackrel{?}{=} 0$$

The state of the Data Bus is determined by the first five Data Bus instructions (see Section C below).

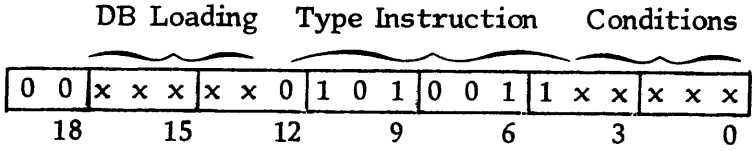
2. Given the above conditions, the choice is available whether to execute or skip the following instruction.

SKIP Skip the Following Instruction



Skip the following instruction if the test condition(s) is (are) satisfied. See section above for test conditions.

EXCT Execute the Following Instruction



Execute the following instruction if the test condition(s) is (are) satisfied.

3. As an example, if the instruction

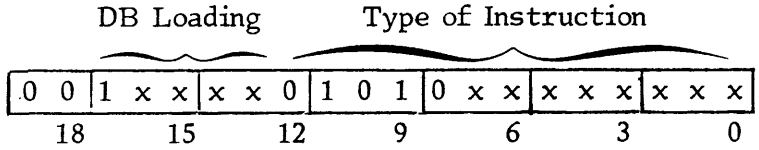
SKIP ACØ AC19

were given, it would cause the instruction following it to be skipped if the data in the accumulator is odd or if it is negative. This same instruction could also mean that the instruction following it is to be executed if the data in the accumulator is positive and an even number.

C. Data Bus Instructions

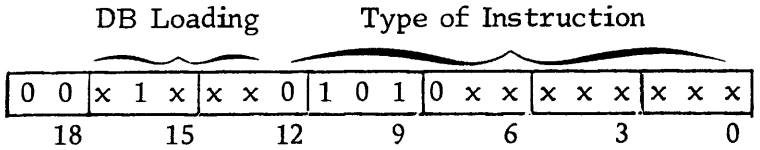
The following six instructions are combinable. If two or more are combined their action is a logical "AND" operation. They can be used to load the data bus preparatory to testing for a ZDB or a COUT and to transfer the data bus to the AC.

LAC Load Accumulator on Data Bus



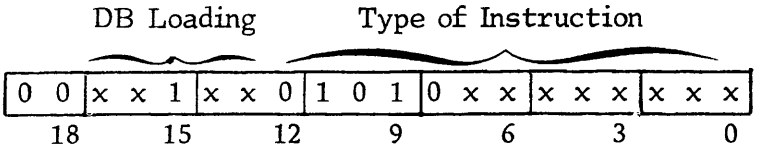
C(AC) — DB

LCAC Load Complement of Accumulator on Data Bus



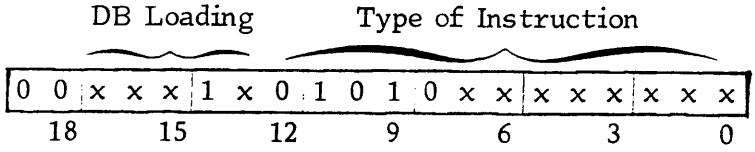
C($\overline{AC}$) — DB

LM Load Memory Buffer on Data Bus



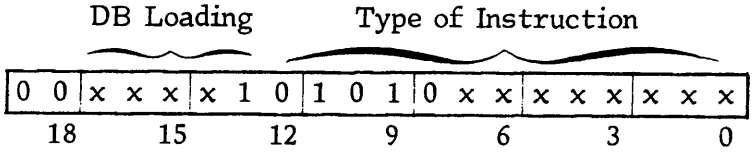
C(MB) — DB

LCM Load Complement of Memory Buffer on Data Bus



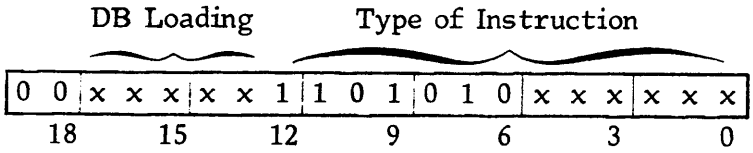
$C(\overline{MB}) \rightarrow DB$

CIN Increment Data Bus



DB + 1 → DB

TDAC Transfer Data Bus to the Accumulator

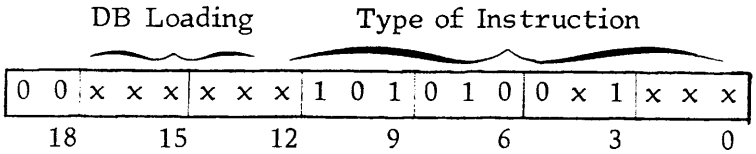


DB → C(AC)

The state of the data bus (DB) is determined by one of the other loading instructions given in this group.

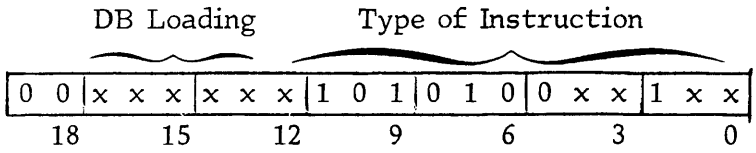
D. Miscellaneous Instructions

CLL Clear Link to a "0"



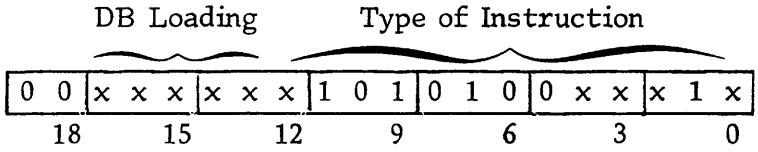
0 → C(L)

STL Set Link to a "1"



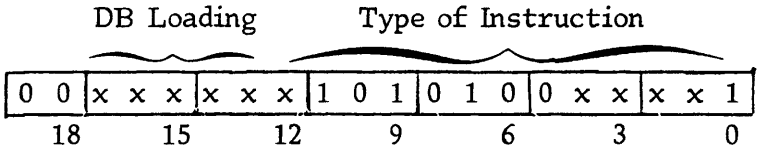
1 — C(L)

TLAC Transfer Link to Bit 19 of the Accumulator



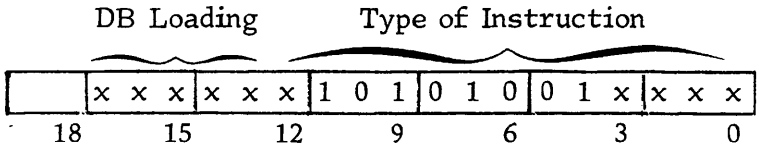
C(L) — $C(AC_{19})$; C(L) remains unchanged

TACC Transfer Bit 19 of the Accumulator to the Link



$C(AC_{19})$ — C(L); $C(AC_{19})$ remains unchanged

STOP Stop Executing Instructions



Operations on the accumulator may be combined with the Miscellaneous Instructions. The order of events is as follows:

1. CLL or STL

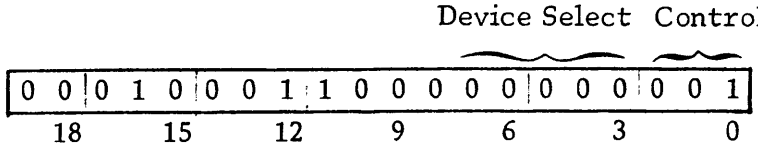
2. Operation on accumulator (may complement link)

3. TLAC or TACL

IV. INPUT- OUTPUT (I/O) INSTRUCTIONS

A. Internal

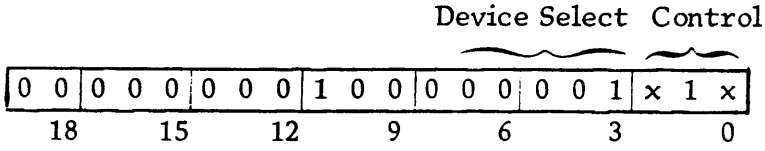
TACXD Transfer Accumulator to X-Display Register



$$C(AC) \rightarrow C(X)$$

This instruction transfers the lower 14 bits of the AC to the X register. Ø corresponds to positioning the display device to the left.

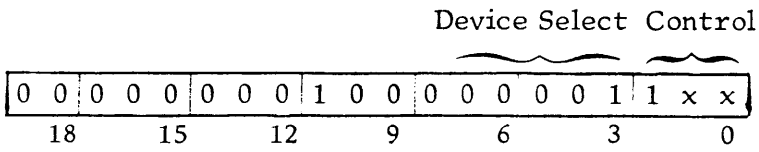
TACYD Transfer Accumulator to Y-Display Register



$$C(AC) \rightarrow C(Y)$$

This instruction transfers the upper 14 bits of the AC to the Y register. Ø corresponds to positioning the display device at mid-scale.

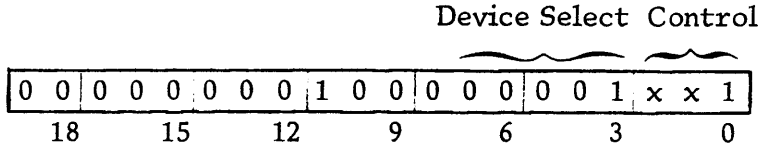
INCXD Increment X Register



$$C(X) + 1 \rightarrow C(X)$$

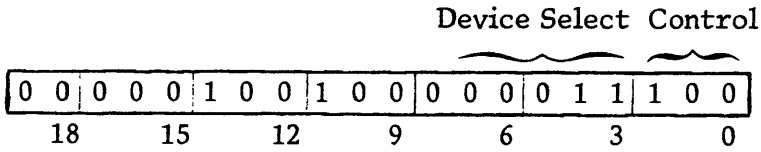
This instruction is normally combined with TACYD to provide a sweep on the display device while the data is loaded in the Y register.

INTENS Intensify the Displayed Point



This instruction generates a pulse that intensifies the point currently displayed. This pulse is available externally. Alternatively, the pulse may be used to trigger external devices.

STATUS Read Status of Front Panel Switches

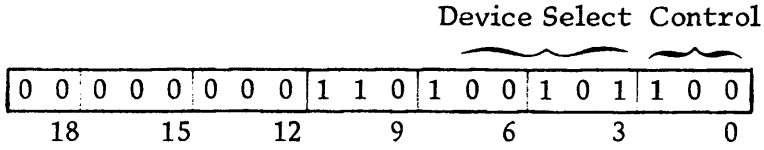


C(S) → C(AC)

The status of some of the front panel switches is available for inspection when this instruction is used. The table on page 20 indicates which switches can be inspected and their corresponding bits.

B. External I/O Instructions

TTYRF Test Teletype Reader Flag



This instruction will test the teletype (TTY) reader flag and skip next instruction if TTY is ready.

STATUS WORD INTERPRETATION

READOUT MEMORY ALLOCATION:

STARTING

1K (pushbutton in = "1")	Bit 0
2K (pushbutton in = "1")	Bit 1
4K (pushbutton in = "1")	Bit 2
8K (pushbutton in = "1")	Bit 3

SIZE

1K (pushbutton out = "1")	Bit 4
2K (pushbutton out = "1")	Bit 5
4K (pushbutton out = "1")	Bit 6
8K (pushbutton out = "1")	Bit 7

MEASURE MEMORY ALLOCATION:

STARTING

1K (pushbutton in = "1")	Bit 8
2K (pushbutton in = "1")	Bit 9
4K (pushbutton in = "1")	Bit 10
8K (pushbutton in = "1")	Bit 11

SIZE

1K (pushbutton out = "1")	Bit 12
2K (pushbutton out = "1")	Bit 13
4K (pushbutton out = "1")	Bit 14
8K (pushbutton out = "1")	Bit 15

INPUT DATA ADD/SUBTRACT:

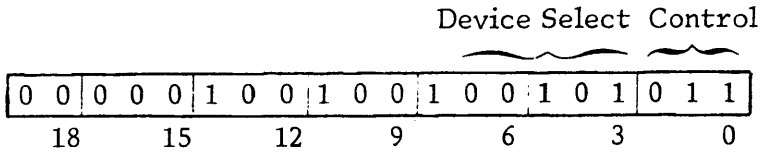
ADD (Pushbutton in = "1")	Bit 16

VIEW INPUT/MEMORY/CONTINUOUS

INPUT (pushbutton out = "1")	Bit 17
MEMORY (pushbutton out = "1")	Bit 18

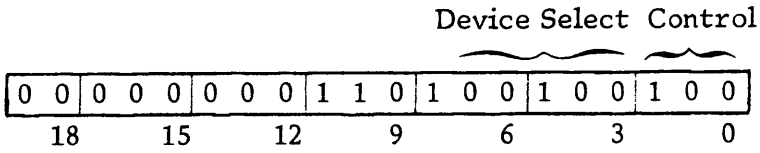
MEASURE (pushbutton in = "1")	Bit 19

RDTTY Read Teletype



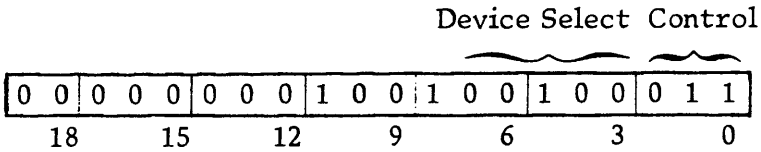
This instruction will clear the accumulator, move the contents of the teletype reader buffer to the accumulator, fetch the next character, and clear the TTY reader flag.

TTYPF Teletype Print Flag



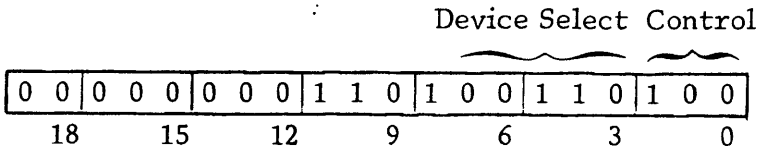
This instruction will test the teletype printer flag and skip next instruction if printer is ready.

PRTTY Print Teletype



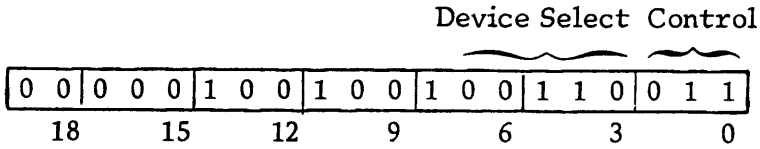
This instruction will print (or punch) a character determined by the contents of bits Ø - 7 of the accumulator on the teletype and clear the flag.

HSRF High Speed Reader Flag



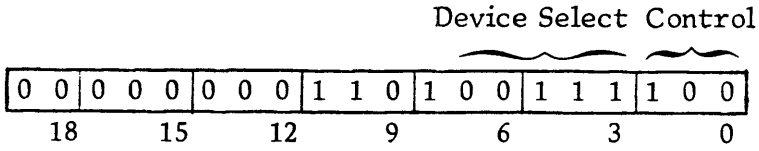
This instruction will test the high speed reader flag and skip the next instruction if the reader is ready.

RHSR Read High Speed Reader



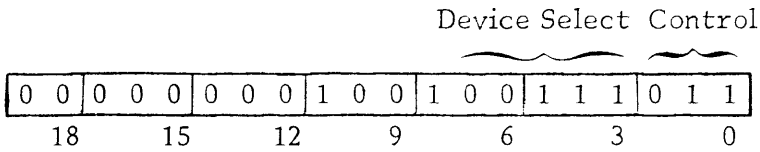
This instruction will clear the accumulator and move the contents of the high speed reader buffer to the accumulator, then clear the flag and fetch the next character.

HSPF High Speed Punch Flag



This instruction will test the high speed punch flag and skip the next instruction if the punch is ready.

PHSP Punch High Speed Punch



This instruction will punch the contents of bits Ø - 7 of the accumulator on the high speed punch and clear the flag.

Bit 14 being set will cause accumulator to be set to all "0"s before input/output instruction is executed.