CONTROL DATA®

6000 SERIES COMPUTER SYSTEMS

ASPER/COMPASS Programming Training Guide for Peripheral Processors

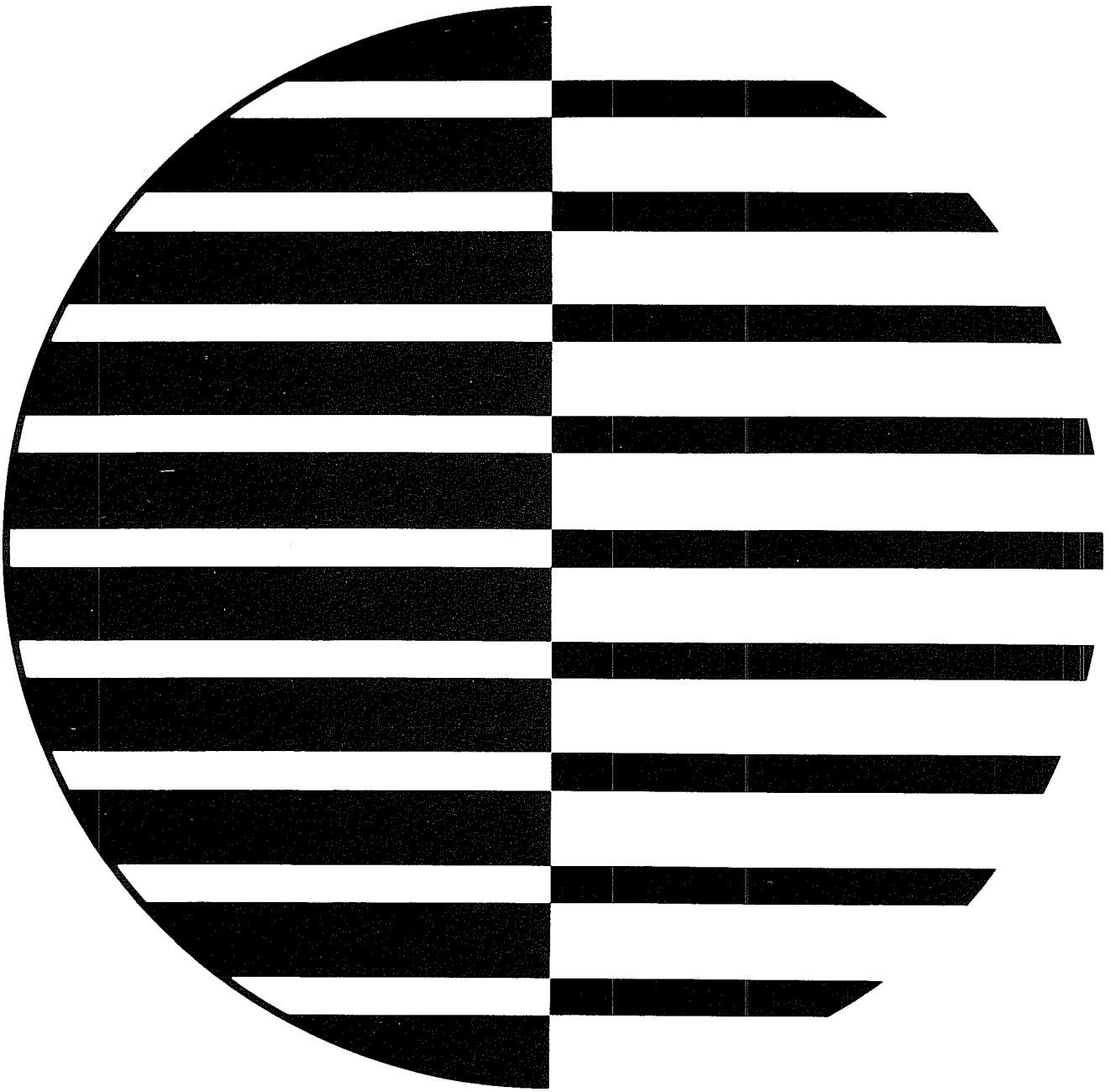


TABLE OF CONTENTS

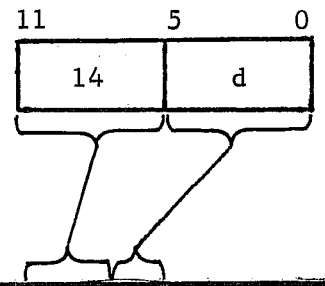
LDN

FORMATS

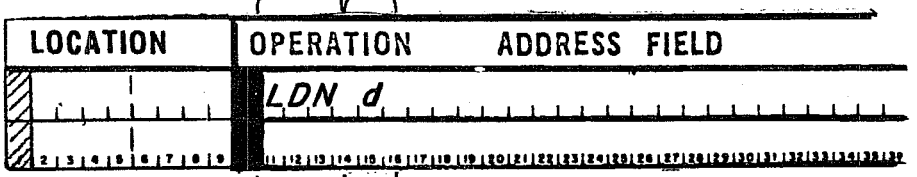
LØAD A NØ ADDRESS

(P)

MACHINE



ASPER



Constant
Symbol
Symbol ± Constant
Symbol ± Symbol

The above values must result in an octal value in the range of 00-77. The decimal equivalents are 00-63.

Mnemonic Operation Code

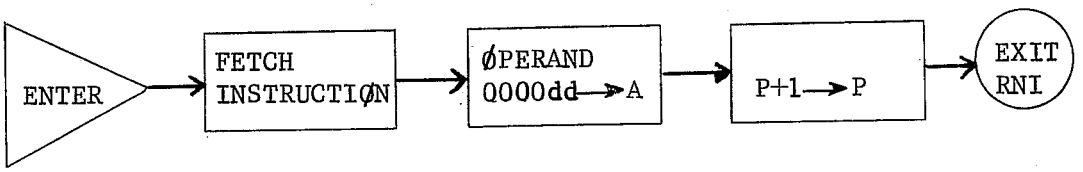
DESCRIPTION

This instruction clears the A-Register and loads d into the lower 6 bits of A. The upper 12 bits of A are set to zero. RNI @ P+1

REFERENCES :

LDN

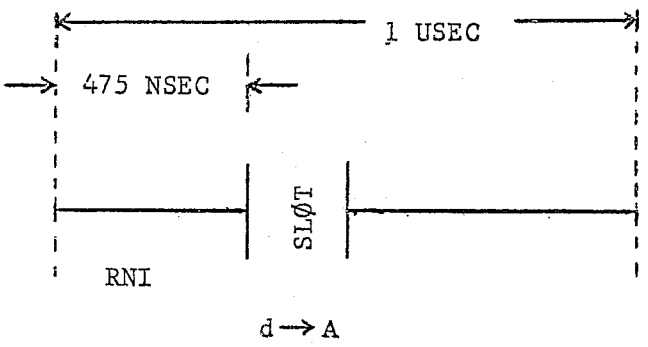
FLOW DIAGRAM



TIMING

6400/6600

1 USEC

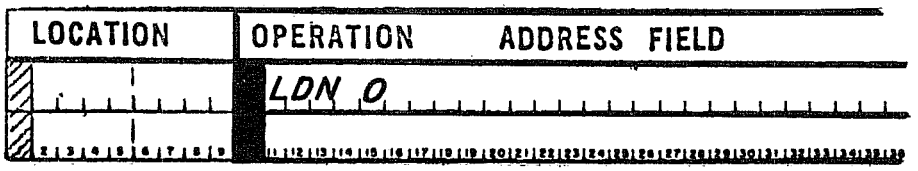


SLØT TIME = 100 NSEC

EXAMPLES

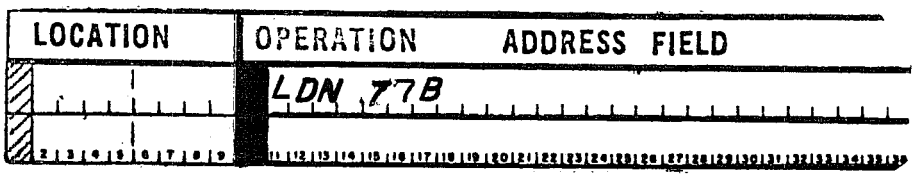
Exercise #1 - Code an instruction to clear the A-Register

ANSWER



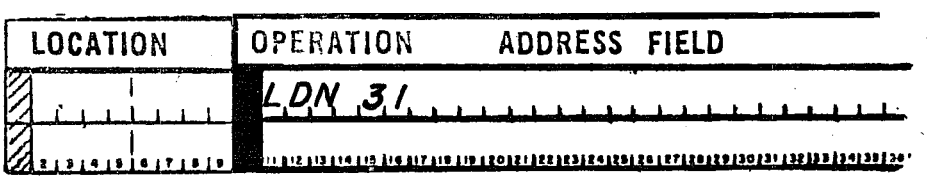
Exercise #2 - Code an instruction to set the lower 6-bits of the A-Register to ones.

ANSWER



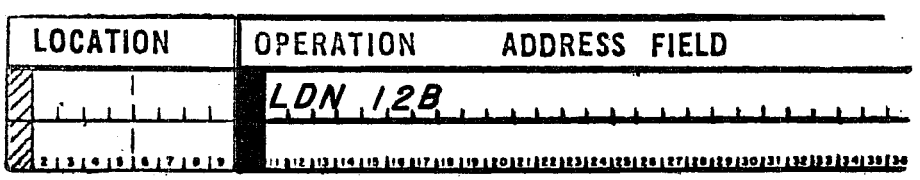
Exercise #3 - Code an instruction that will set the A-Register to 31 decimal, so it could be stored and counted down later. ($31_{10} = 37_8$)

ANSWER



Exercise #4 - Code an instruction that will set the A-Register to the constant 12, octal.

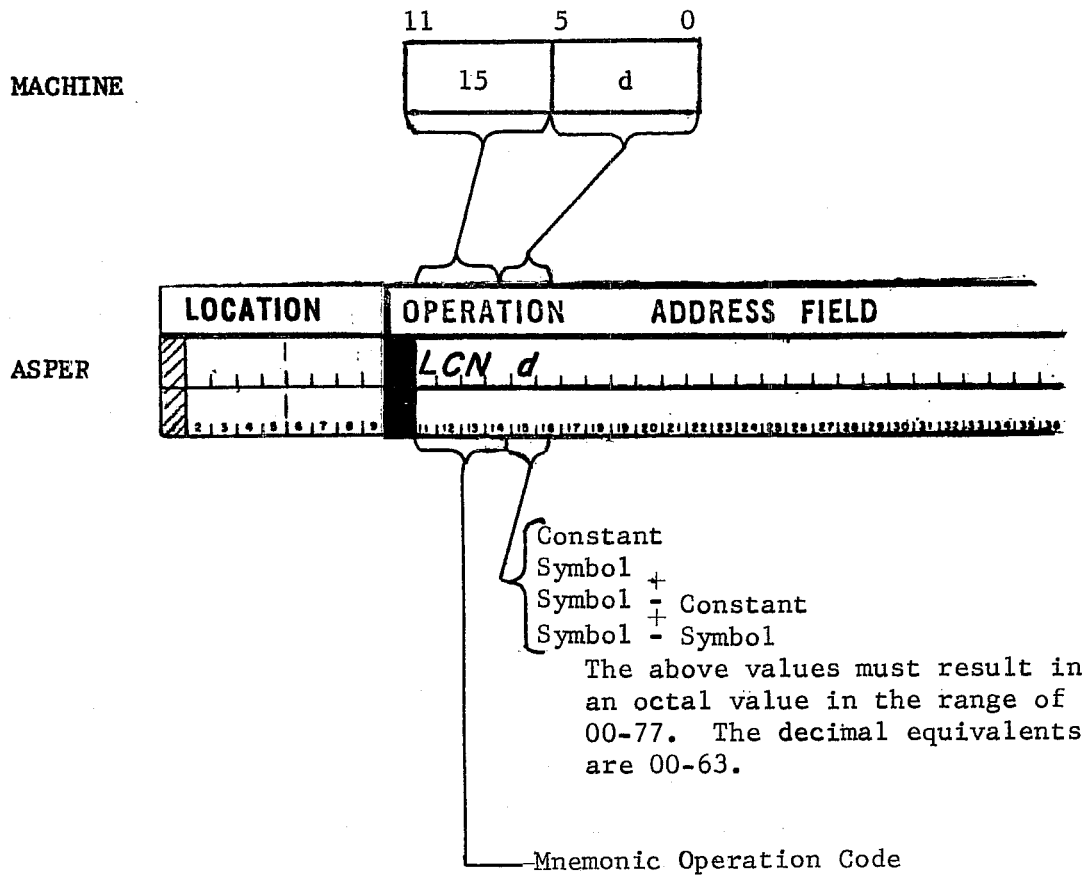
ANSWER



FORMATS

LØAD A CØMPLEMENT
NØ ADDRESS

(P)



DESCRIPTION

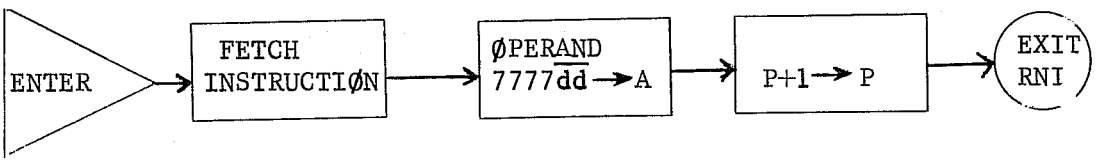
This instruction clears the A-Register and loads the complement of d into the lower 6-bits of A. The upper 12-bits are set to ones.

RNI @ P+1

REFERENCES :

LCN

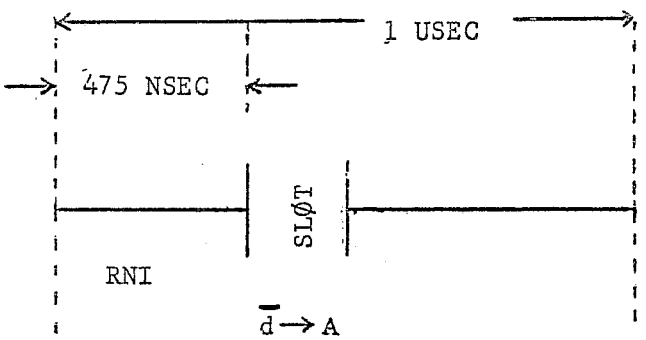
FLOW DIAGRAM



TIMING

6400/6600

1 USEC

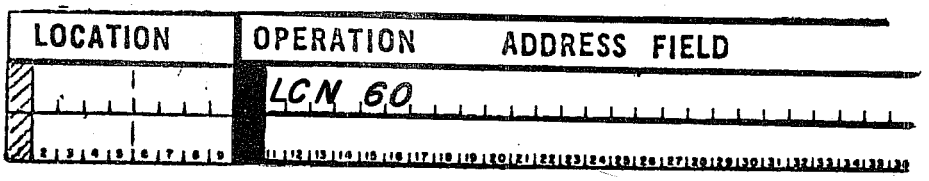


SLØT TIME = 100 NSEC

EXAMPLES

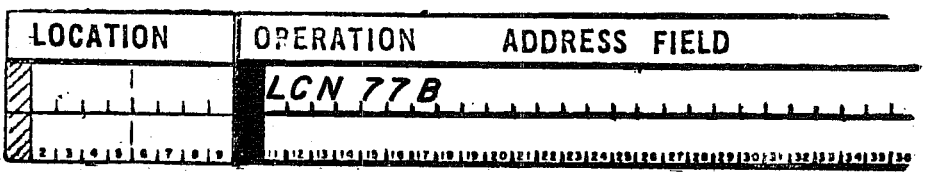
Exercise #1 - Code an instruction that will set the A-Register to a negative 60 decimal.

ANSWER



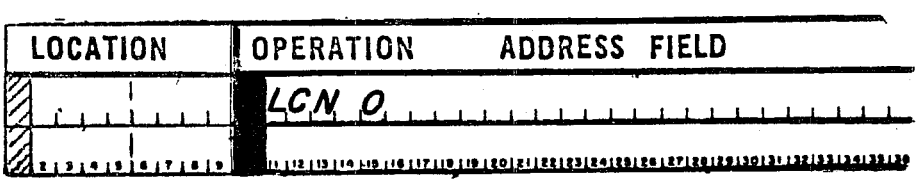
Exercise #2 - Code an instruction that will give us a mask, with the lower 6-bits cleared and the upper 12-bits ones.

ANSWER



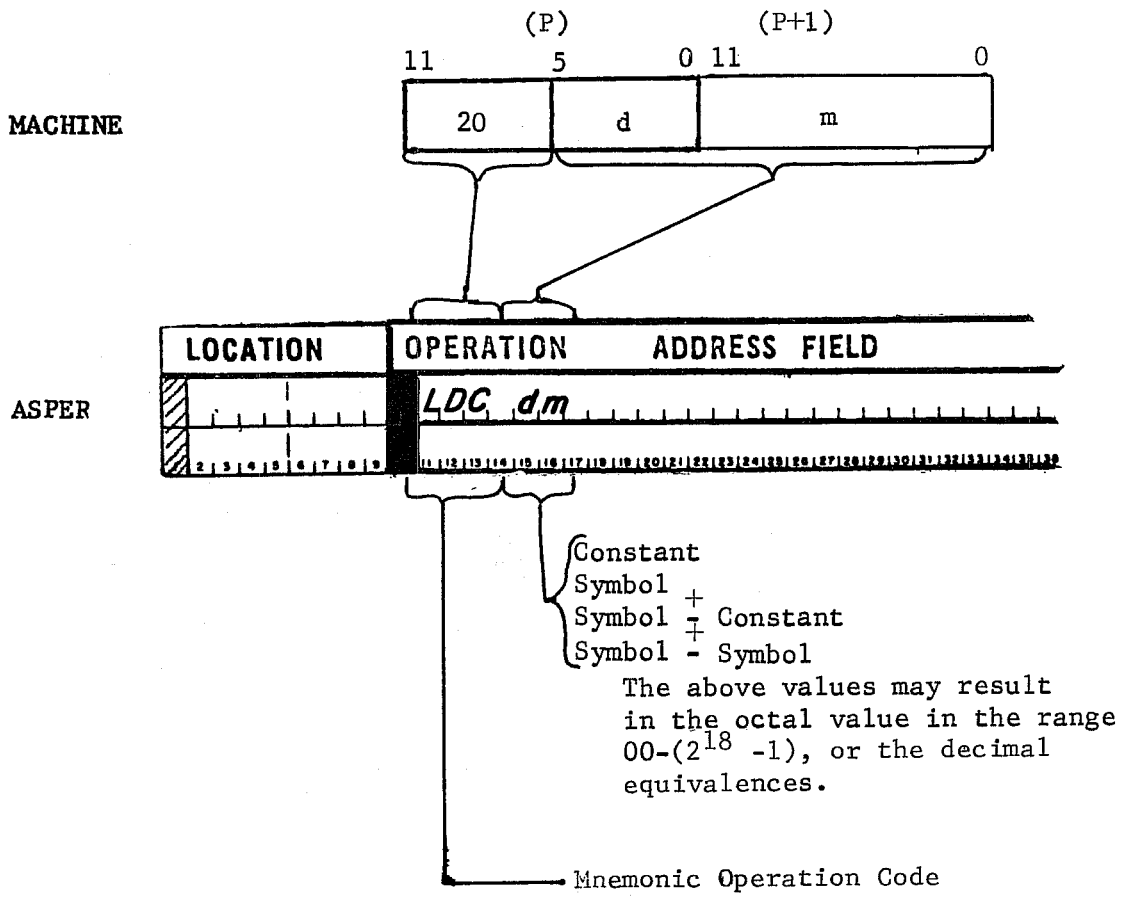
Exercise #3 - Code an instruction to set the A-Register to all ones.

ANSWER



FORMATS

LØAD A CØNSTANT



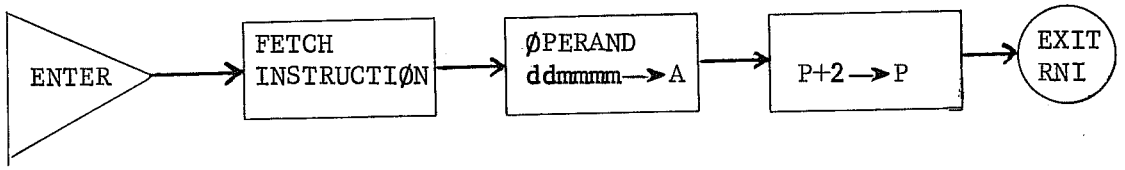
DESCRIPTION

This instruction clears the A-Register and loads the 18-bit quantity consisting of d as the upper 6 bits and m as the lower 12 bits. RNI @ P+2

REFERENCES :

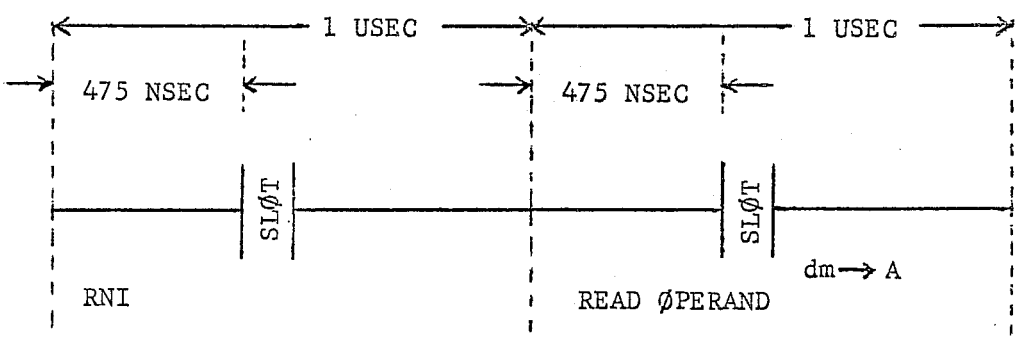
LDC

FLOW DIAGRAM



TIMING

6400/6600 2 USEC

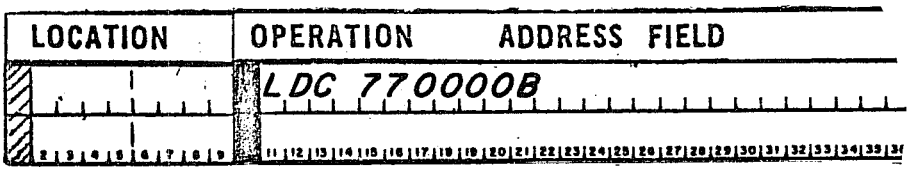


SLØT TIME = 100 NSEC

EXAMPLES

Exercise #1 - Code an instruction that will set the upper 6-bits of the A-Register to ones, in order that two 12-bit quantities may be added later (with end around carries).

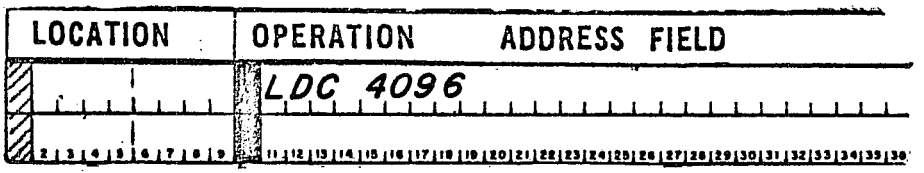
ANSWER



Exercise #2 - Code an instruction to load the A-Register with the constant 4096_{10}, which could next be used for an input/output instruction.

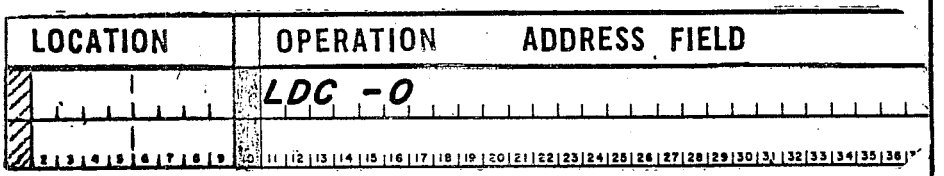
($4096_{10} = 10000_8$)

ANSWER



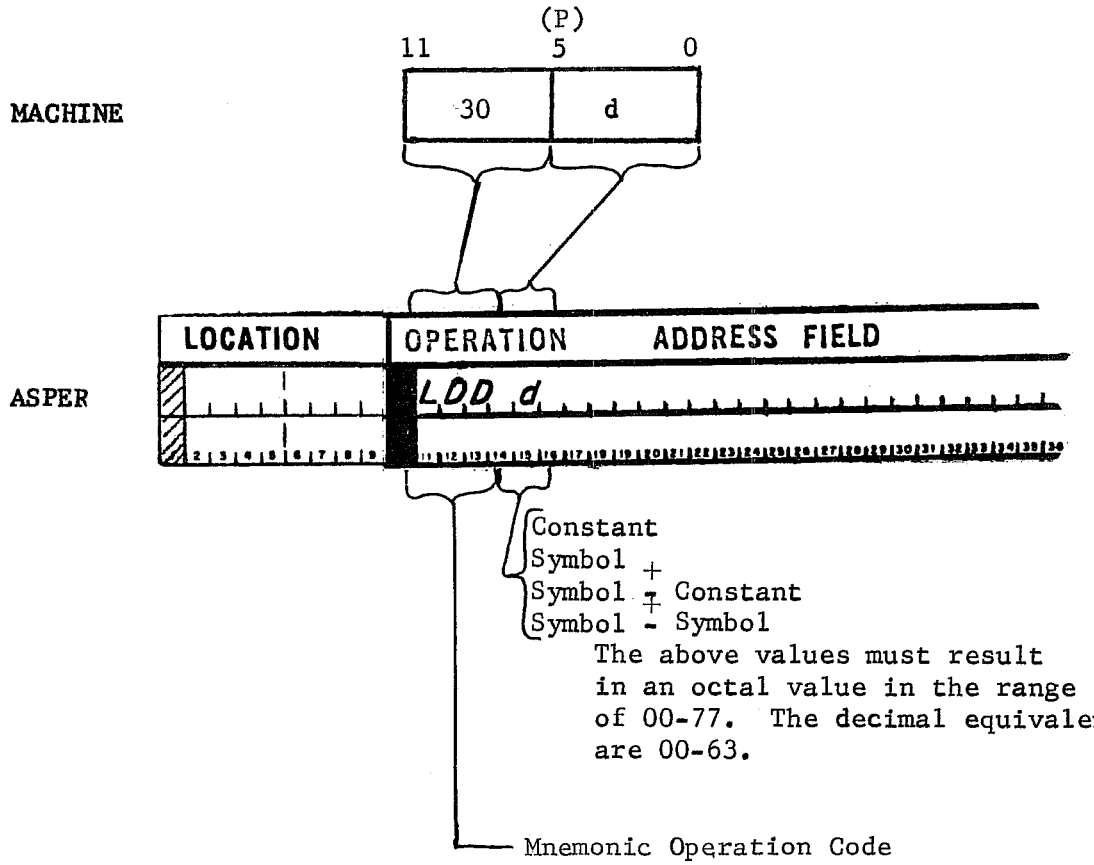
Exercise #3 - Code an instruction that will load the A-Register with all ones into the A-Register.

ANSWER



FORMATS

LØAD A DIRECT



DESCRIPTION

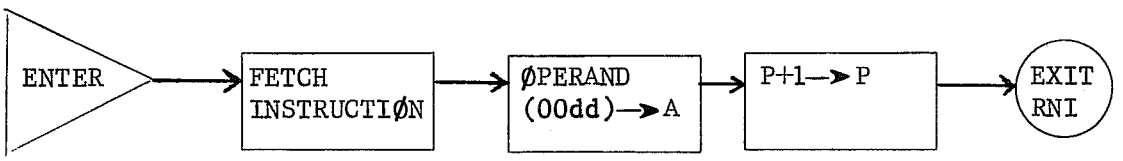
This instruction clears the A-Register and loads the contents of location d into the lower 12 bits of A. The upper 6 bits of A are set to zero.

RNI @ P+1.

REFERENCES :

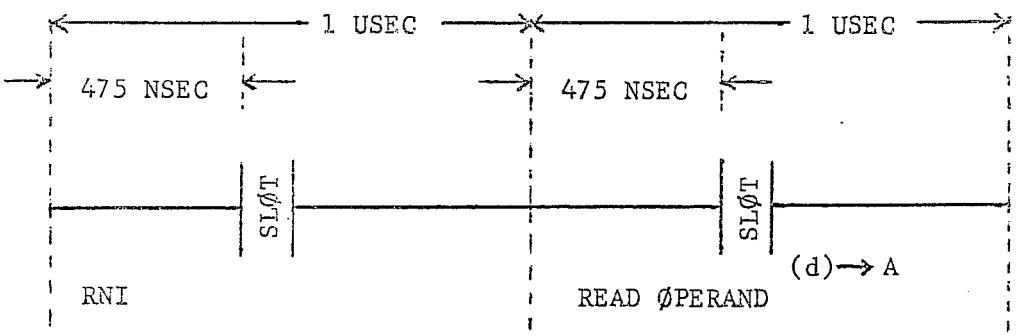
LDD

FLOW DIAGRAM



TIMING

6400/6600 2 USEC



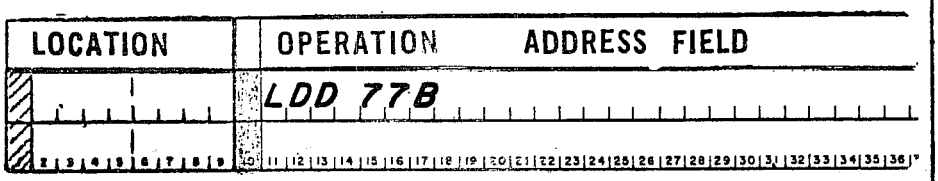
SLØT TIME = 100 NSEC

EXAMPLES

Exercise #1 - Code an instruction to load 7654 into the A-Register

7654 is in M.L. 0077 (Øctal).

ANSWER

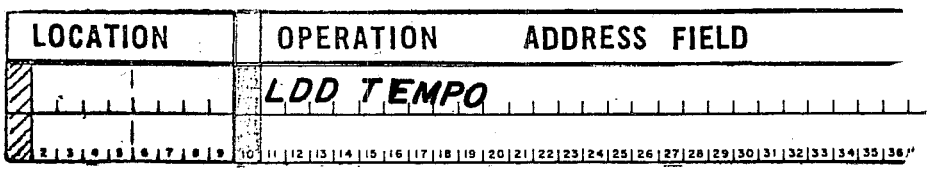


Exercise #2 - Code an instruction to load the contents of Tempo.

Tempo = 0070, but could have been any value $00\text{-}77_8$.

(Tempo) = 0010, but could have been an octal value in the range of $(2^{12} - 1)$.

ANSWER



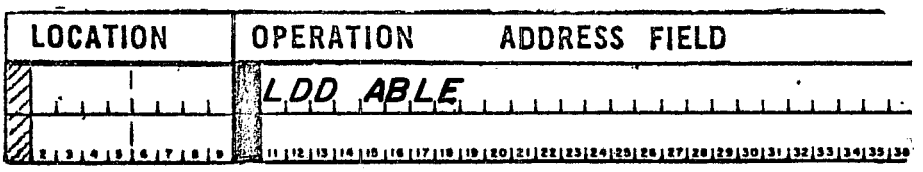
Exercise #3 - Code an instruction to load 1234 into the A-Register.

The Øperand 1234 is in a location called ABLE, which happens to equal 0000.

ABLE = 0000, but could have been any octal value 00-77.

(ABLE) = 1234, but could have been an octal value in the range of $(2^{12} - 1)$.

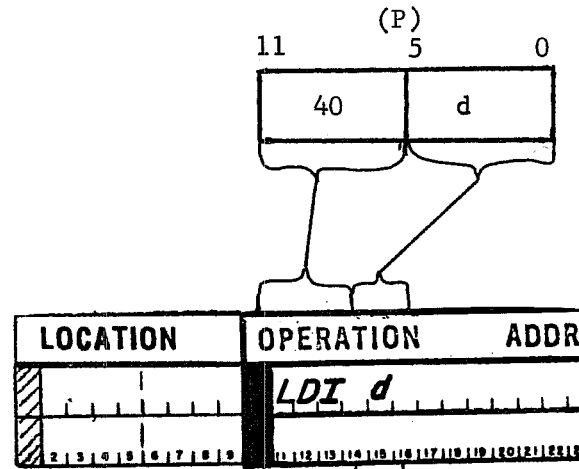
ANSWER



FORMATS

LØAD A INDIRECT

MACHINE



ASPER

Constant
Symbol
Symbol ± Constant
Symbol ± Symbol

The above values must result in an octal value in the range of 00-77. The decimal equivalents are 00-63.

Mnemonic Operation Code

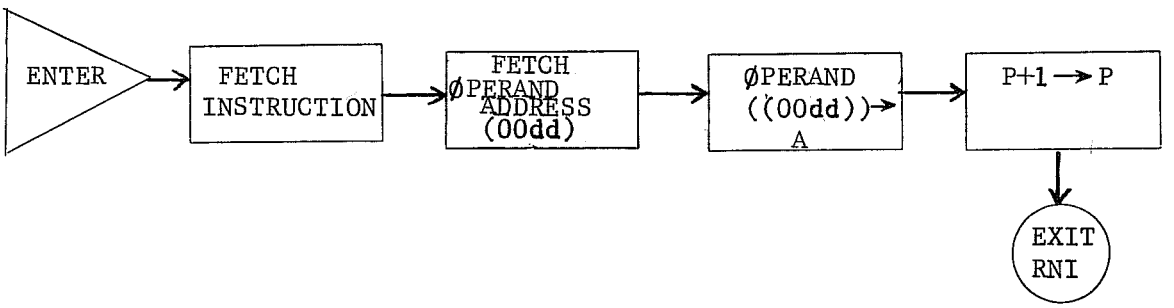
DESCRIPTION

This instruction clears the A-Register and loads into A the 12-bit quantity obtained by indirect addressing, ((d)) → A. The upper six bits of A are zero. Location d is read out of memory, and the word obtained is used as the operand address. RNI @ P+1

REFERENCES :

LDI

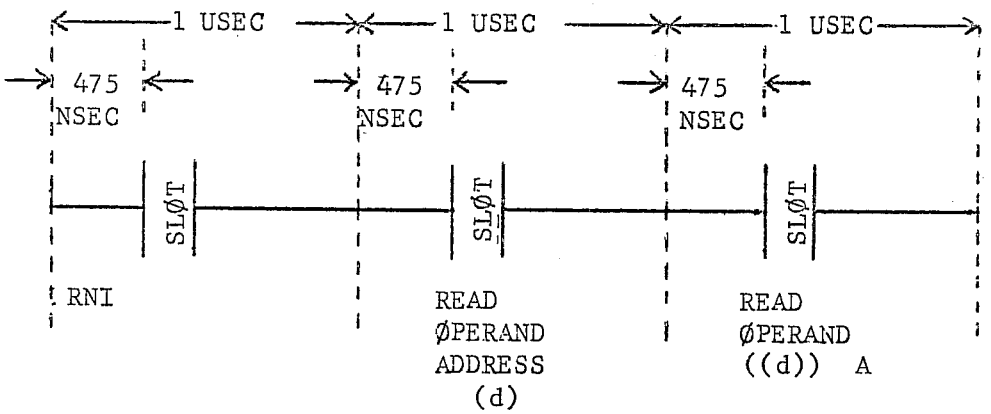
FLOW DIAGRAM



TIMING

6400/6600

3 USEC



SLØT TIME = 100 NSEC

EXAMPLES

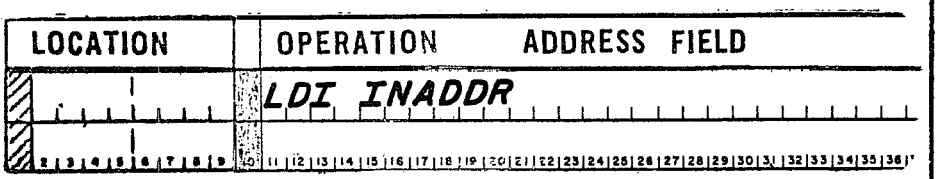
Exercise #1 - Code an instruction that loads the A-Register, and uses INDADDR as a 1st reference, and ADDR as a 2nd reference to obtain the operand 4311.

INDADDR = 50, but could have been any octal value 00-77.

(INDADDR) = ADDR = 3024, but could have any value $0000\text{-}7777_8$.

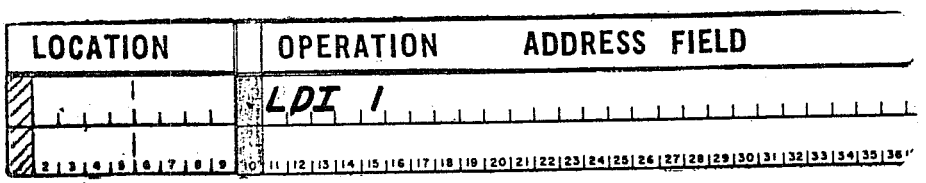
(ADDR)= 4311, and again could have been any value 0000-7777.

ANSWER



Exercise #2 - Code an instruction that loads the A-Register, and uses location 0001 as its 1st reference to obtain the operand address. The operand address is equal to 6351_8, and the operand equals 0100_8.

ANSWER

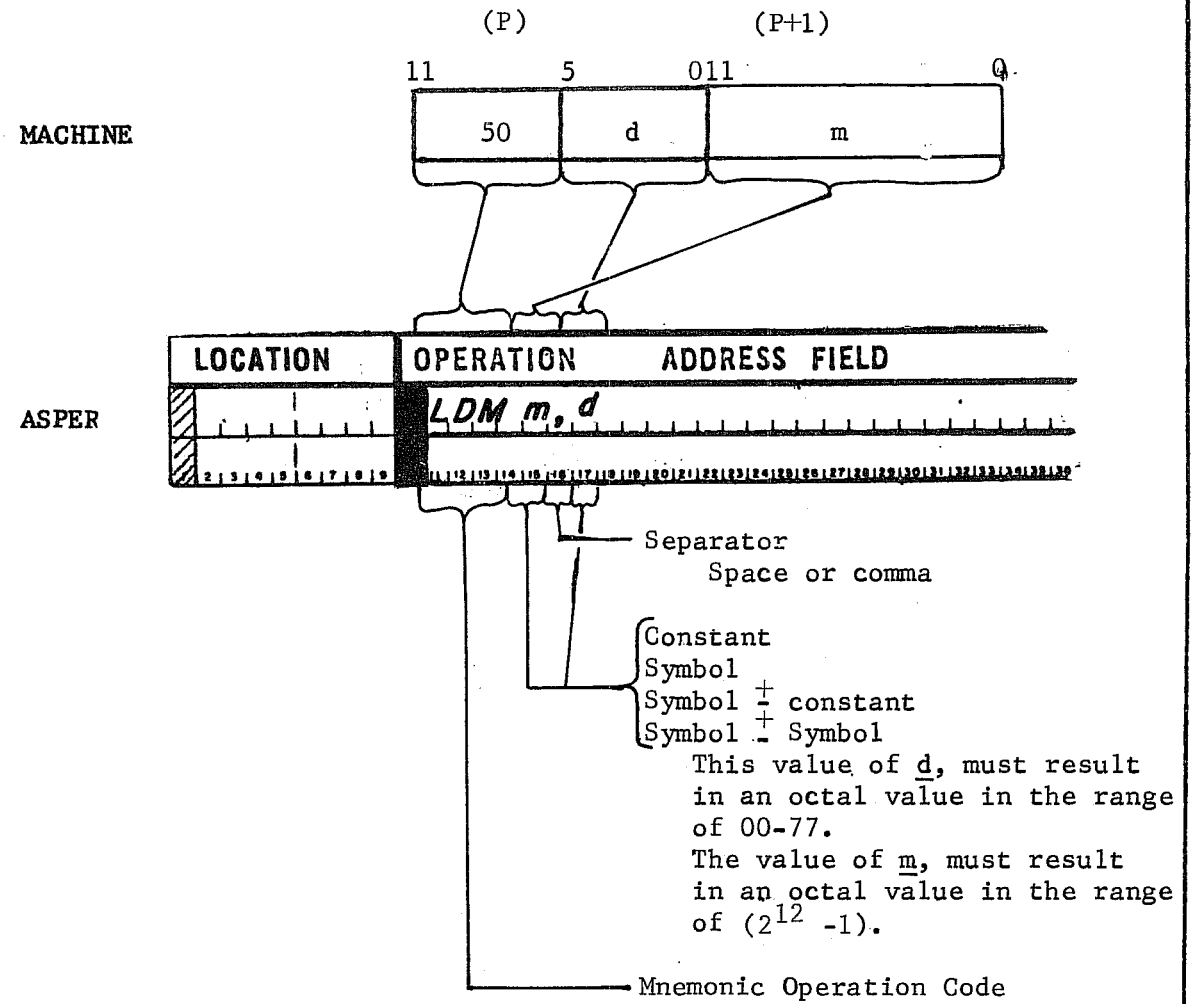


$(0001) = 6351_8$

$(6351) = 0100_8$

FORMATS

LØAD A MEMØRY INDEX



DESCRIPTION

This instruction clears the A-Register and loads a 12 bit operand obtained by indexed addressing into the lower 12 bits of A. The upper 6 bits of A are set to zero.

Note: If d=0, the operand address is simply m.

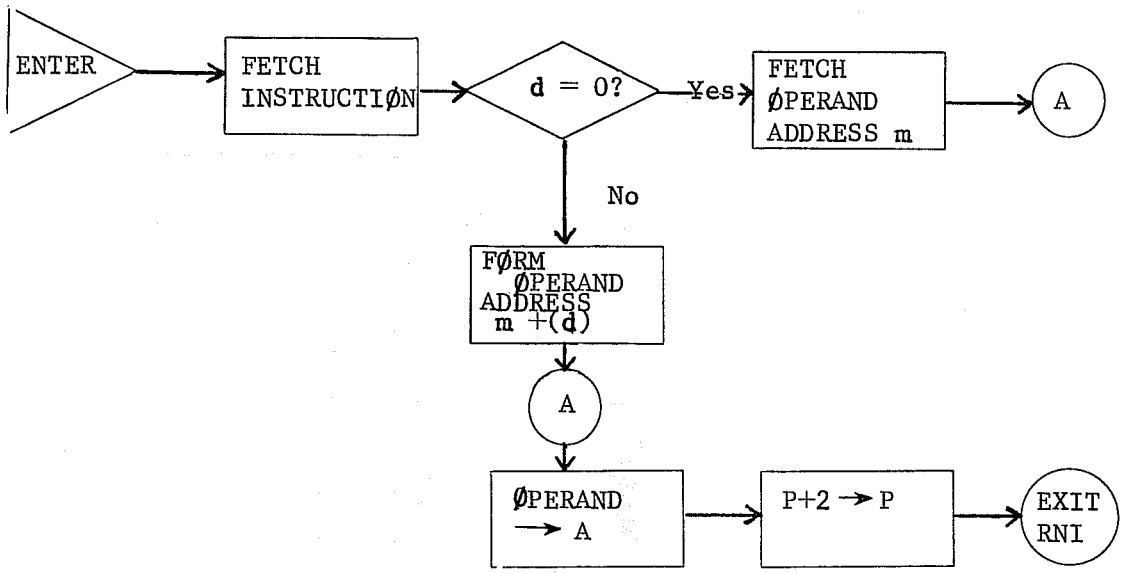
If d ≠ 0, then m plus the contents of location d,

m + (d) is the operand address. Thus the contents of d may be used as an index quantity to modify operand addresses. RNI @ P+2

REFERENCES :

LDM

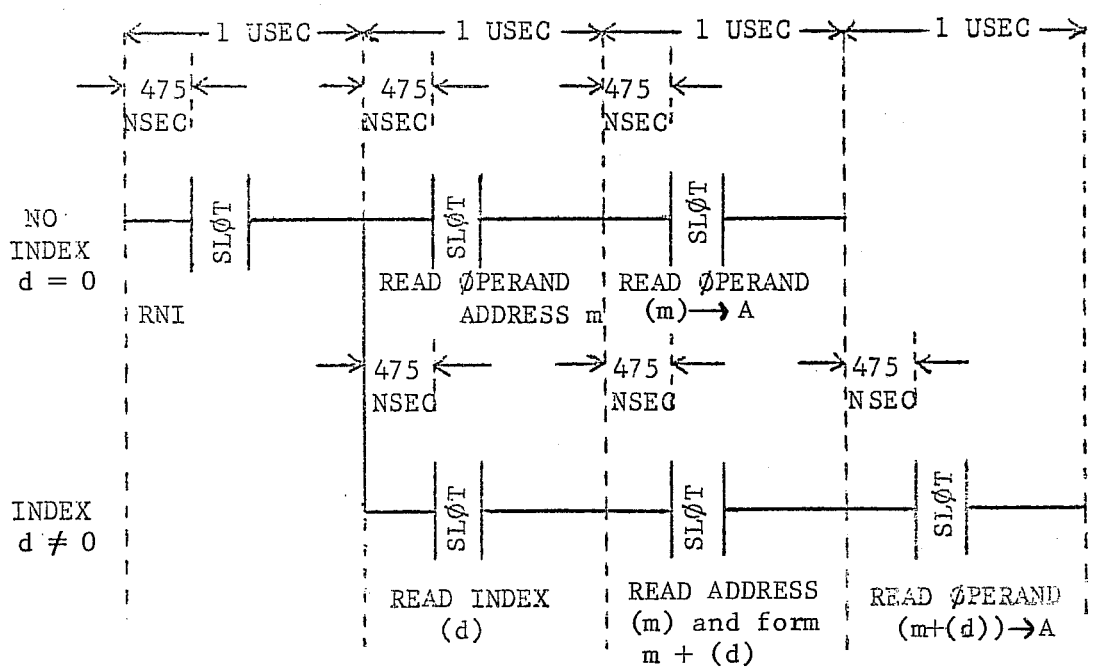
FLOW DIAGRAM



TIMING

6400/6600

3-4 USEC



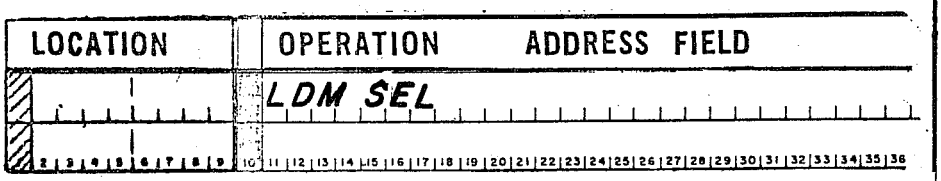
SLØT TIME = 100 NSEC

EXAMPLES

Exercise #1 - Code an instruction that will load the A-Register with an operand 0601, whose address is called SEL. Use the memory index instruction without indexing. SEL = 6322, but could have been equal to any value in the range (2^{12} -1).

(SEL) = 0601

ANSWER

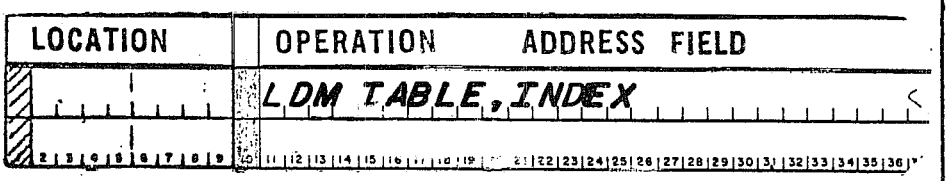


Exercise #2 - Code an instruction that will load the A-Register with the first a series of numbers from a table. Using the memory index instruction, the basic address will be called TABLE, and the index address called INDEX.

INDEX = 70, but could have been any octal value 01-77.

TABLE = 5000, but could have been any value in the range of 2^{12} -1.

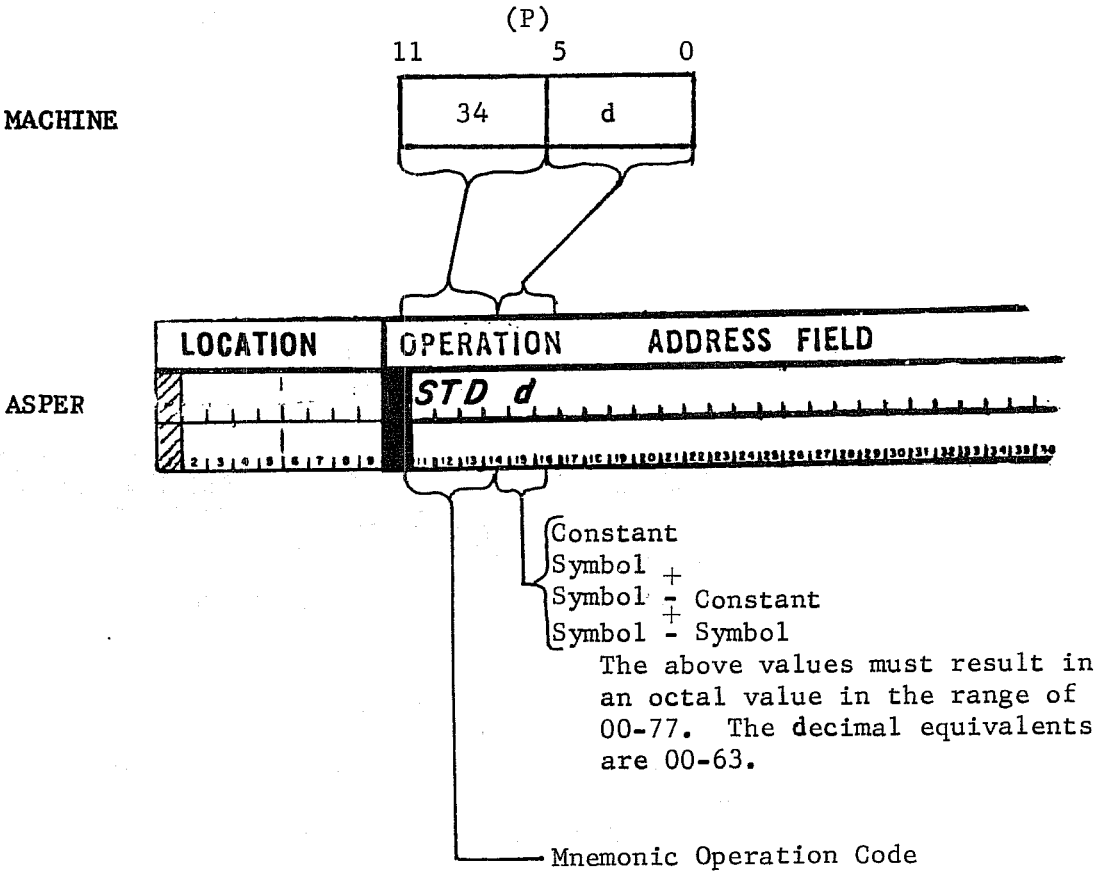
ANSWER



Note: The contents of 70 should be zero, the 1st time a load is made, and then updated by one for each new reference.

FORMATS

STØRE A DIRECT



DESCRIPTION

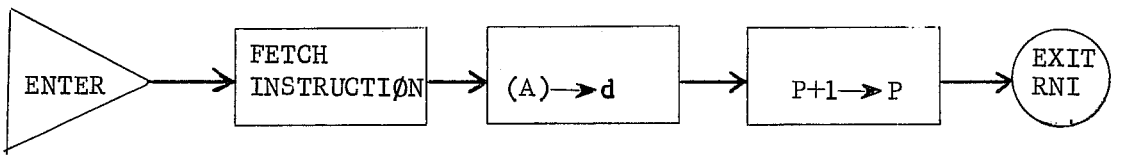
This instruction stores the lower 12 bits of the A-Register into location d. The contents of A are not altered.

RNI @ P+1

REFERENCES :

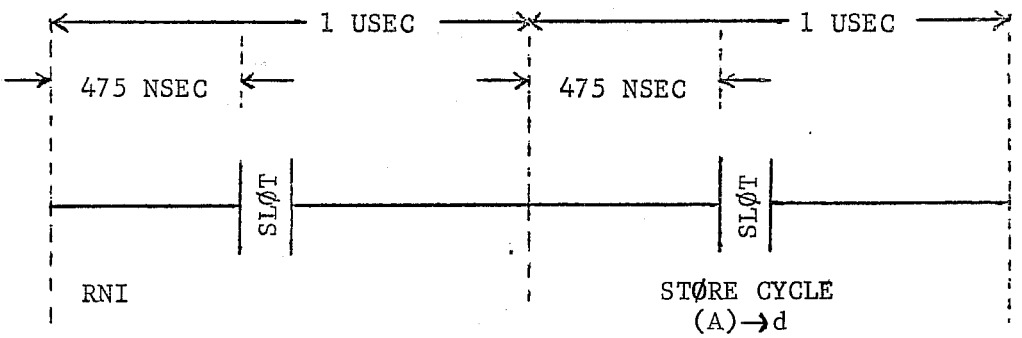
STD

FLOW DIAGRAM



TIMING

6400/6600 2 USEC



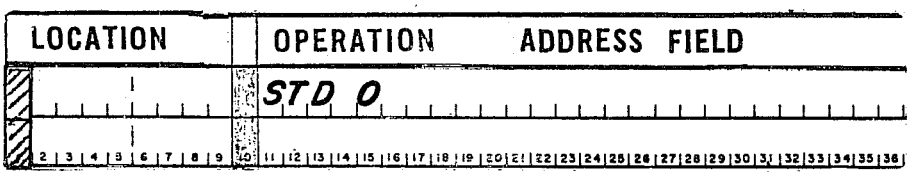
SLØT TIME = 100 NSEC

EXAMPLES

Exercise #1 - Code an instruction to store the contents of the A-Register, which is 765432, in M.L. 0000.

Note: At the completion of the instruction, M.L. 0000 will contain 5432.

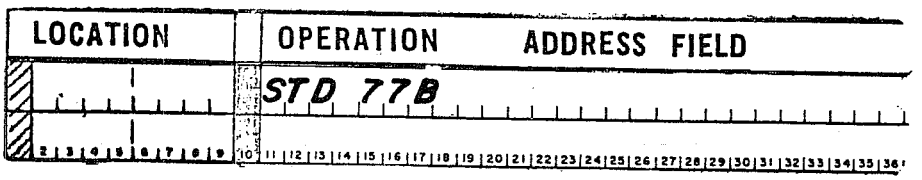
ANSWER



Exercise #2 - Code an instruction to store the contents of the A-Register which is 000000, in M.L. 0077.

Note: At the completion of the instruction, M.L. 0077 will contain 0000. This effectively clears that M.L.

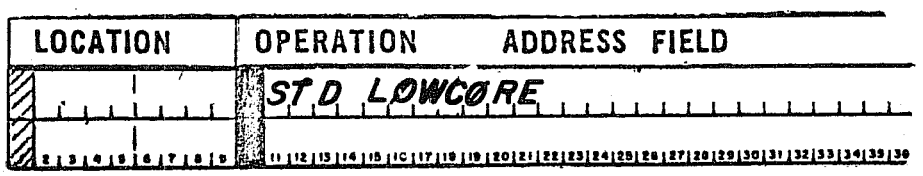
ANSWER



Exercise #3 - Code an instruction that will save the contents of the A-Register, and store it in some location $00\text{-}77_8$. In this example the location will be called LØWCØRE.

LØWCØRE = 10

ANSWER

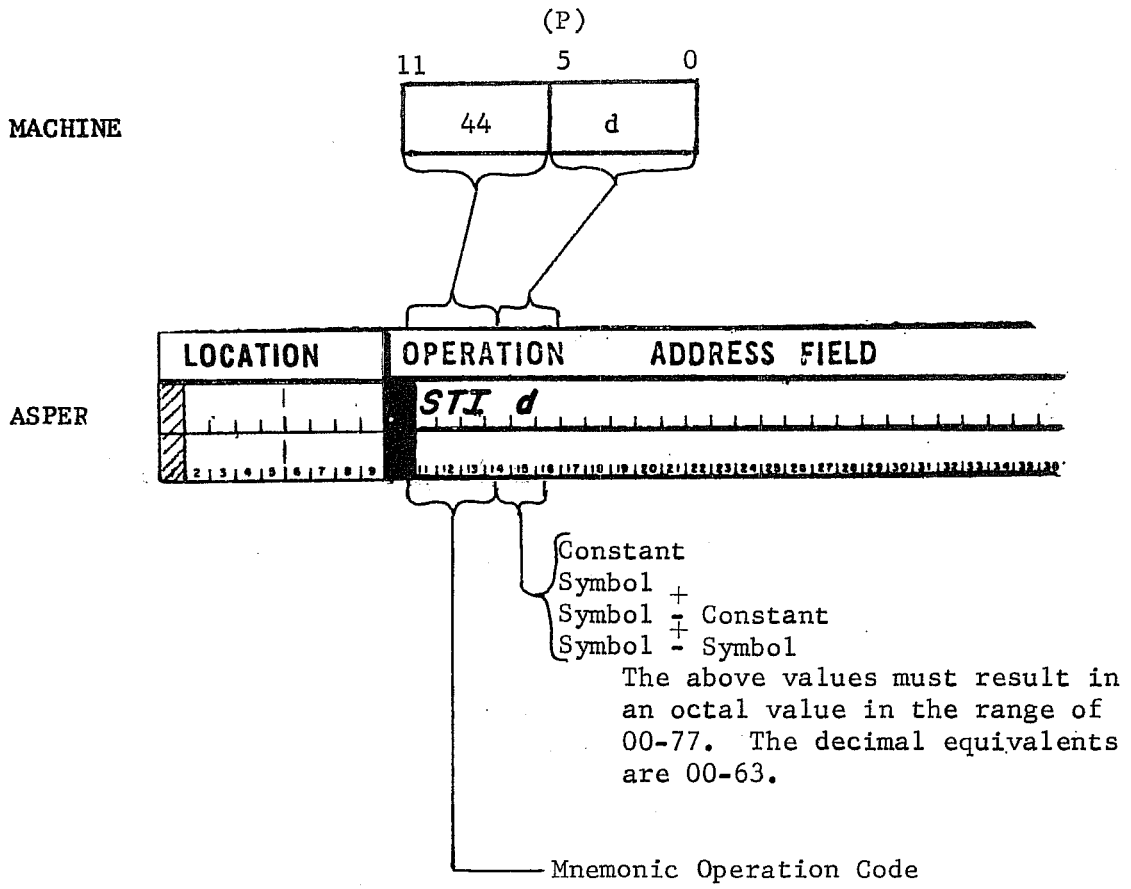


Note: Only the lower 12-bits of the A-Register will be stored in memory.

The A-Register remains unaltered.

FORMATS

STØRE A INDIRECT



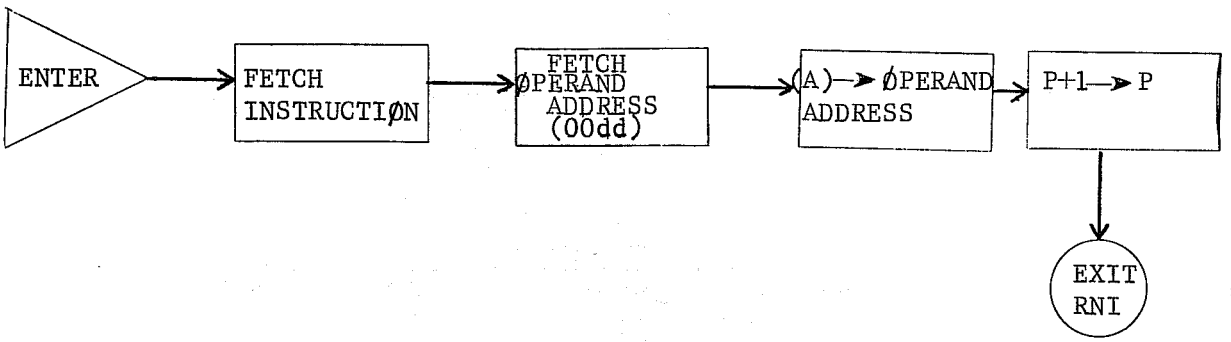
DESCRIPTION

This insturction stores the lower 12 bits of the A-Register into the location specified by the contents of location d, (A) → ((d)). The contents of A are not altered. RNI @ P+1.

REFERENCES :

FLOW DIAGRAM

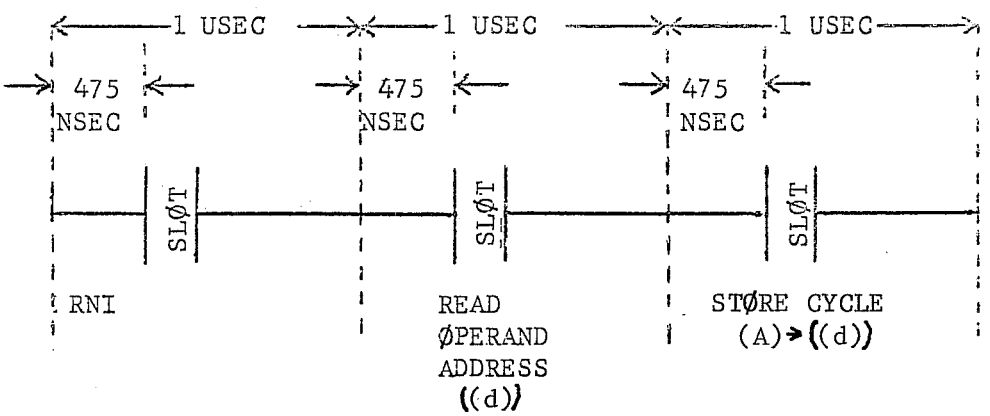
STI



TIMING

6400/6600

3 USEC



SLØT TIME = 100 NSEC

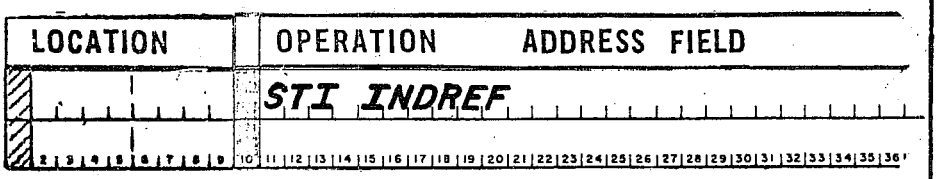
EXAMPLES

Exercise #1 - Code an instruction that stores the contents of the A-Register, and uses INDREF as a 1st reference and STØRE as a 2nd reference.

A-Register contains 514233.

INDREF = 00, but could have been any octal value 00-77.

(INDREF) = STØRE = 6071, but could have been any octal value in the range (2^{12} -1).



Note: Only the 4233 of A will be stored in location 6071.

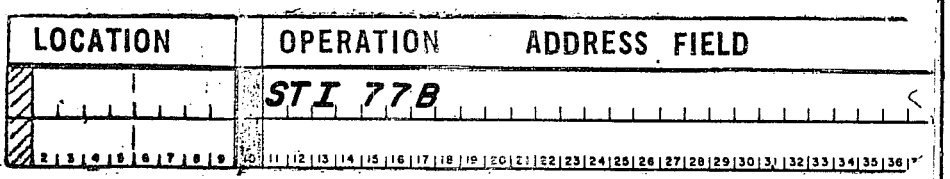
Exercise #2 - Code an instruction that stores the contents of the A-Register, and uses M.L. 0077 as its 1st reference to obtain the operand address.

A-Register = 123456

(0077) = 3347

(3347) = 1000

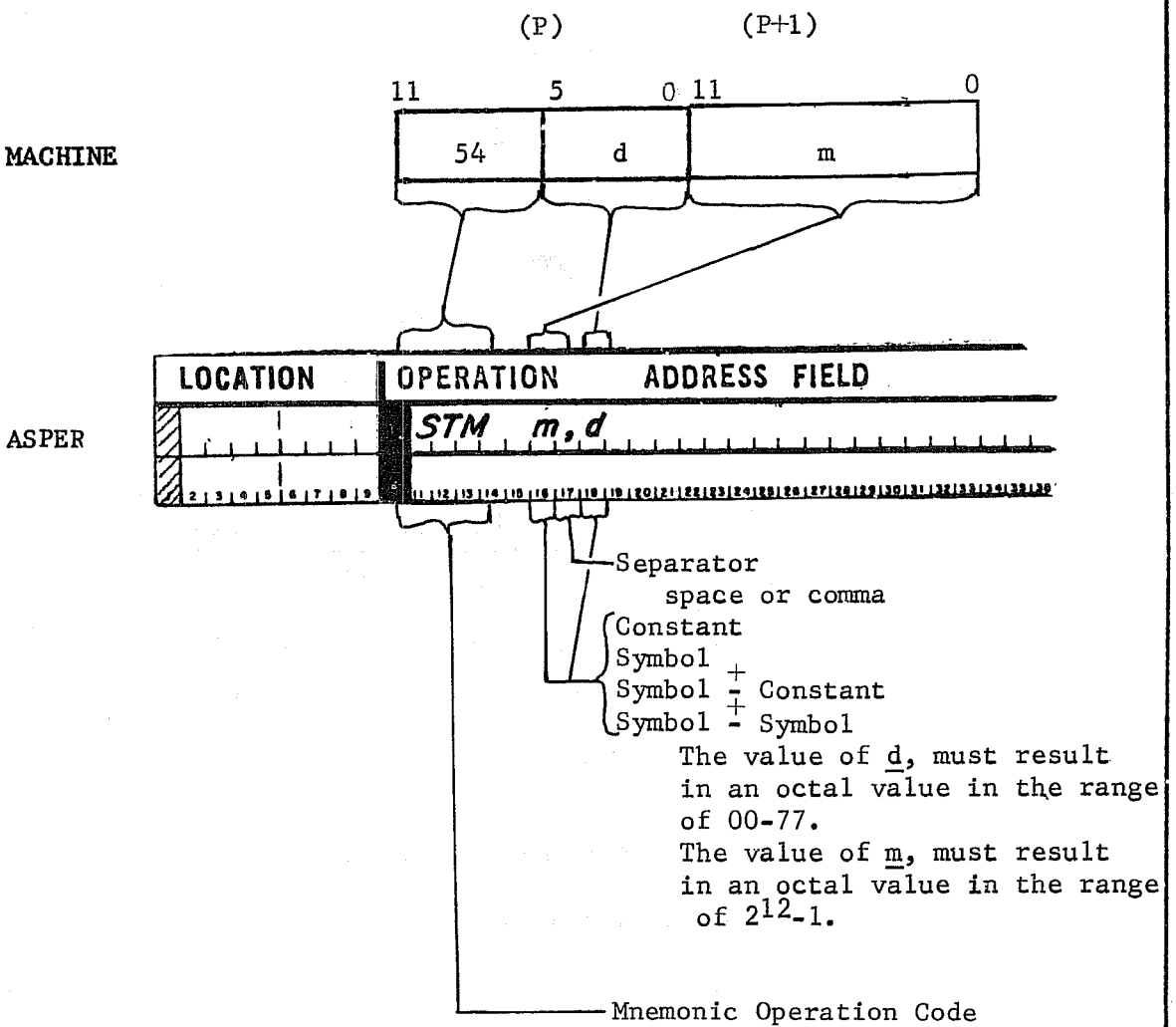
ANSWER



Note: At the completion of this instruction, memory location 3347 will contain 3456, and 1000 will be destroyed.

FORMATS

STØRE A MEMØRY INDEX



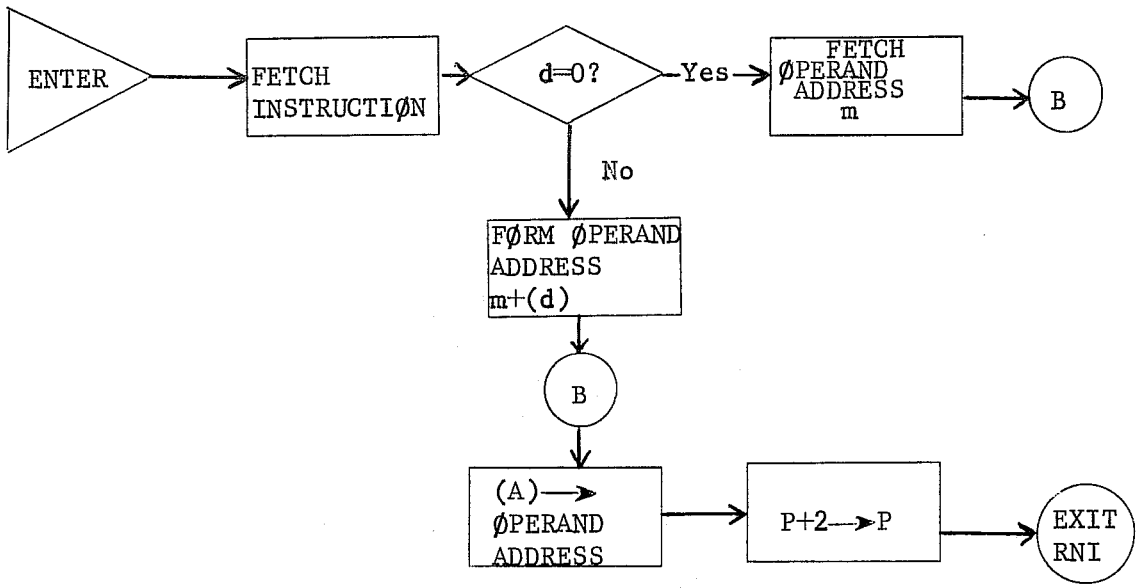
DESCRIPTION

This instruction stores the lower 12 bits of the A-Register in the location determined by indexed addressing. The contents of A are not altered.

Note: If d=0, the operand address is simply m. If d ≠ 0, the m plus the contents of location d, m+(d), is the operand address. Thus the contents of d may be used as an index quantity to modify operand addresses. RNI @ P+2

REFERENCES :

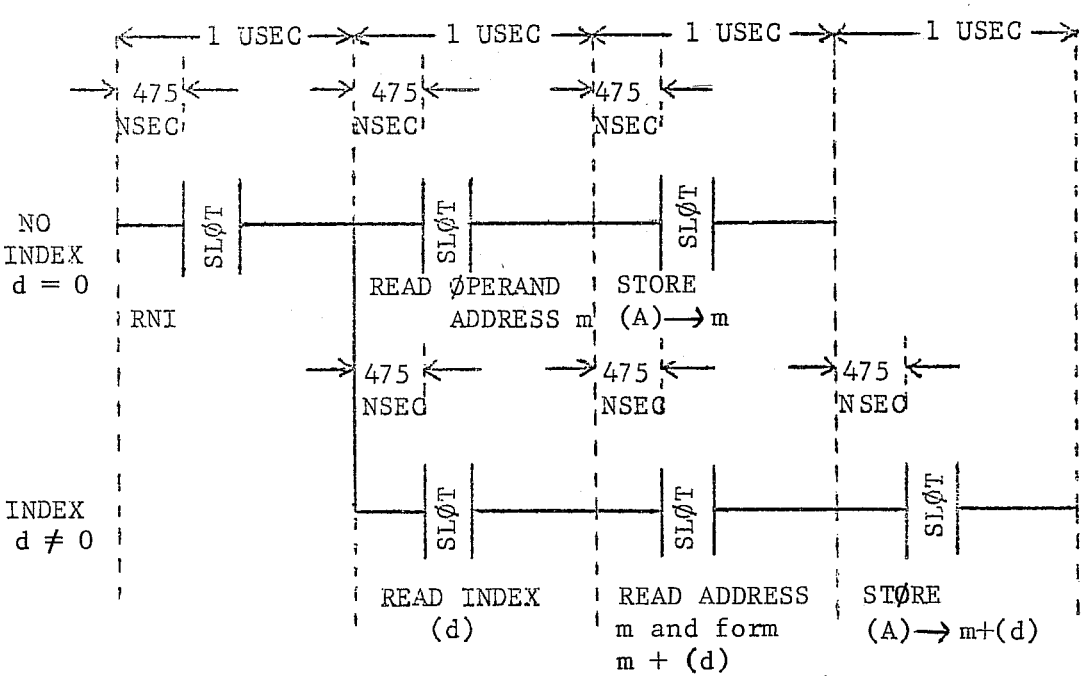
FLOW DIAGRAM



TIMING

6400/6600

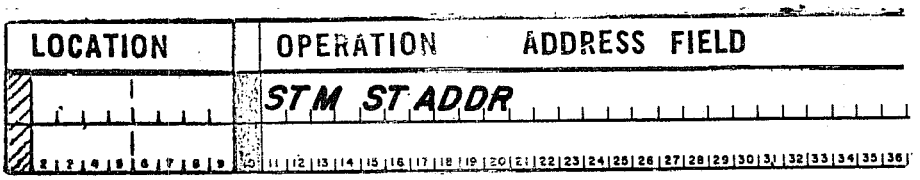
3-4 USEC



SLØT TIME = 100 NSEC

Exercise #1 - Using memory index, with no indexing, code an instruction to store the contents of the A-Register in some location called STADDR.

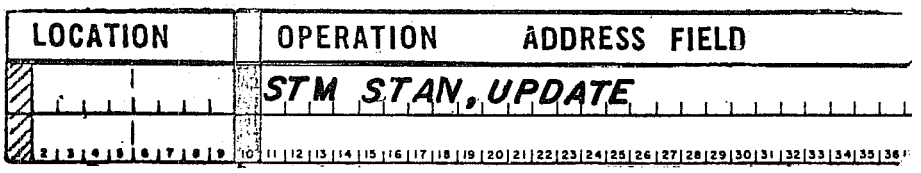
ANSWER



Exercise #2 - A series of numbers may be coming into the computer. Using Memory Index, with indexing, code an instruction to store the numbers in sequential locations beginning the location called STAN. Use the location called UPDATE, as an index.

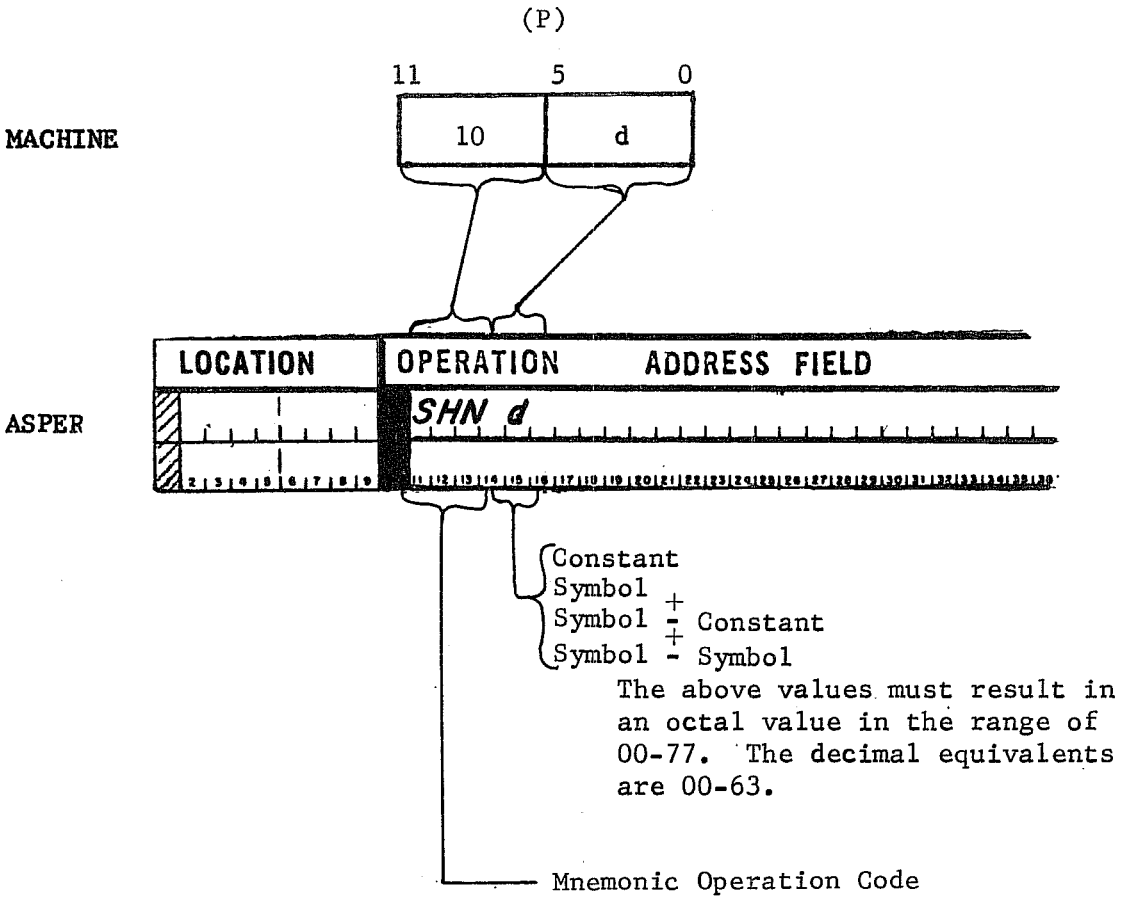
Assume UPDATE = 70 and contains 0000 at 1st STAN = any value (see P. 9-0)

ANSWER



FORMATS

SHIFT A NØ ADDRESS



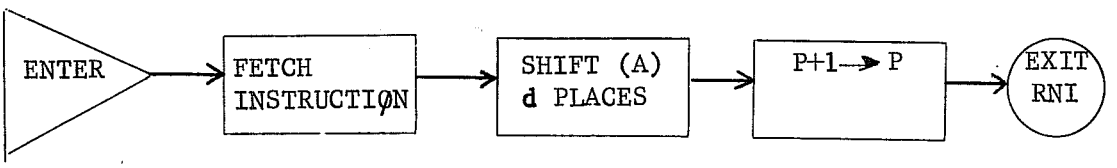
DESCRIPTION

This instruction shifts the contents of the A-Register (which is 18-bits) right or left d places. If d is positive ($00\text{-}37_8$), the shift is left and circular; if d is negative ($40\text{-}77_8$), A is shifted right, end off with no sign extension. RNI @ P+1.

REFERENCES :

SHN

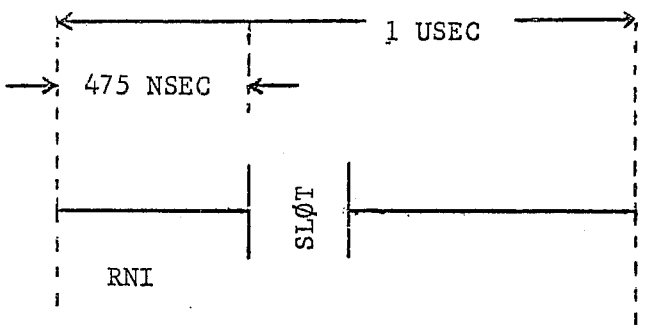
FLOW DIAGRAM



TIMING

6400/6600

1 USEC



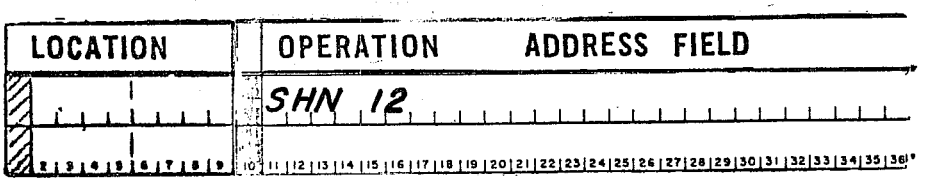
SLØT TIME = 100 NSEC

EXAMPLES

Exercise #1 - Code an instruction to shift the contents of the A-Register left 12_{10} places. This will place the lower 6-bits of A in the upper 6-bits of A.

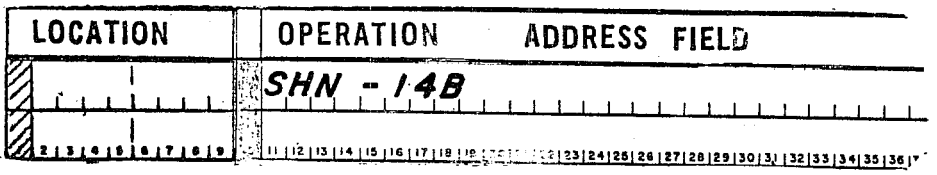
Note: A left shift of more than 17_{10} would be impractical. Also, note each left shift would be a multiplication by 2.

ANSWER



Exercise #2 - There is an 18-bit quantity in the A-Register we wish to store. The lower 12-bits may be stored without any trouble. However, we must do some manipulation with the A-Register before the upper 6-bits can be stored. Code an instruction to shift the contents of the A-Register right 12 places.

ANSWER



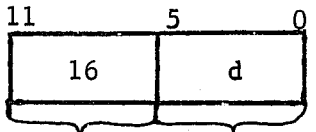
Note: A right shift of more than 18_{10} would be impractical also, note each right shift would be a division by 2.

FORMATS

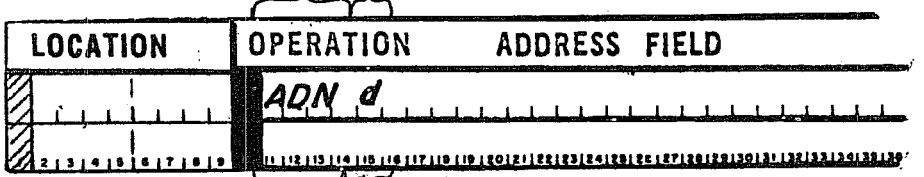
ADD NØ ADDRESS

(P)

MACHINE



ASPER



Constant
Symbol
Symbol ± Constant
Symbol ± symbol

The above values must result in an octal value in the range of 00-77. The decimal equivalents are 00-63.

Mnemonic Operation Code

DESCRIPTION

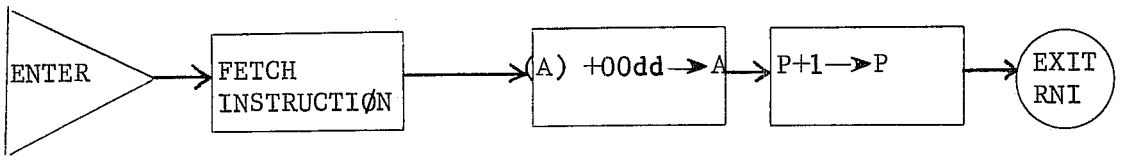
This instruction adds the 6-bit positive quantity d (0000dd) to the contents of the A-Register (which is 18-bits).

RNI @ P+1

REFERENCES :

ADN

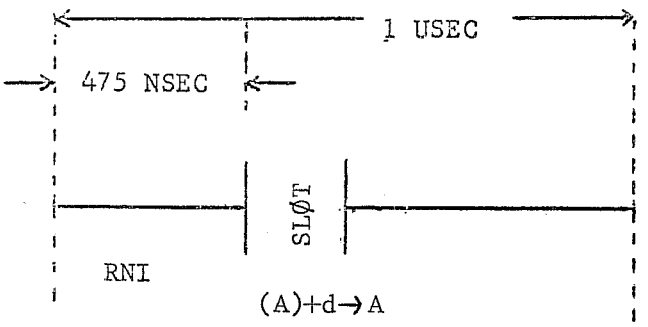
FLOW DIAGRAM



TIMING

6400/6600

1 USEC

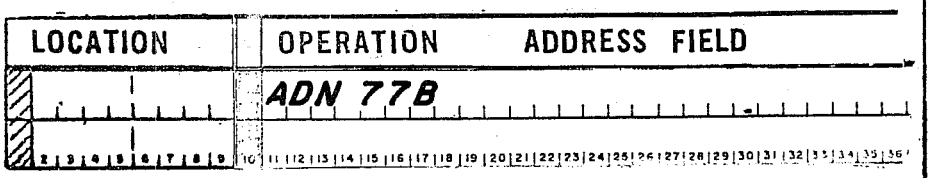


SLØT TIME = 100 NSEC

EXAMPLES

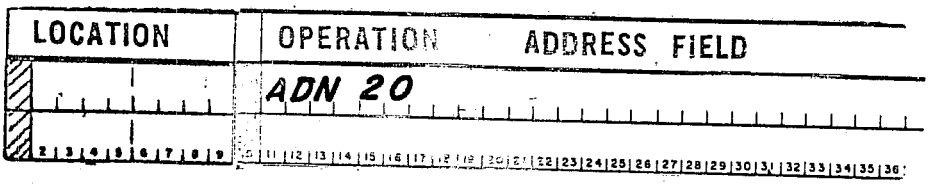
Exercise #1 - Code an instruction to add the constant 77_8 to the A-Register.

ANSWER



Exercise #2 - Code an instruction to add the constant 20_{10} to the A-Register.

ANSWER

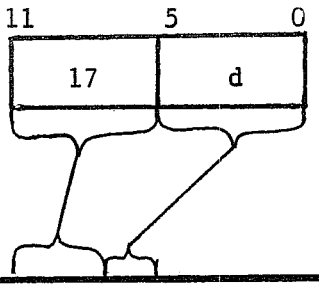


FORMATS

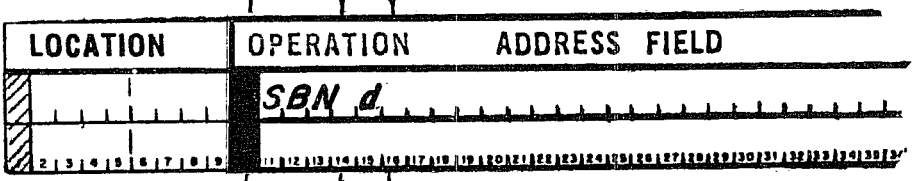
SUBTRACT NØ ADDRESS

(P)

MACHINE



ASPER



Constant
Symbol
Symbol ± Constant
Symbol - Symbol

The above values must result in an octal value in the range of 00-77. The decimal equivalents are 00-63.

Mnemonic Operation Code

DESCRIPTION

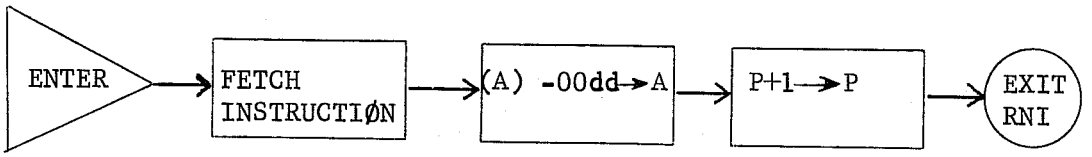
This instruction subtracts the 6-bit positive quantity d (0000dd) from the contents of the A-Register (which is 18-bits).

RNI @ P+1.

REFERENCES :

SBN

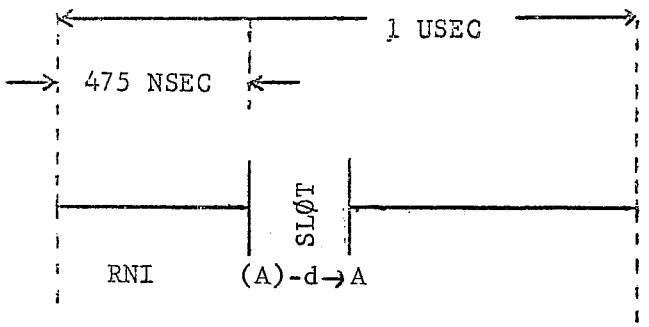
FLOW DIAGRAM



TIMING

6400/6600

1 USEC



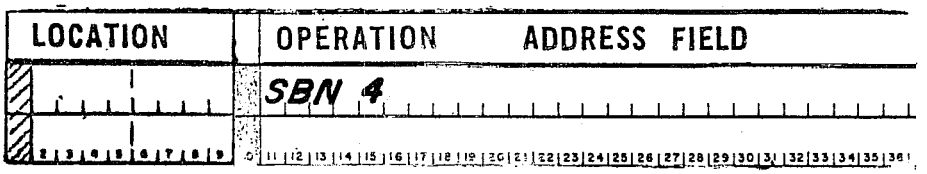
SLØT TIME = 100 NSEC

EXAMPLES

Exercise #1 - Code an instruction to test if a program has gone through a loop the required number of times (4).

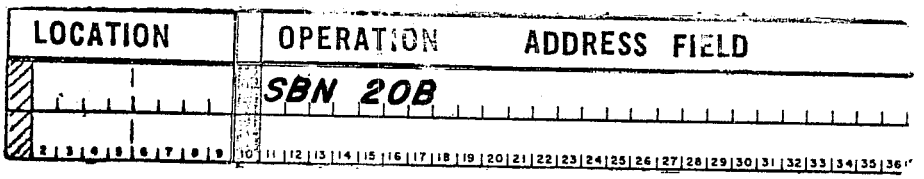
Note: This instruction assumes the A-Register contains the loop count, and the next instruction will finish the test.

ANSWER



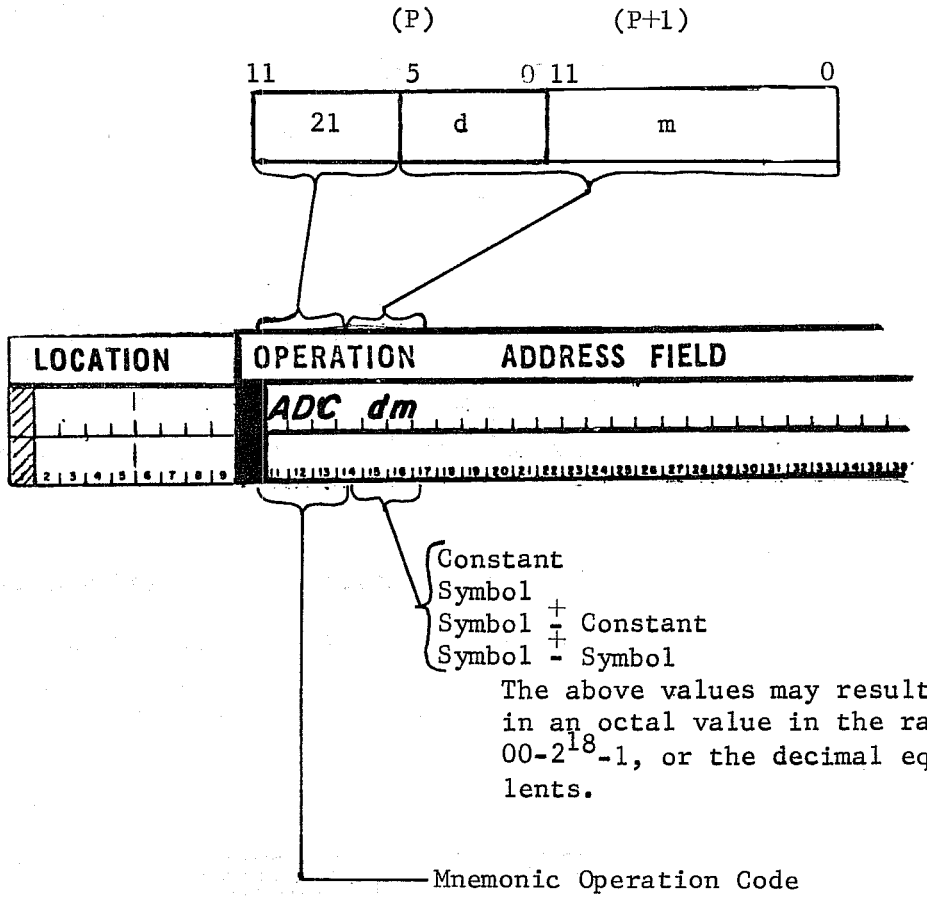
Exercise #2 - Code an instruction to help find a number which is 20B. Assume one number from a list of numbers is already in the A-Register, and another instruction will finish the test.

ANSWER



FORMATS

ADD CØNSTANT



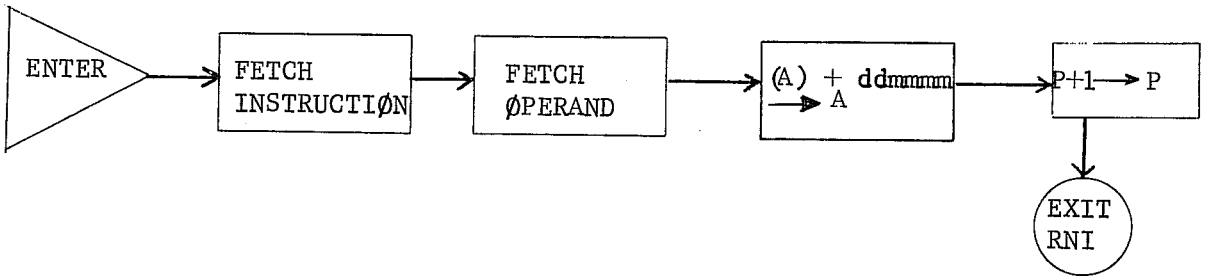
DESCRIPTION

This instruction adds to the A-Register (which is 18-bits) the 18 bit quantity dm (consisting of d as the upper 6-bits and m as the lower 12-bits). RNI @ P+2.

REFERENCES :

ADC

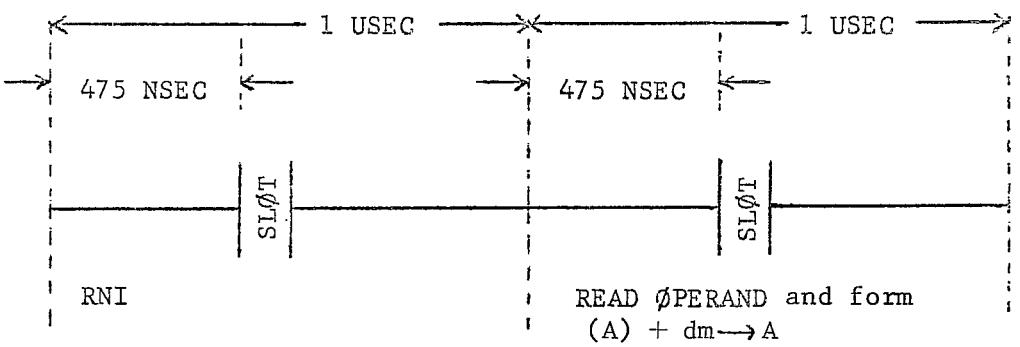
FLOW DIAGRAM



TIMING

6400/6600

2 USEC

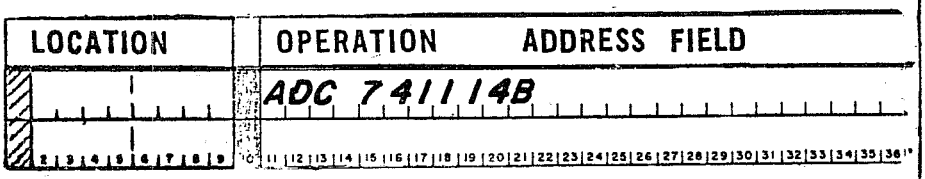


SLØT TIME = 100 NSEC

EXAMPLES

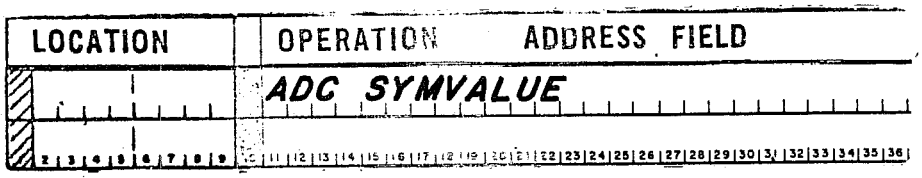
Exercise #1 - Code an instruction to add the bias of 741114 to the contents of A.

ANSWER



Exercise #2 - Code an instruction to add the value "symvalue" to the A-Register.

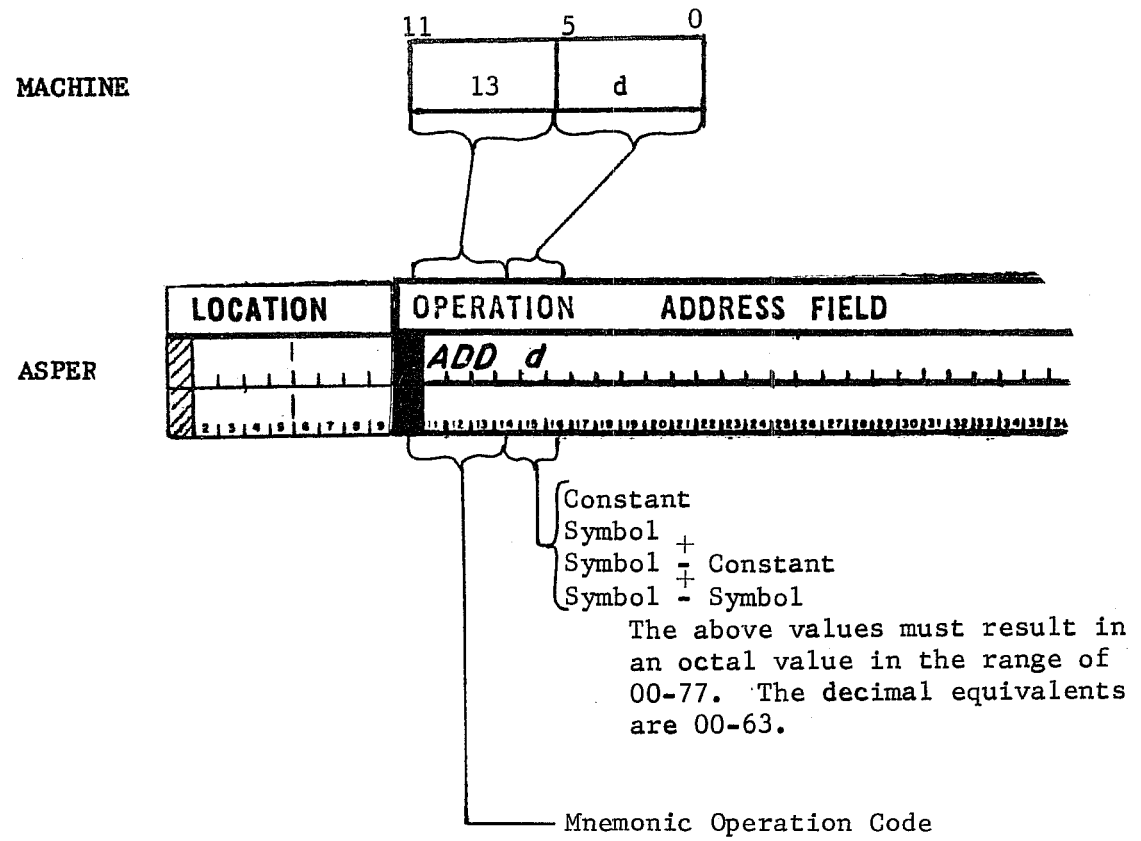
ANSWER



FORMATS

ADD DIRECT

(P)



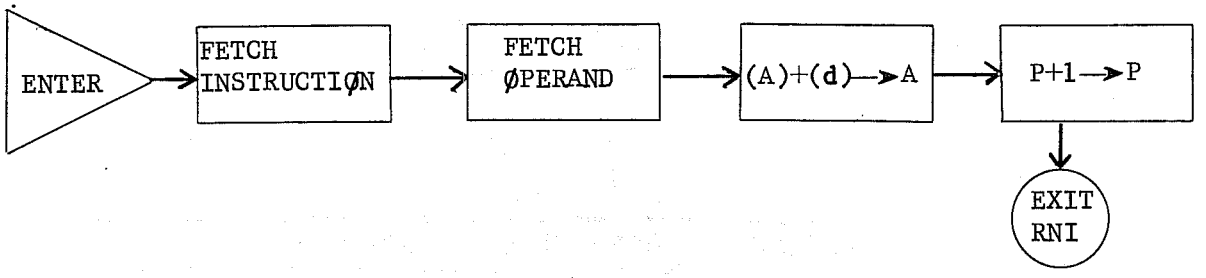
DESCRIPTION

This instruction adds to the A-Register (which is 18-bits) the 12-bit positive quantity contained in memory location d. RNI @ P+1.

REFERENCES :

ADD

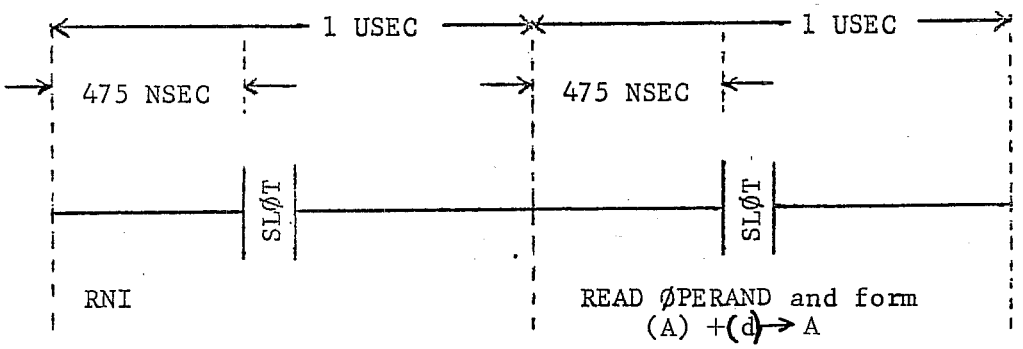
FLOW DIAGRAM



TIMING

6400/6600

2 USEC

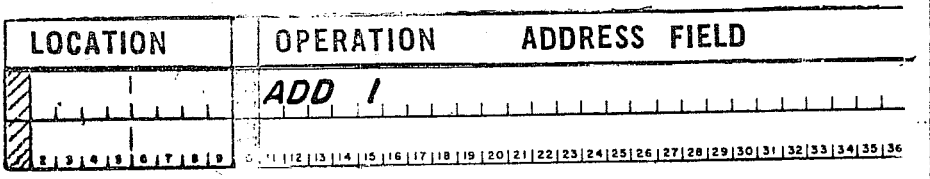


SLØT TIME = 100 NSEC

EXAMPLES

Exercise #1 - Code an instruction to add the contents of low-core location 01, which might be a 12-bit constant to the A-Register.

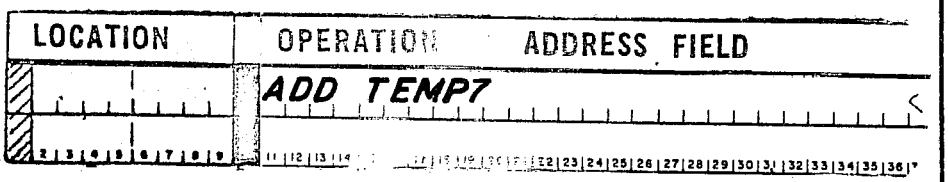
ANSWER



Exercise #2 - Code an instruction to add the contents of location TEMP7, which is a temporary value, to the A-Register.

TEMP7, = 77

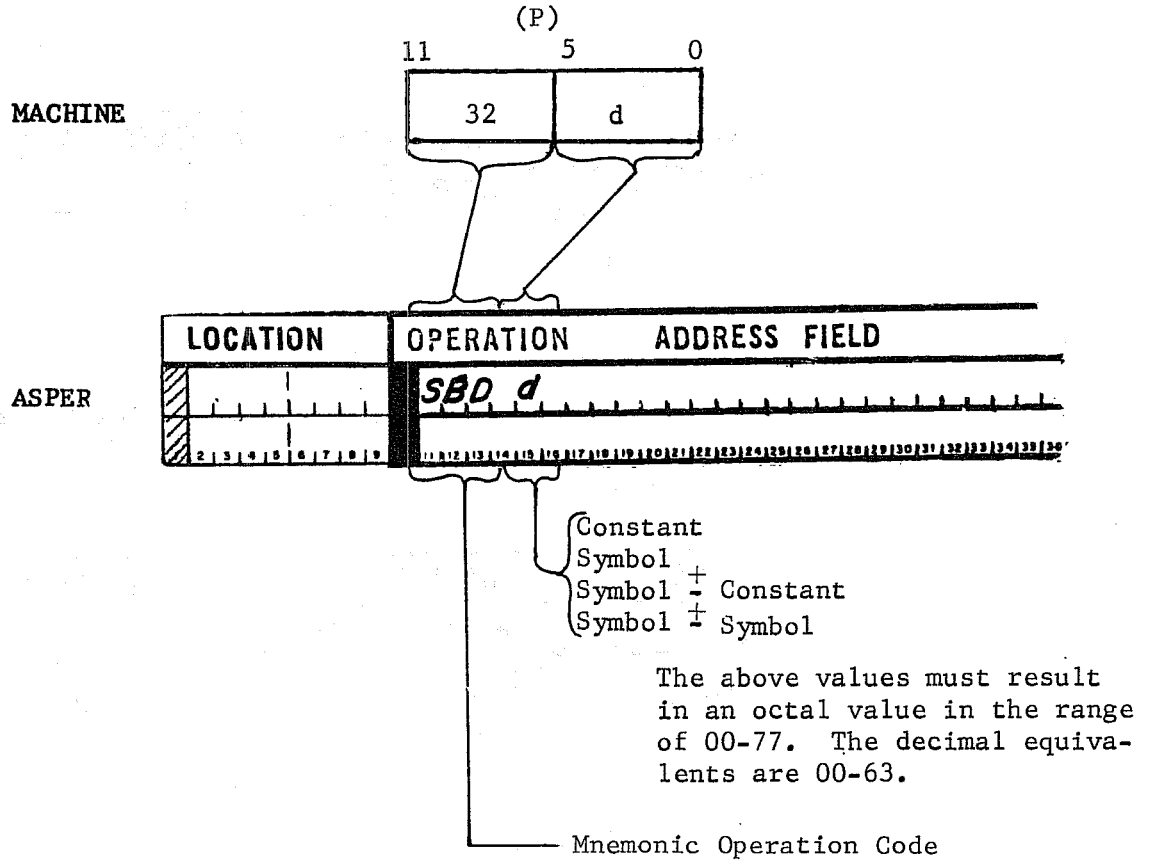
ANSWER



SBD

SUBTRACT DIRECT

FORMATS



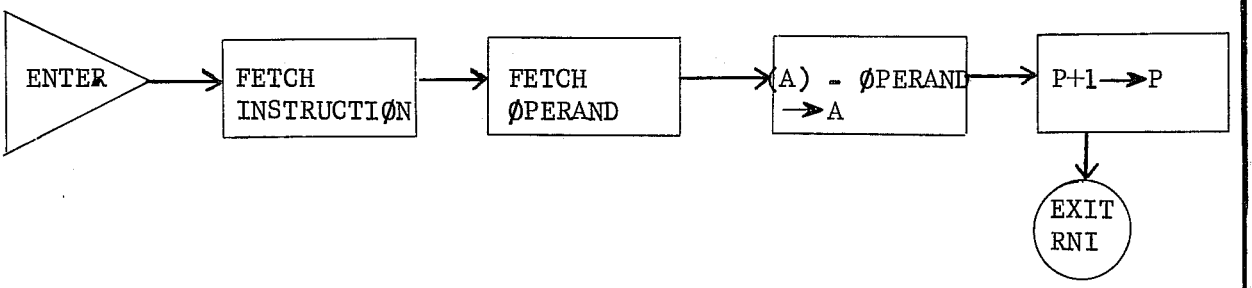
DESCRIPTION

This instruction subtracts from the A-Reigister (which is 18 bits) the 12 bit positive quantity contained in location d. RNI @ P+1.

REFERENCES :

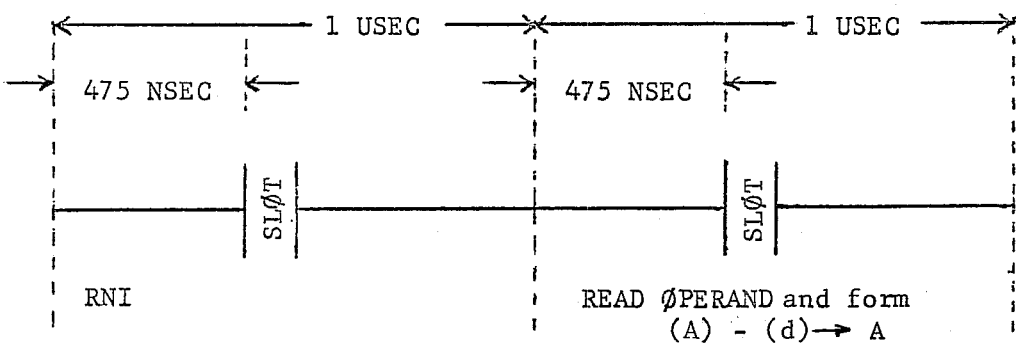
SBD

FLOW DIAGRAM



TIMING

6400/6600 2 USEC

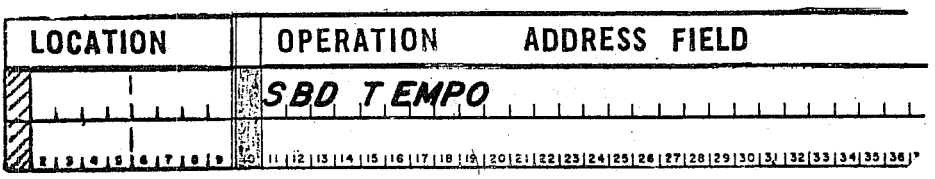


SLØT TIME = 100 NSEC

EXAMPLES

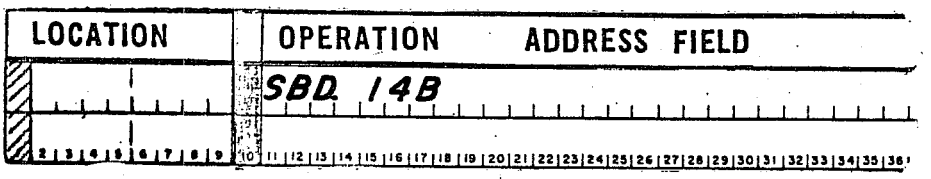
Exercise #1 - Code an instruction to decrement the A-Register by a value of 120_{10}. The value, 120_{10} is in location 70, called TEMPO.

ANSWER



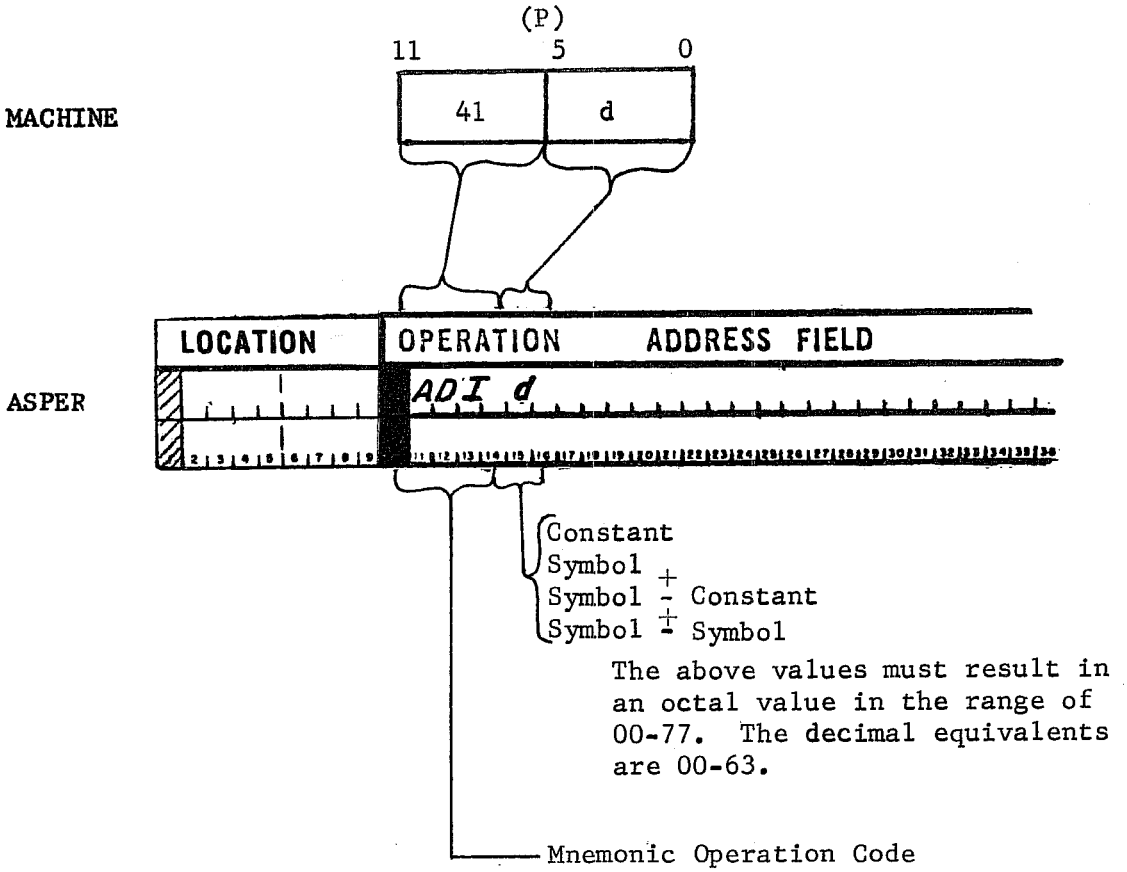
Exercise #2 - Code an instruction to subtract a bias of 3333 from a value in the A-Register. The bias value is in location 12_{10}.

ANSWER



FORMATS

ADD INDIRECT



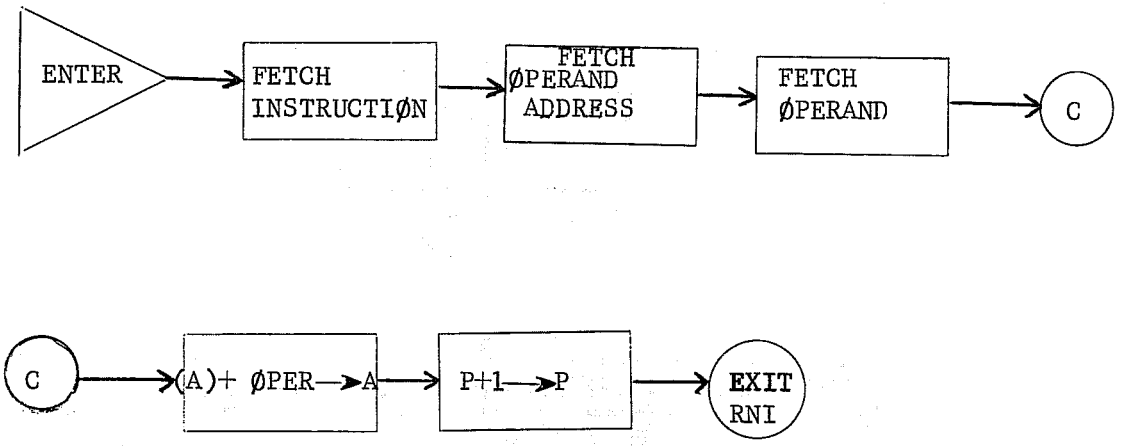
DESCRIPTION

This instruction adds to the contents of the A-Register (which is 18-bits), a 12-bit positive operand obtained by indirect addressing. Location d is read out of memory, and the word obtained is used as the operand address. RNI @ P+1

REFERENCES :

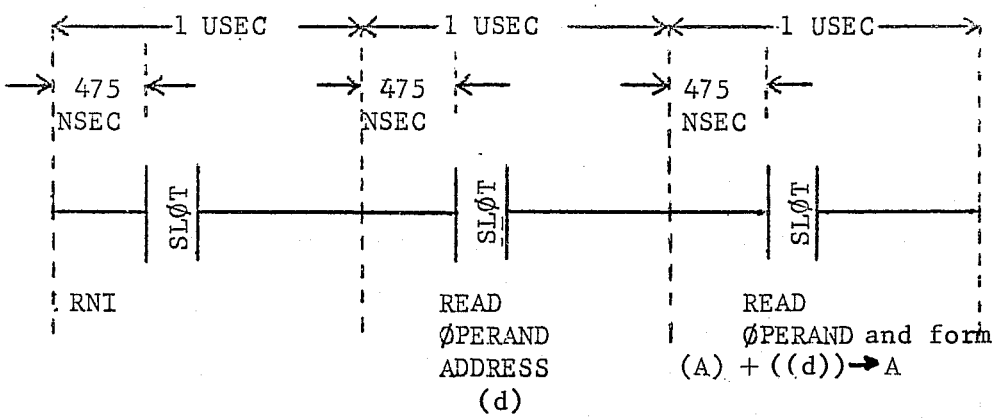
ADI

FLOW DIAGRAM



TIMING

6400/6600 3 USEC



SLØT TIME = 100 NSEC

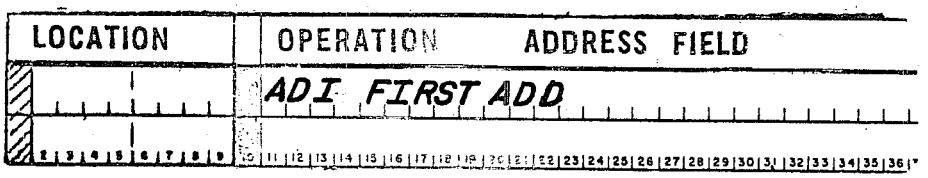
EXAMPLES

Exercise #1 - Various instructions throughout a program need to refer to the same location. Code an instruction that needs the contents of location 6000, and 6000 is also found in location 76.

Assume FIRSTADD = 0076

(0076) = SECONADD = 6000

ANSWER

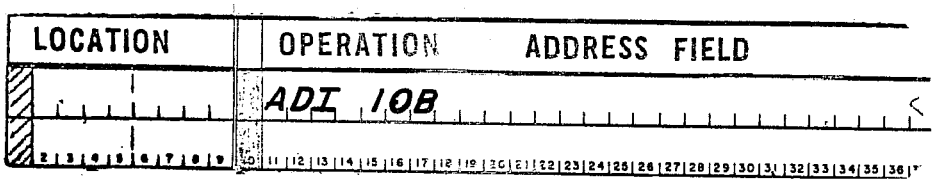


Exercise #2 - Code an instruction which uses location 10_8 as a first reference and location 7000_8 as a second reference to pick up a flag, to add to A.

(0010) = 7000

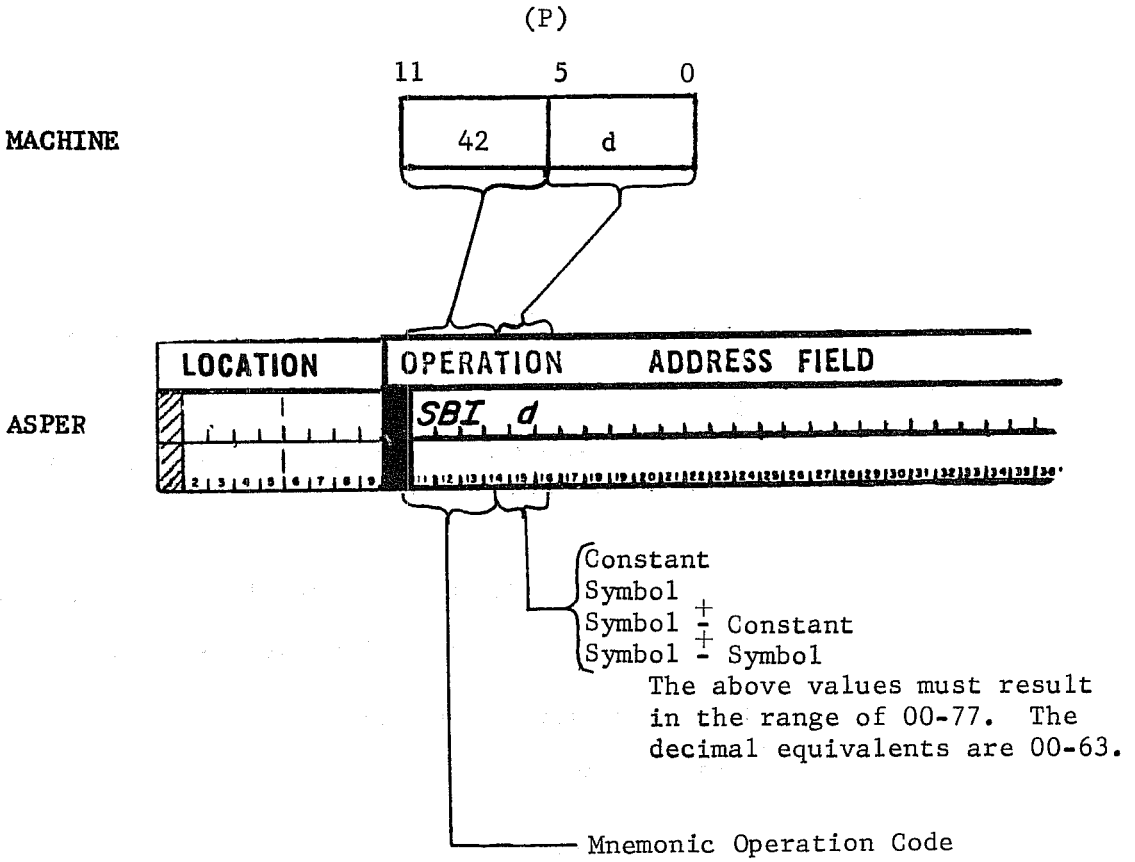
(7000) = Flag

ANSWER



FORMATS

SUBTRACT INDIRECT



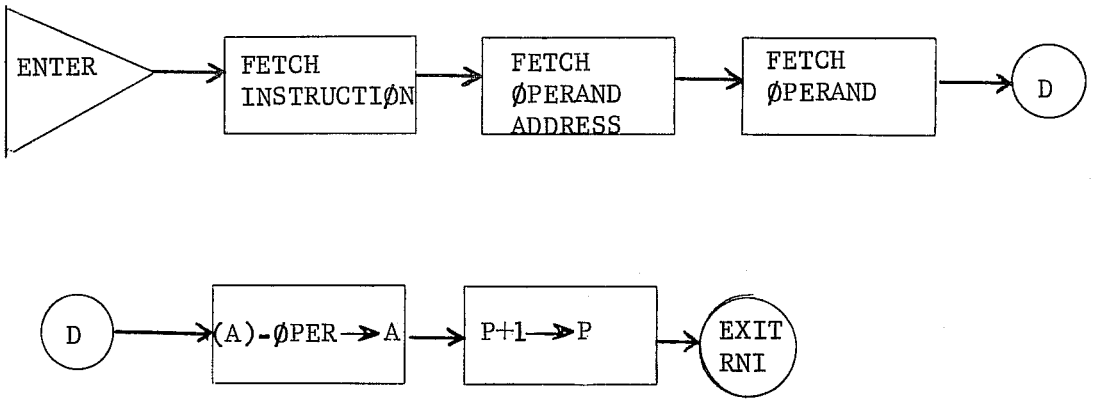
DESCRIPTION

This instruction subtracts from the A-Register (which is 18-bits) a 12-bit positive operand obtained by indirect addressing. Location d is read out of memory, and the word obtained is used as the operand address. RNI @ P+1

REFERENCES :

SBI

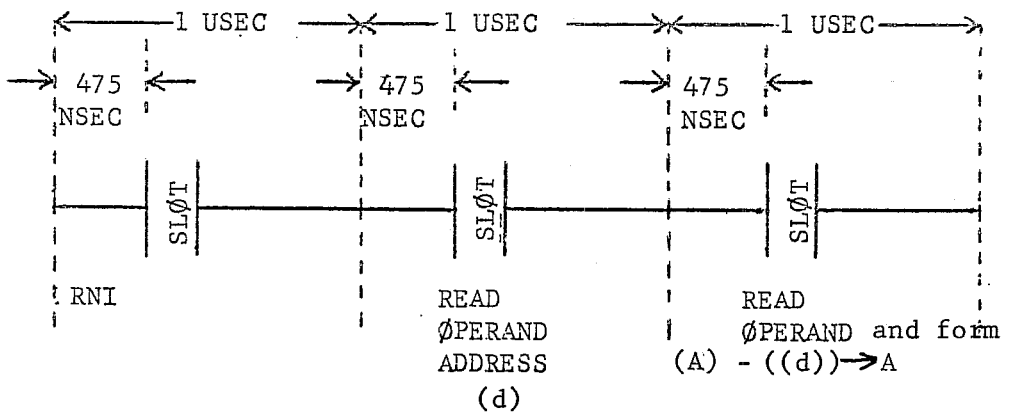
FLOW DIAGRAM



TIMING

6400/6600

3 USEC



SLØT TIME = 100 NSEC

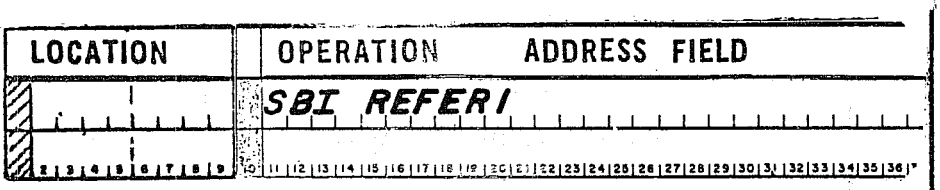
EXAMPLES

Exercise #1 - Code an instruction to subtract from the A-Register the value found in location 7021, called RESULT. The address "RESULT" is also found in location 2 called "REFER1".

Assume (0002)=(REFER1) = 7021 = RESULT

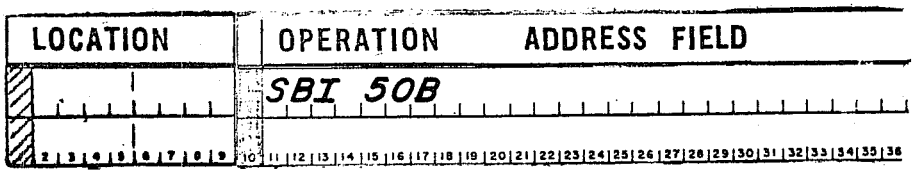
(RESULT) = ØPERAND

ANSWER



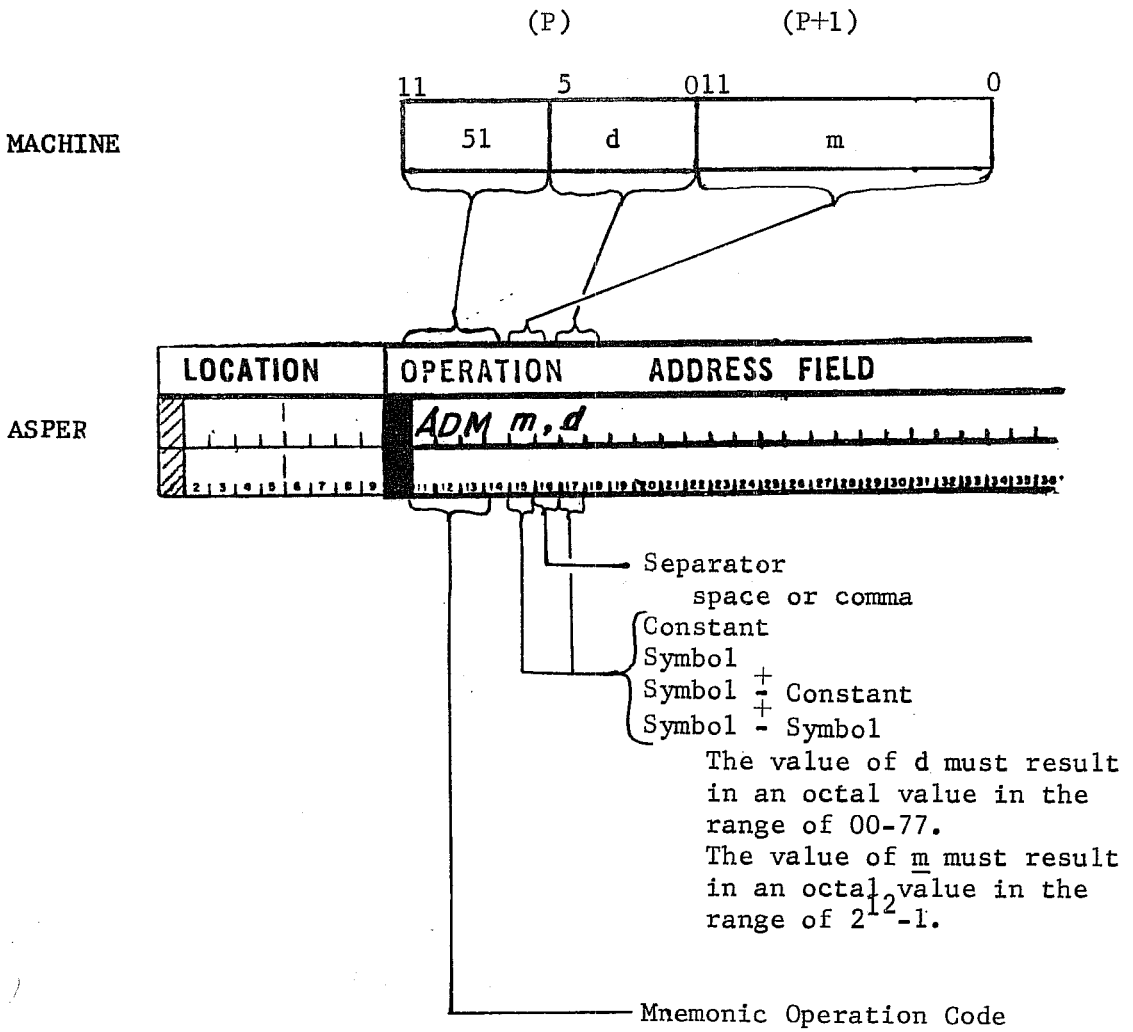
Exercise #2 - Code an instruction to decrement the contents of the A-Register by a constant, whose address is found in location 50_8.

ANSWER



FORMATS

ADD MEMØRY INDEX



DESCRIPTION

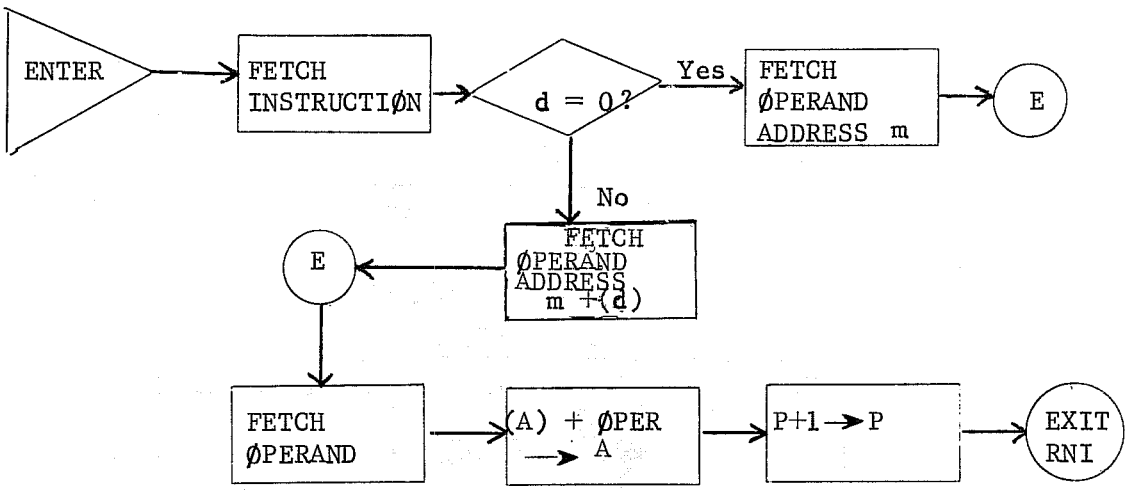
This instruction adds to the contents of the A-Register (which is 18-bits) a 12-bit positive operand obtained by indexed addressing.

Note: If $d=0$, the operand address is simply m.

If $d \neq 0$, then m plus the contents of location d, m+(d) is the operand address. Thus the contents of d may be used as an index quantity to modify operand addresses. RNI @ P+2.

REFERENCES :

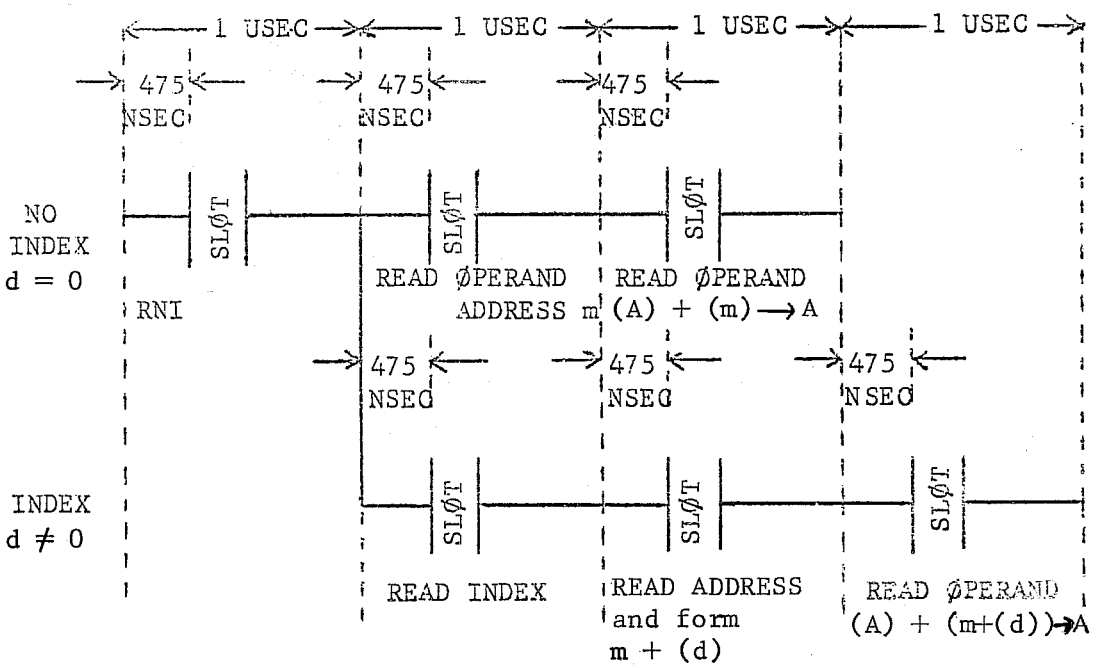
FLOW DIAGRAM



TIMING

6400/6600

3-4 USEC

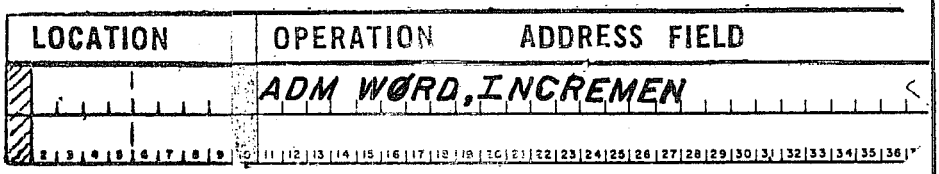


SLØT TIME = 100 NSEC

EXAMPLES

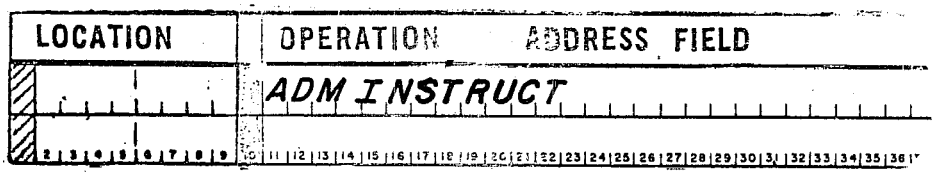
Exercise #1 - Code an instruction that is used to add a constant from a series of constants to the A-Register. Let the base address equal WØRD, which might happen to equal 1244, and the index address equal INCREMEN which may equal 01.

ANSWER



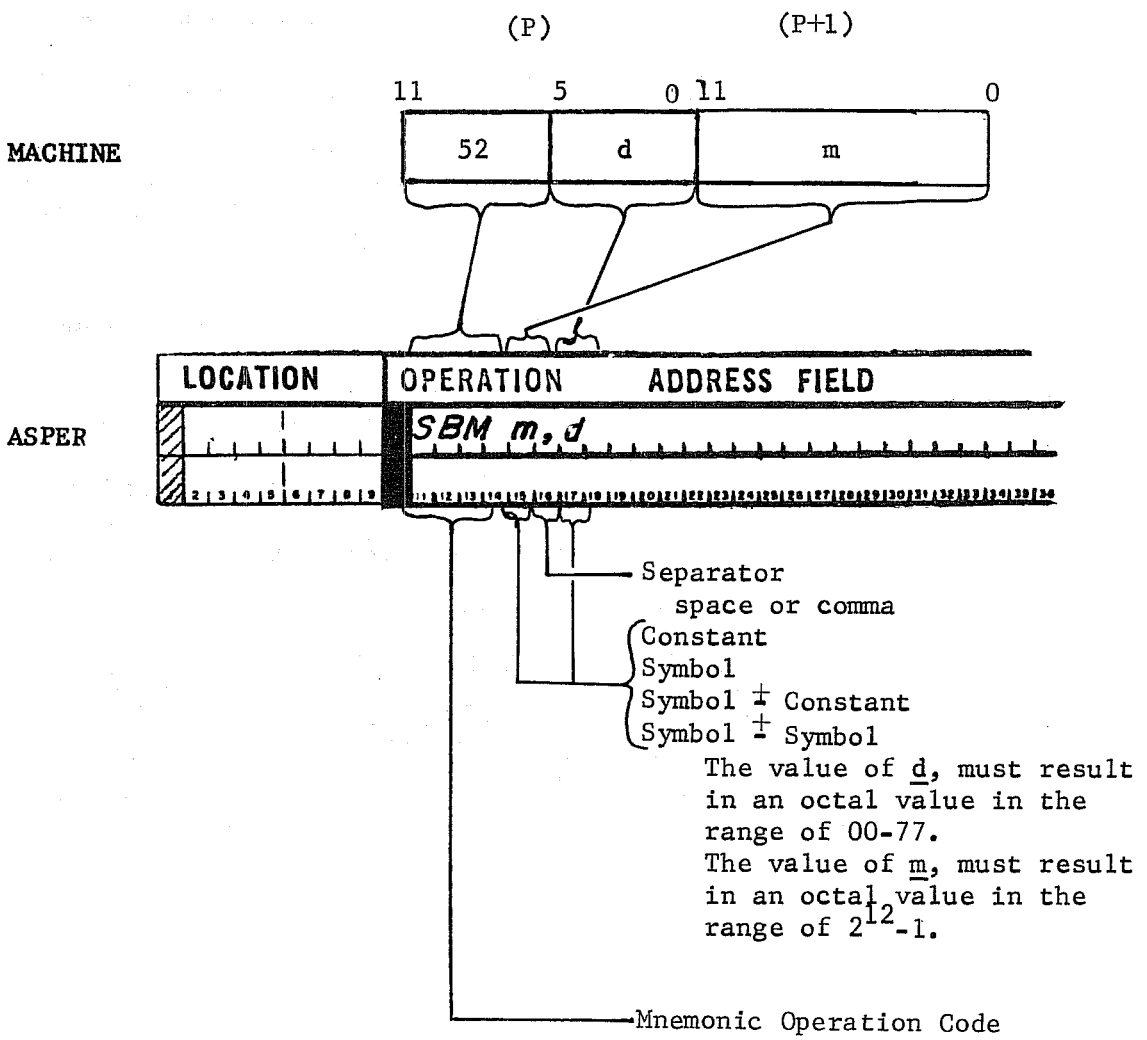
Exercise #2 - Code an instruction to add a constant to the A-Register. The constant has a 12-bit address and needs no indexing. The address of the constant is called INSTRUCT and may happen to equal 5670.

ANSWER



FORMATS

SUBTRACT MEMØRY INDEX



DESCRIPTION

This instruction subtracts from the 18-bit, A-Register a 12-bit positive operand obtained by indexed addressing.

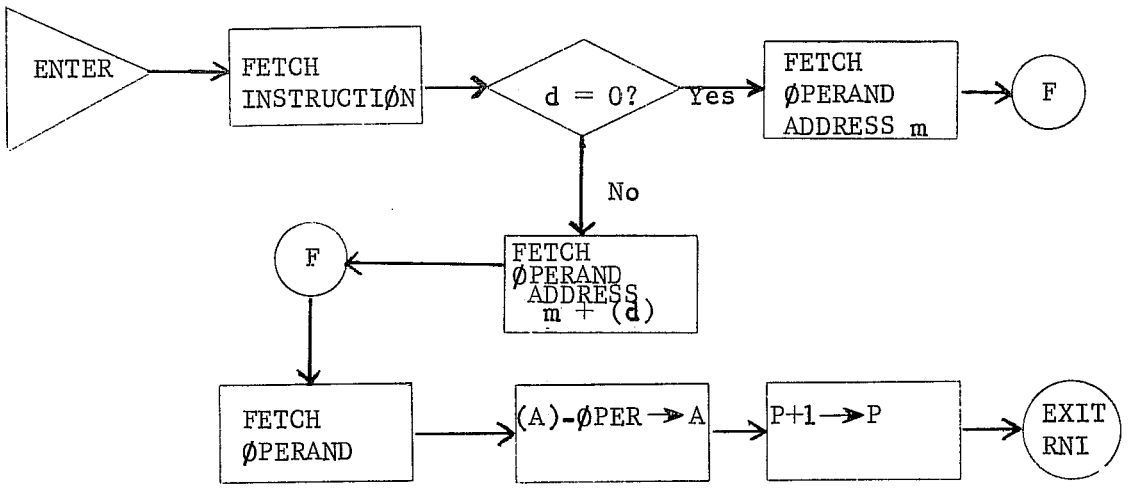
Note: If d=0, the operand address is simply m.

If d ≠ 0, then m plus the contents of location d, m + (d) is the operand address. Thus the contents of d may be used as an index quantity to modify operand addresses.

RNI @ P+2

REFERENCES :

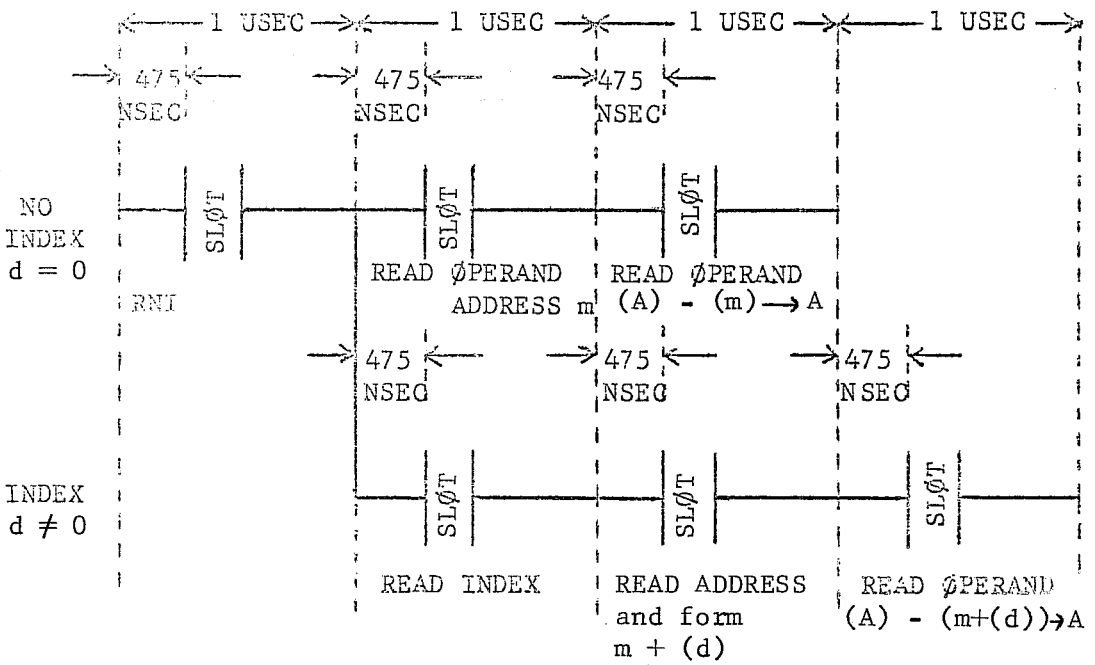
FLOW DIAGRAM



TIMING

6400/6600

3-4 USEC



SLØT TIME = 100 NSEC

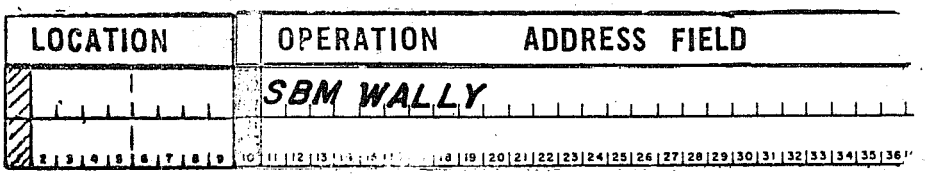
EXAMPLES

Exercise #1 - Code an instruction to subtract the number found in location WALLY (in which WALLY equals 0245_8), from the A-Register. No indexing is needed on this instruction.

WALLY = 0245

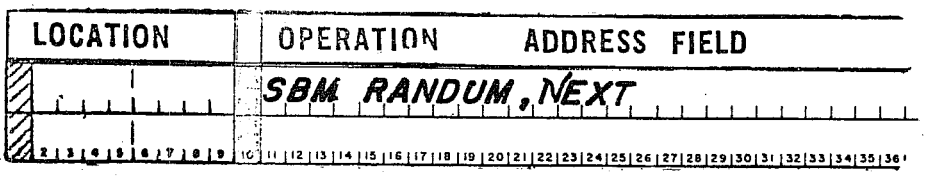
(WALLY) = ØPERAND

ANSWER



Exercise #2 - Code an instruction using memory index to subtract a number found in a table from the A-Register. The index address equals NEXT, which is equal to the arbitrary value 07. The base address of the table is called RANDUM, which might equal 1030.

ANSWER



NEXT = 07

(NEXT) - some index value

RANDUM = 1030

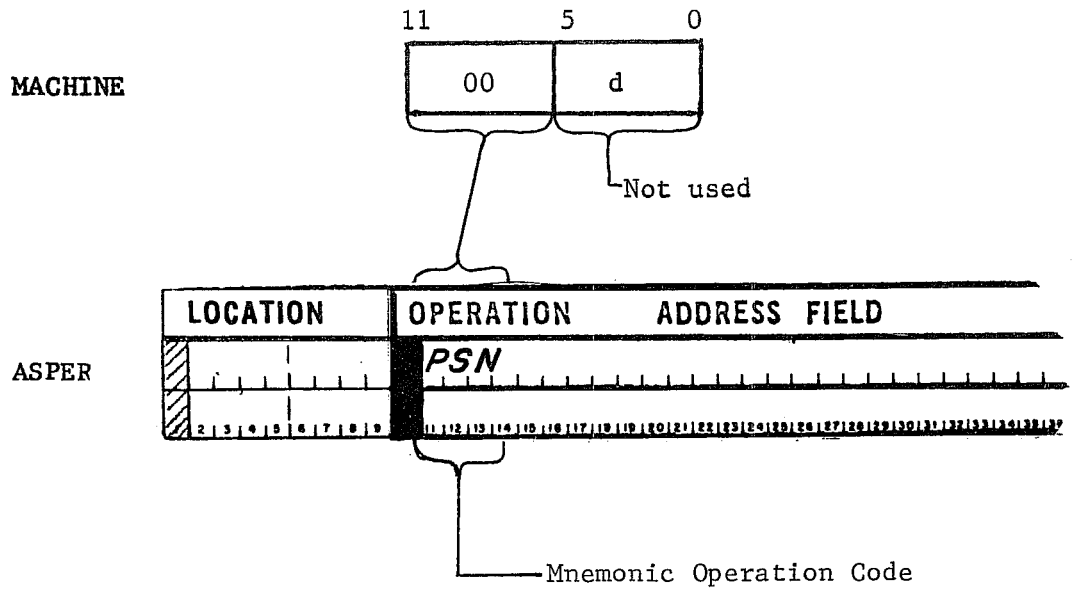
(RANDUM + (NEXT))= ØPERAND

PSN

FORMATS

PASS

(P)



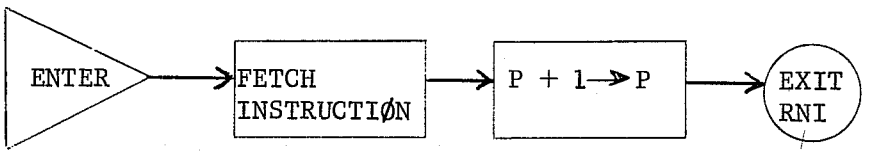
DESCRIPTION

This is a no operation instruction and is a convenience for padding out a program. RNI @ P+1.

REFERENCES :

PSN

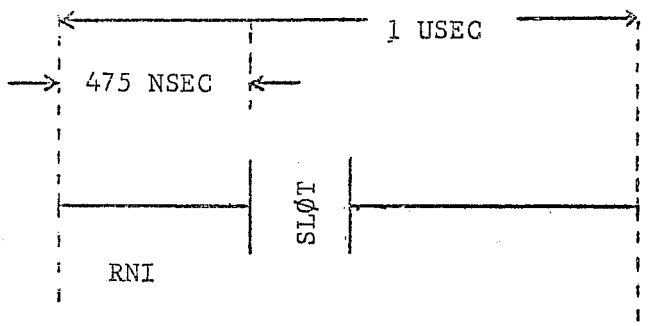
FLOW DIAGRAM



TIMING

6400/6600

1 USEC

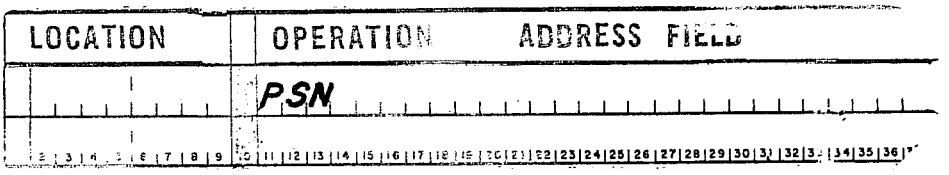


SLØT TIME = 100 NSEC

EXAMPLES

Exercise #1 - Code an instruction that takes up space in a program, so that instructions may be generated, and stored there during execution.

ANSWER

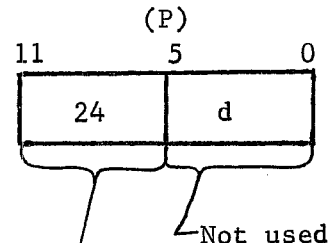


PSN

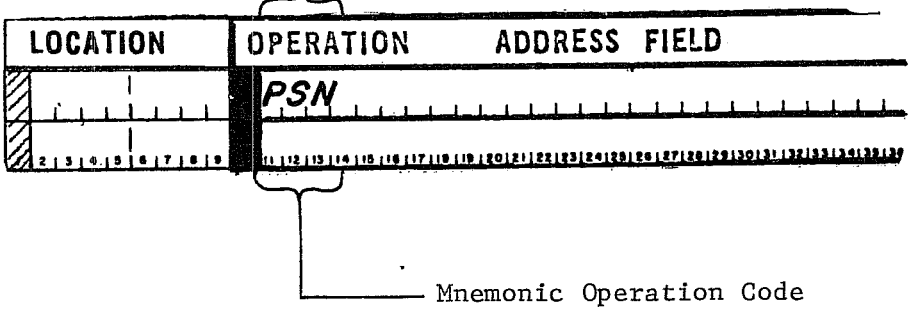
FORMATS

PASS

MACHINE



ASPER



DESCRIPTION

This is a no operation instruction, and is a convenience for padding out a program. RNI + 1 (See pages 20-0 - 20-2)

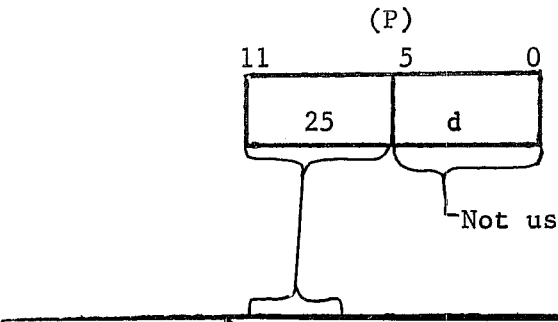
REFERENCES :

PSN

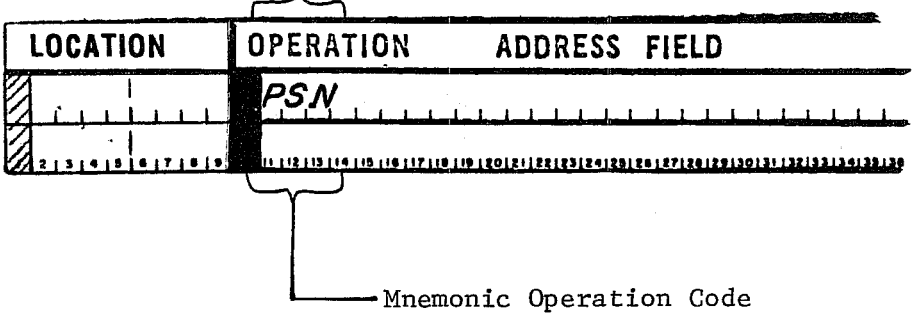
FORMATS

PASS

MACHINE



ASPER



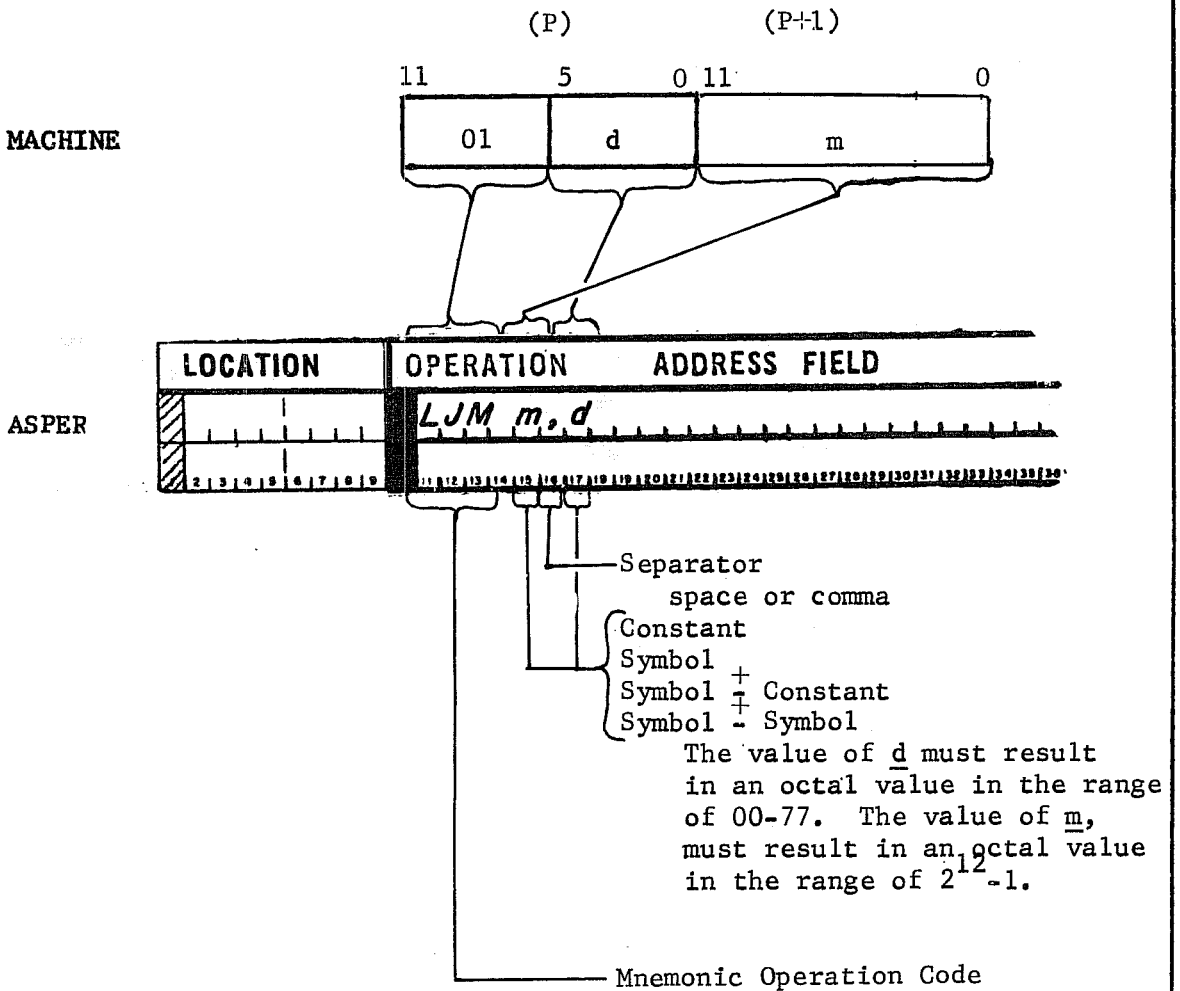
DESCRIPTION

This is a no operation instruction, and is a convenience for padding out a program. RNI + 1 (See pages 20-0 -20-2)

REFERENCES :

FORMATS

LØNG JUMP



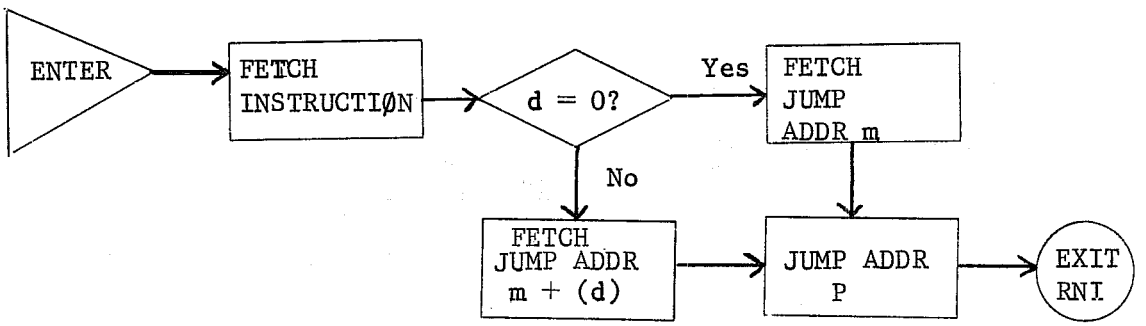
DESCRIPTION

This instruction jumps to the sequence of instructions beginning at the address given by m + (d) .

If d = 0, then m is not modified.

REFERENCES :

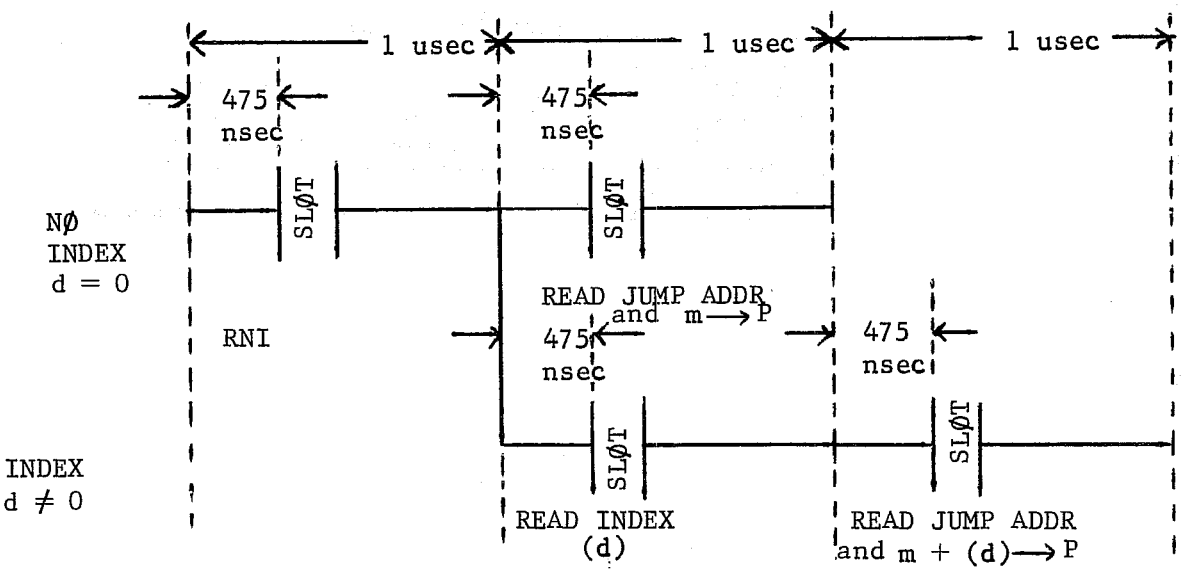
FLOW DIAGRAM



TIMING

6400/6600

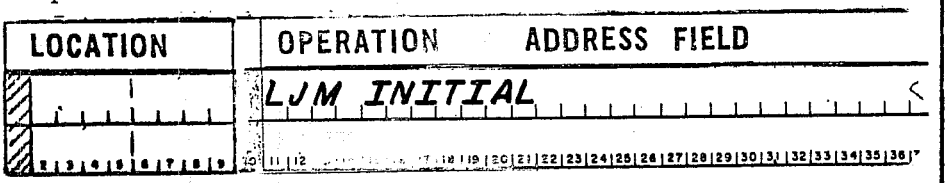
2-3 usec



EXAMPLES

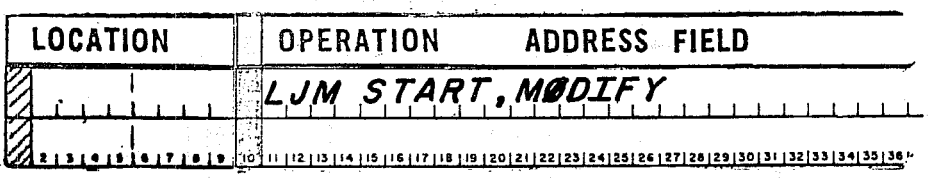
Exercise #1 - Code an instruction that unconditionally transfers control to a new sequence of instructions, whose beginning address is called INITIAL. Use no indexing. INITIAL might happen to equal 4320.

ANSWER



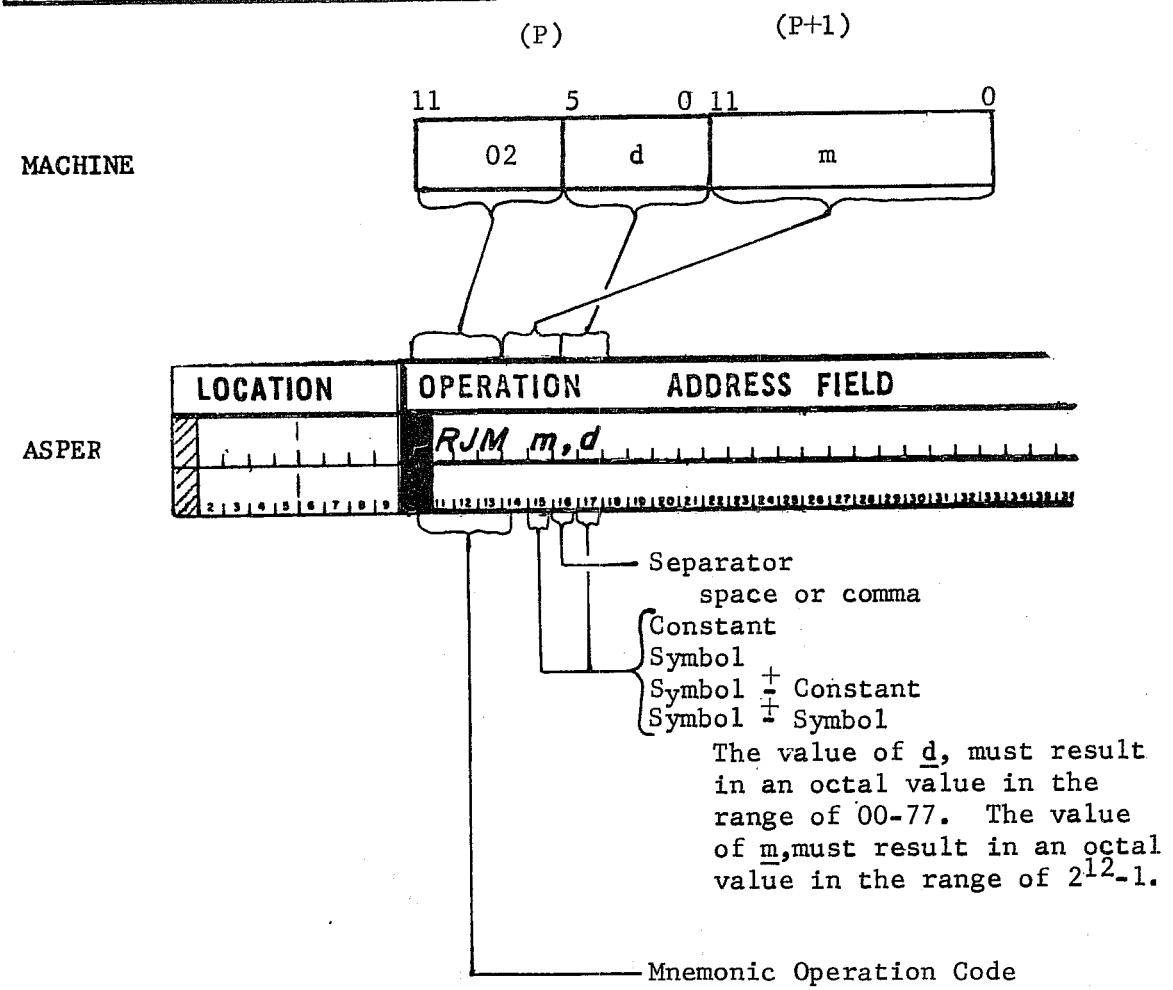
Exercise #2 - Code an instruction that unconditionally transfers control to a new sequence of instructions depending upon the contents of a switch. Let the address of the first program be called START and assume the index is generated during execution and adds to START to give the sequence of instructions. The index location is called MØDIFY.

ANSWER



FORMATS

RETURN JUMP



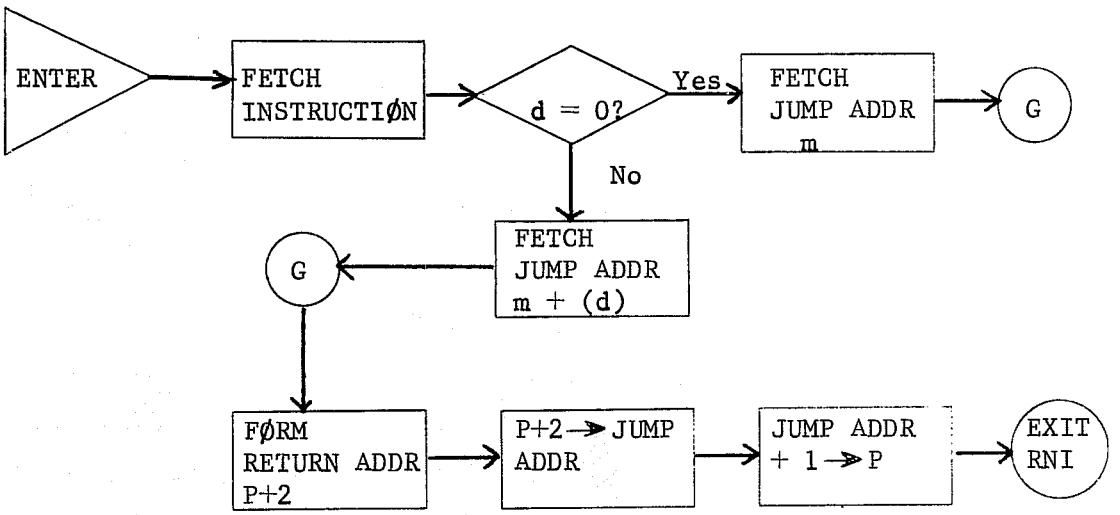
DESCRIPTION

This instruction jumps to the sequence, of instructions, beginning at the location given by m + (d) . If d = 0, then m is not modified. The current program address plus two (P +2) is stored at the jump address, and is used as the return address to the main routine when the new sequence is finished. The new sequence starts at the jump address plus one.

REFERENCES :

RJM

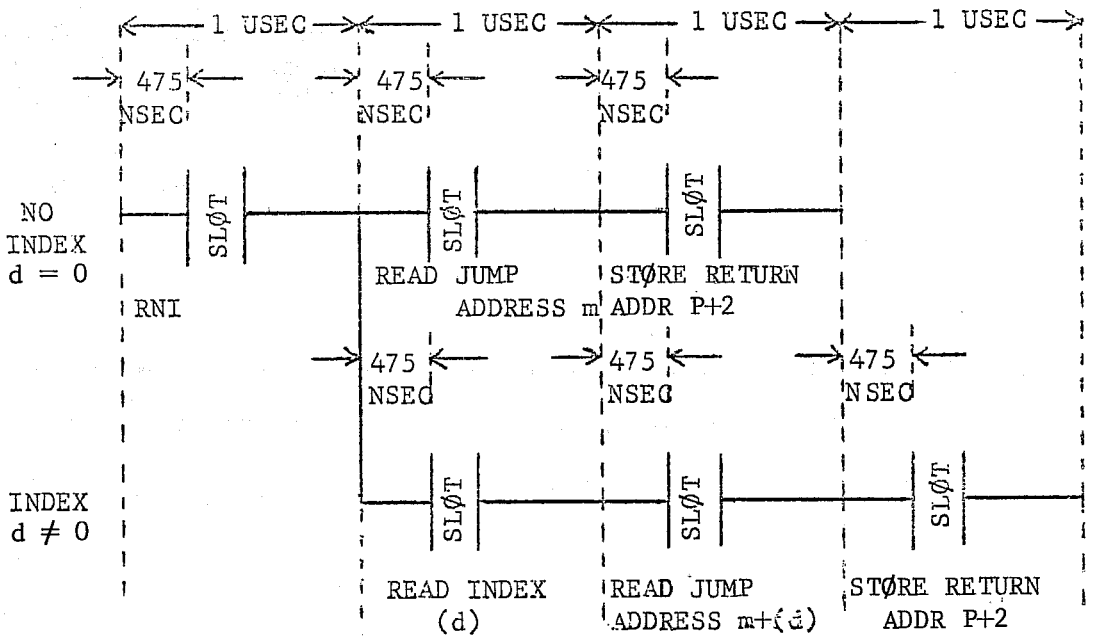
FLOW DIAGRAM



TIMING

6400/6600

3-4 USEC

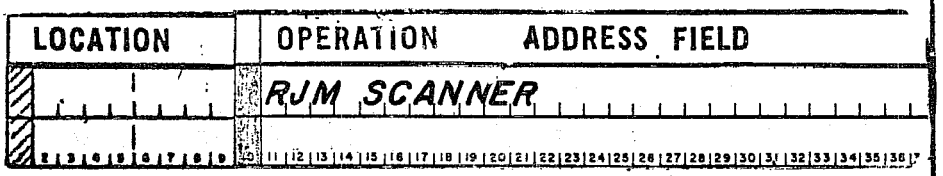


SLØT TIME = 100 NSEC

EXAMPLES

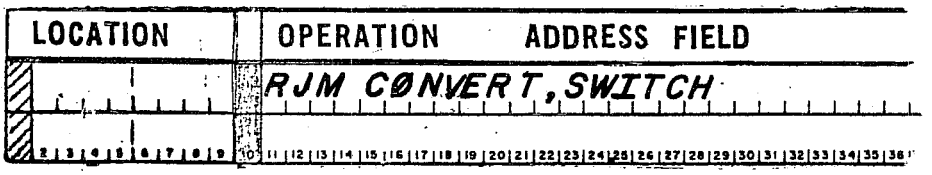
Exercise #1 - Different places in a program it is desired to execute a series of instructions and return to the main program each time. Code an instruction that would allow us to execute a subroutine called SCANNER and return upon completion of the subroutine. No indexing is needed on this instruction.

ANSWER



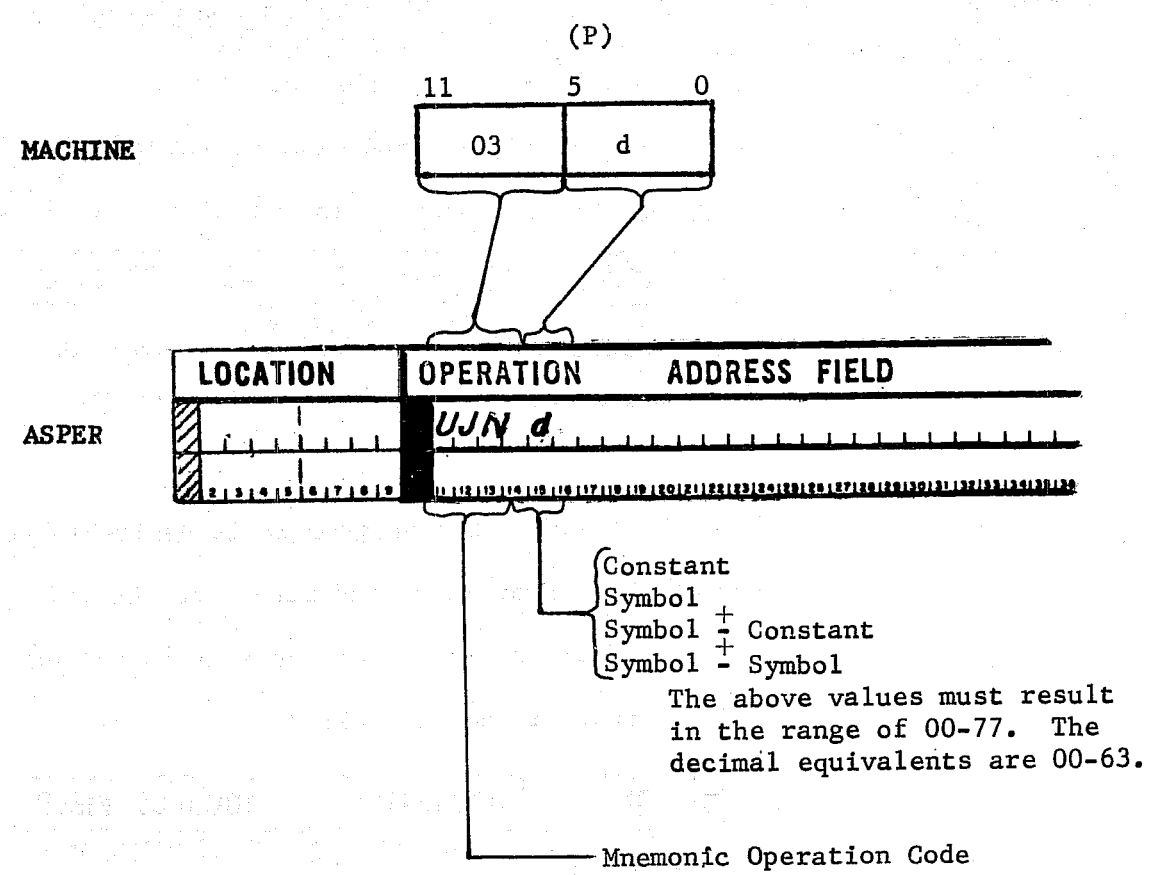
Exercise #2 - In various parts of a program it is desirable to execute one of two subroutines, and return to the main program. The base address of the two routines is called CØNVERT and the index is called SWITCH.

ANSWER



FORMATS

UNCØNDITIØNAL JUMP



DESCRIPTION

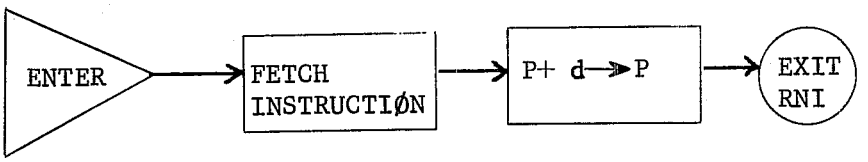
This instruction provides an unconditional jump of up to 31_{10} steps forward or backward from the current program address. The value of d is added to the current program address. If d is positive (01_8 - 37_8) the jump is forward. If d is negative (40_8 - 76_8) the jump is backward.

NOTE: If d = 00 or 77, the processor will hang-up (stop) indefinitely.

REFERENCES :

UJN

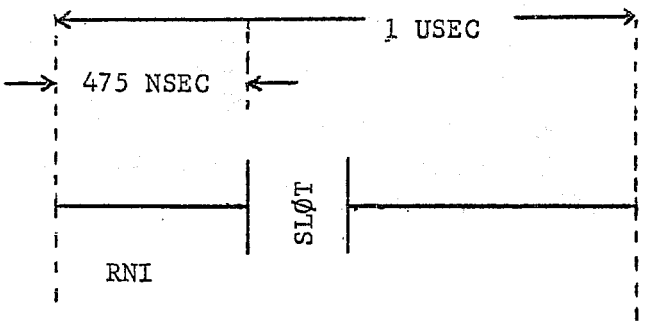
FLOW DIAGRAM



TIMING

6400/6600

1 USEC



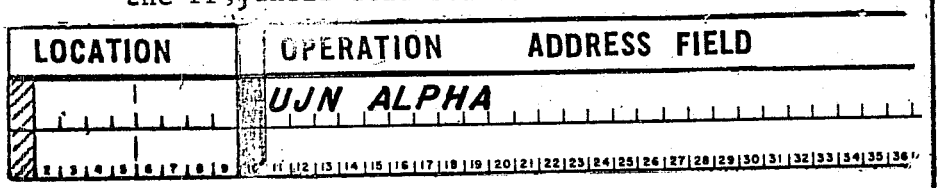
SLØT TIME = 100 NSEC

EXAMPLES

Exercise #1 - Code an instruction that unconditionally transfers control forward to a new set of instructions. The new address is called ALPHA and must be in the range of 01-37 places forward.

Note: d must not equal 00 or 77, for this will hang-up the PP, until dead start.

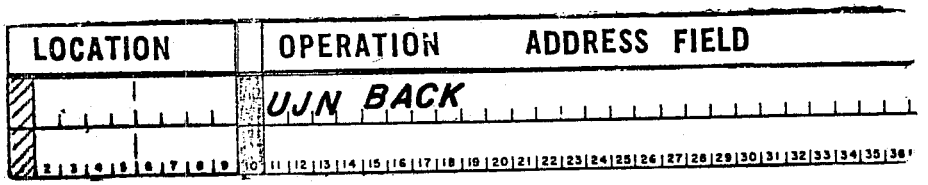
ANSWER



Exercise #2 - Code an instruction that unconditionally transfers control backward to a new set of instructions. The new address is called BACK and must be in the range of 01-37 places back.

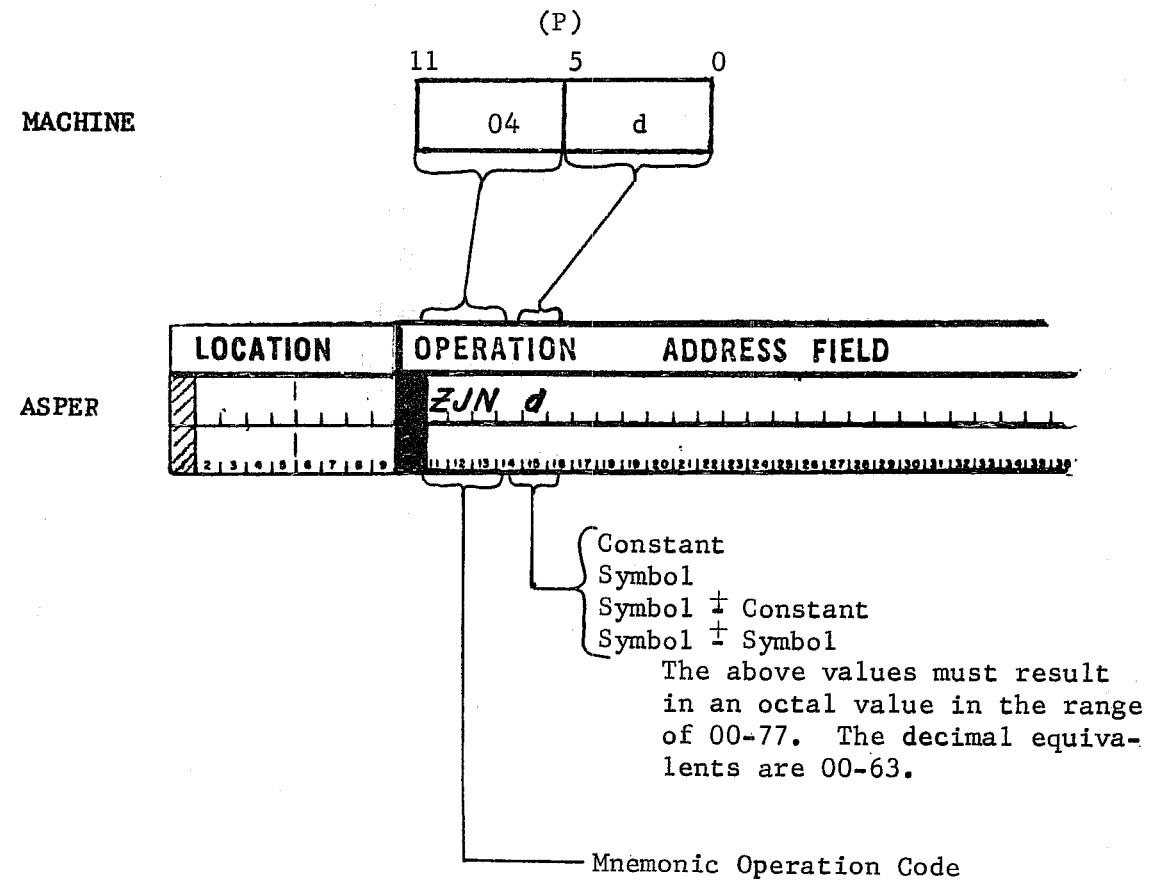
Note: See above note.

ANSWER



FORMATS

ZERØ JUMP



DESCRIPTION

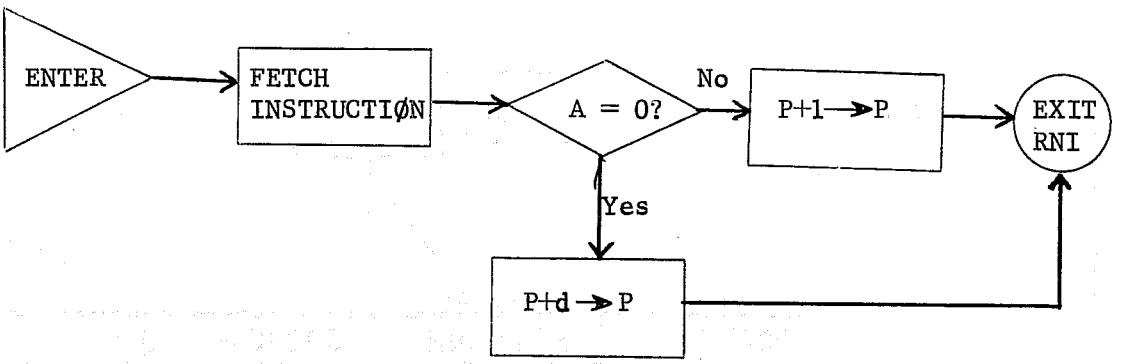
This instruction provides a conditional jump of up to 31 steps forward or backward from the current program address if the content of the A-Register is zero. If A is a nonzero, the next instruction @ P+1 is executed. Negative zero (777777) is treated as nonzero. If d is positive (01_8-37_8) the jump is forward. If d is negative (40_8-76_8) the jump is backward.

Note: If d = 00 or 77 and the test is met, the processor will hang-up (stop) indefinitely.

REFERENCES :

FLOW DIAGRAM

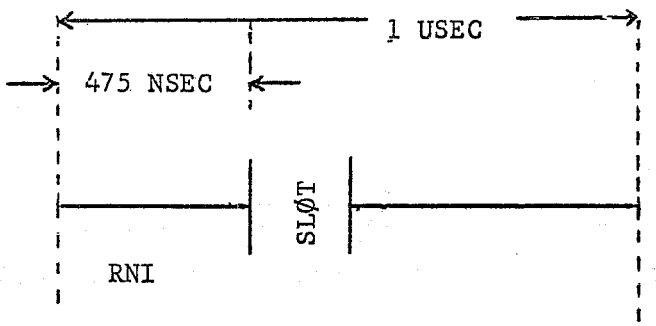
ZJN



TIMING

6400/6600

1 USEC



SLØT TIME = 100 NSEC

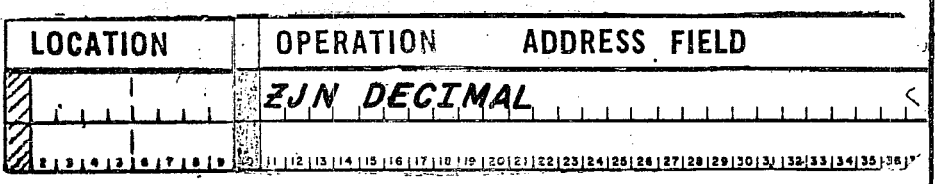
EXAMPLES

Exercise #1 - The computer has decision making ability, depending upon the contents of the A-Register. Code an instruction that tests A for being zero. If it is a Jump forward an address call DECIMAL, if the condition is not met read the next instruction at P+1.

DECIMAL must be in the range of 01-37, places forward.

Note: See note on page 25-2.

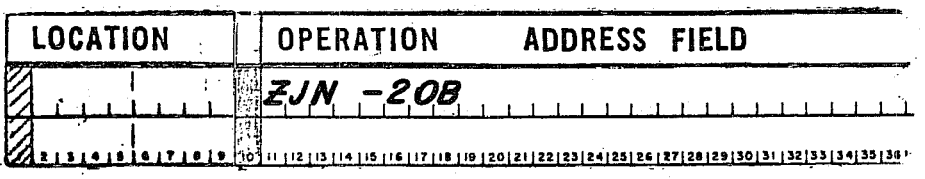
ANSWER



Exercise #2 - Code an instruction that tests the A-Register for zero, and if a zero is found jump back 20_8 locations, otherwise continue at P+1.

Note: See note on page 25-2.

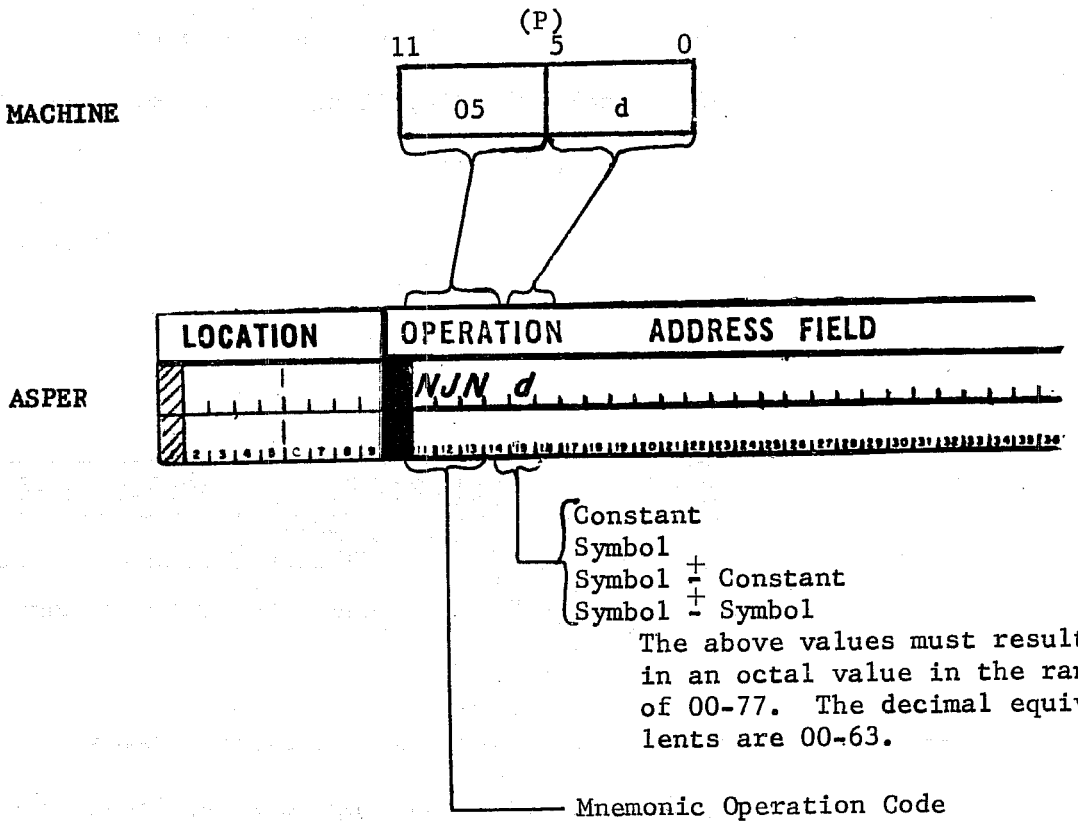
ANSWER



NJN

FORMATS

NØNZERØ JUMP



DESCRIPTION

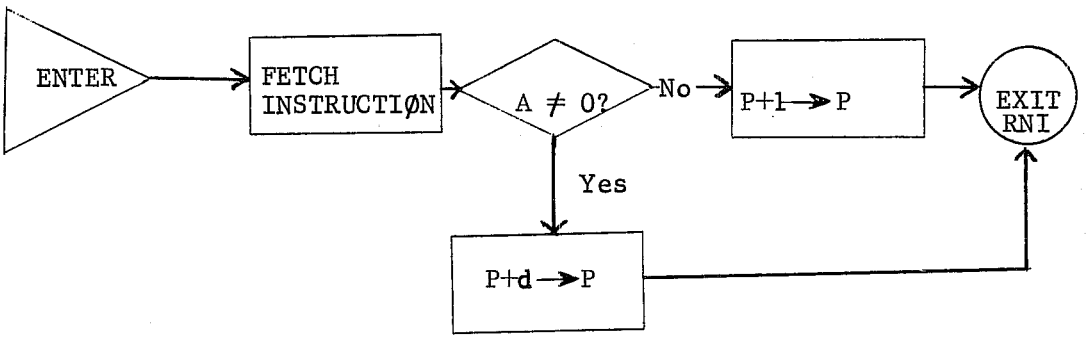
This instruction provides a conditional jump of up to 31 steps forward or backward from the current program address if the content of the A-Register is nonzero. If A is a zero, the next instruction @ P+1 is executed. Negative zero (777777) is treated as nonzero. If d is positive ($01\text{-}37_8$) the jump is forward. If d is negative ($40\text{-}76_8$) the jump is backward.

Note: If d = 00 or 77 and the test is met, the processor will hang-up (stop) indefinitely.

REFERENCES :

NJN

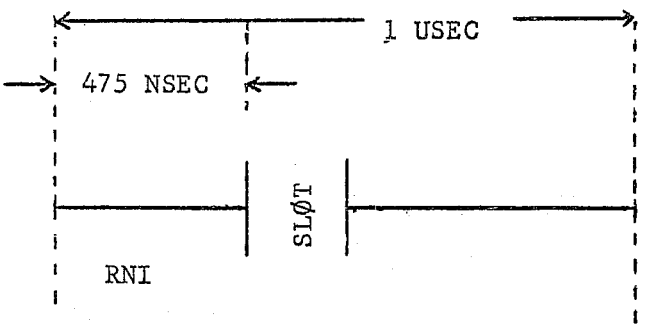
FLOW DIAGRAM



TIMING

6400/6600

1 USEC



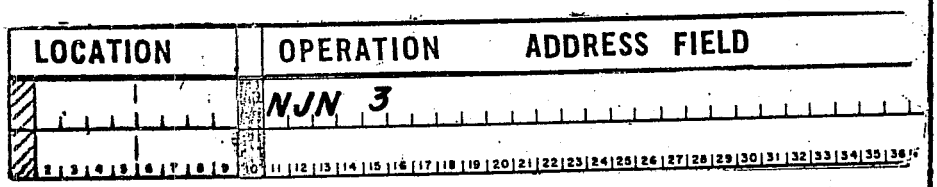
SLØT TIME = 100 NSEC

EXAMPLES

Exercise #1 - Code an instruction that tests the A-Register for a non-zero quantity and jumps forward 3 places if the condition is met.

Note: See note on page 25-2.

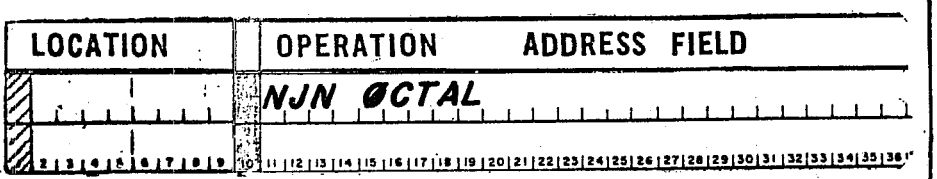
ANSWER



Exercise #2 - Code an instruction that loops back to a location called ØCTAL, if the A-Register is a non-zero quantity. Otherwise RNI @P+1. ØCTAL would be in the range of 01-37 places back.

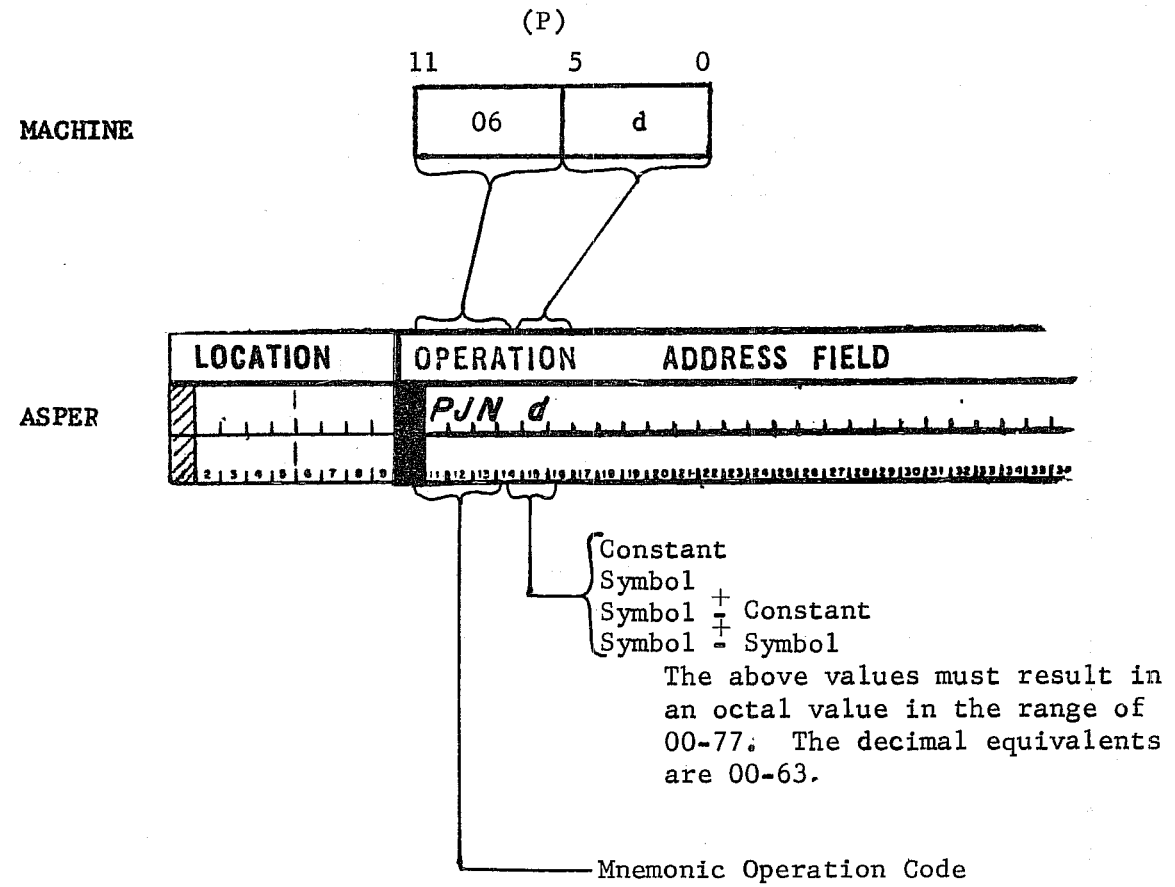
Note: See note on page 25-2.

ANSWER



FORMATS

PLUS JUMP



DESCRIPTION

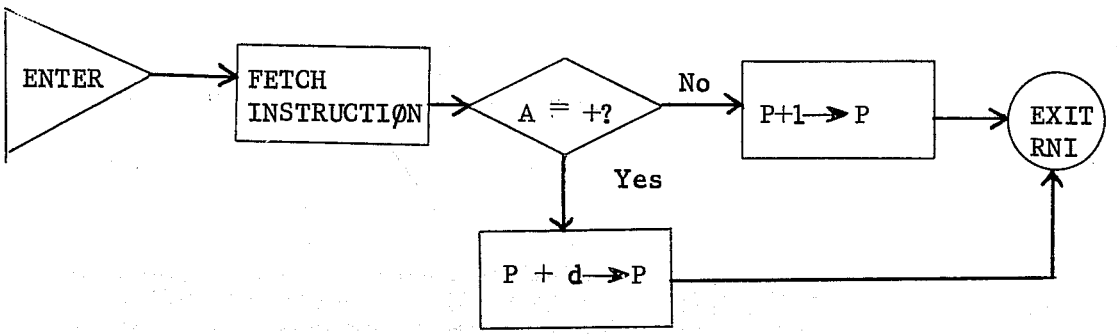
This instruction provides a conditional jump of up to 31 steps forward or backward from the current program address if the contents of the A-Register is positive. If A is negative, the next instruction @ P+1 is executed. A is positive if the 2^{17} is a zero. If d is positive ($01\text{-}37_8$) the jump is forward. If d is negative ($40\text{-}76_8$) the jump is backward.

Note: If d = 00 or 77 and the test is met, the processor will hang-up (stop) indefinitely.

REFERENCES :

PJN

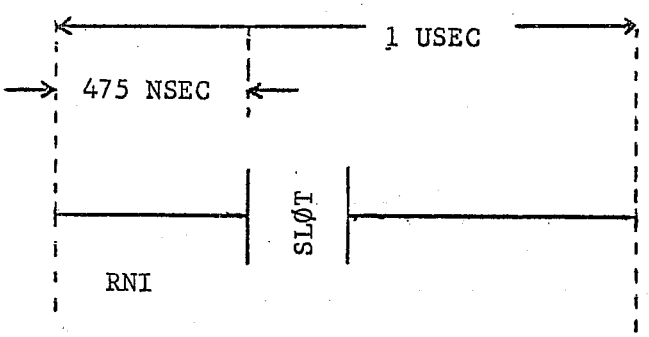
FLOW DIAGRAM



TIMING

6400/6600

1 USEC



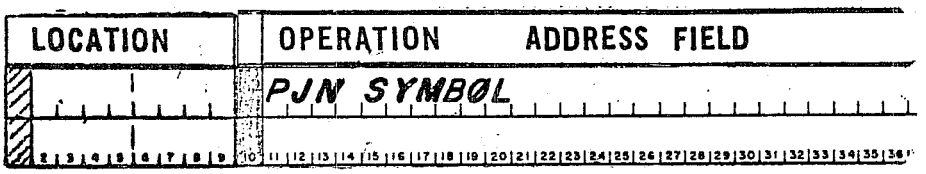
SLØT TIME = 100 NSEC

EXAMPLES

Exercise #1 - Code an instruction that tests the A-Register for a positive number, and if the condition is met, jump forward to location SYMBØL. SYMBØL must be in the range of 01-37 places forward.

Note: See note page 25-2.

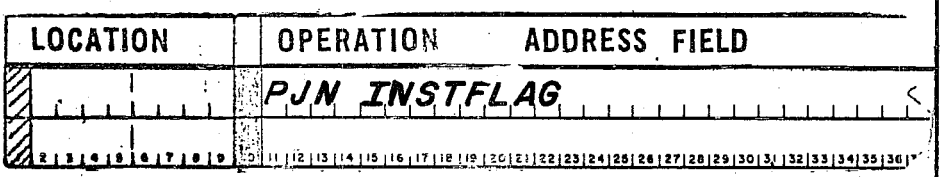
ANSWER



Exercise #2 - Code an instruction that jumps backward if the contents of the A-Register is positive. Go back to location INSTFLAG. INSTFLAG must be back 01-37 places.

Note: See note page 28-2.

ANSWER



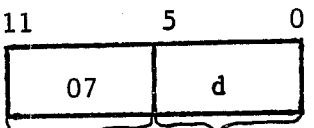
MJN

FORMATS

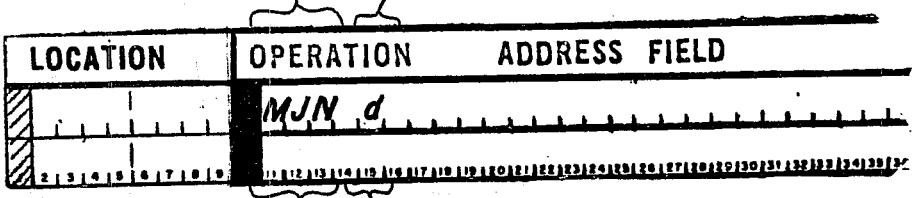
MINUS JUMP

(P)

MACHINE



ASPER



Constant
Symbol
Symbol ± Constant
Symbol ± Symbol

The above values must result in an octal value in the range of 00-77. The decimal equivalents are 00-63.

Mnemonic Operation Code

DESCRIPTION

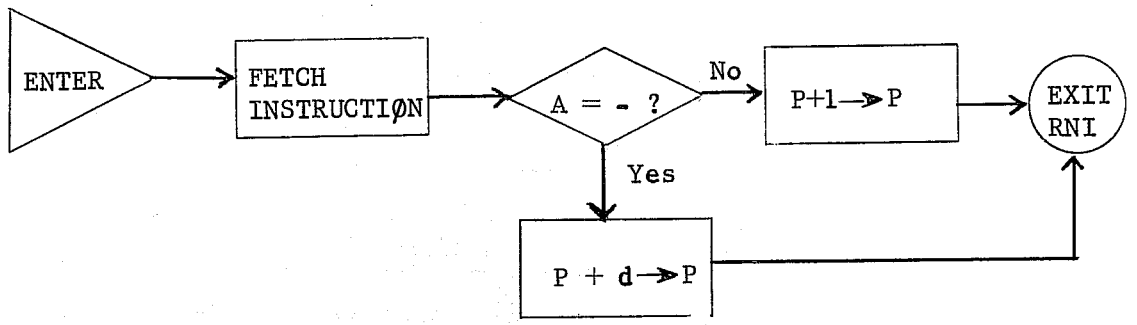
This instruction provides a conditional jump of up to 31 steps forward or backward from the current program address if the content of the A-Register is negative. If A is positive, the next instruction @ P+1 is executed. A is negative (if the 2^{17} bit is a one). If d is positive (01-37$_8$) the jump is forward. If d is negative (40-76$_8$) the jump is backward.

Note: If d = 00 or 77 and the test is met, the processor will hang-up (stop) indefinitely.

REFERENCES :

MJN

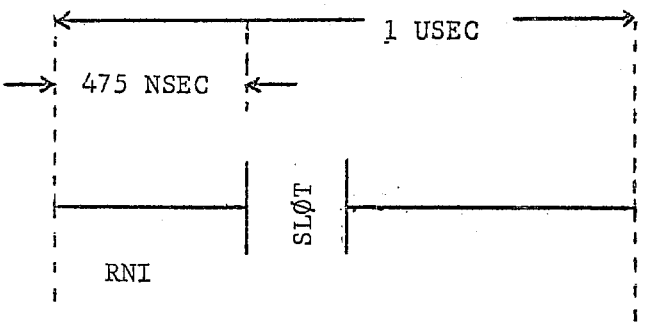
FLOW DIAGRAM



TIMING

6400/6600

1 USEC



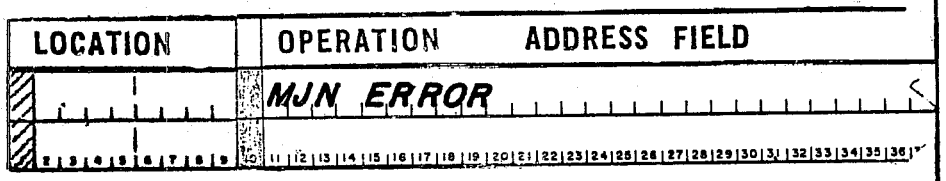
SLØT TIME = NSEC

EXAMPLES

Exercise #1 - Code an instruction that tests the A-Register and if it finds it negative, jumps forward to a set of instructions beginning at location ERRØR. ERRØR must be in the range of 01-37 places forward.

Note: See note on page 25-2.

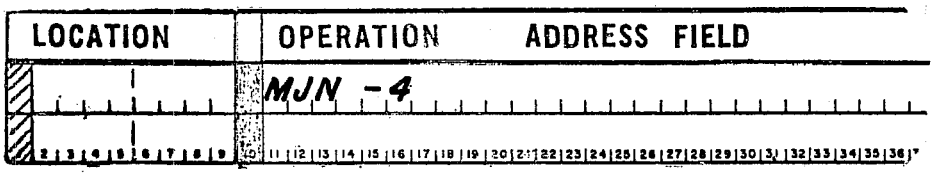
ANSWER



Exercise #2 - Code an instruction that tests the A-Register for a negative number and jumps back 4 locations.

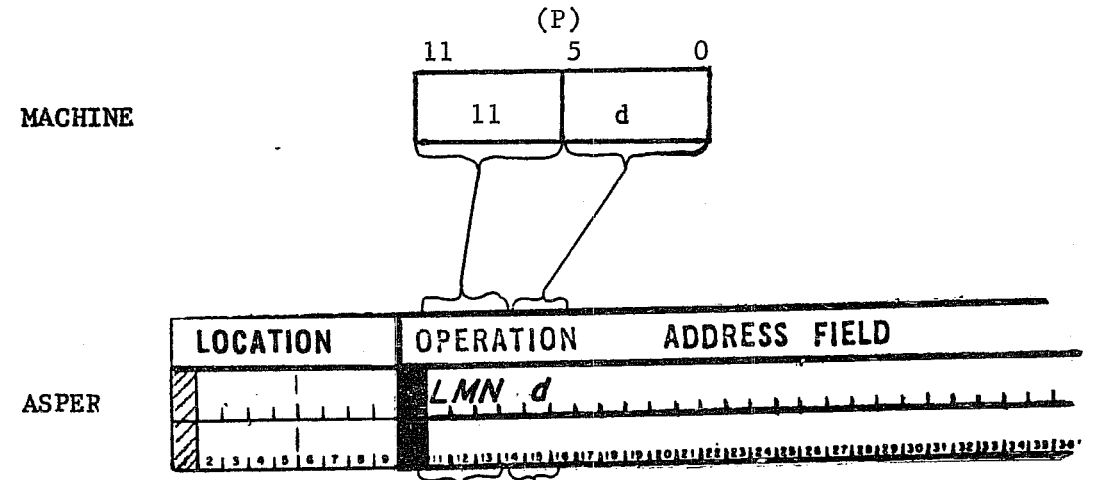
Note: See note page 25-2.

ANSWER



FORMATS

LØGICAL DIFFERENCE
NØ ADDRESS



Constant
Symbol
Symbol ± Constant
Symbol ± Symbol

The above values must result in an octal value in the range of 00-77. The decimal equivalents are 00-63.

Mnemonic Operation Code

DESCRIPTION

The instruction forms in the A-Register the bit by bit logical difference of d and the lower 6 bits of A. This is equivalent to complementing the individual bits in A which correspond to bits in d equal to one. The upper 12-bits of A are not altered.

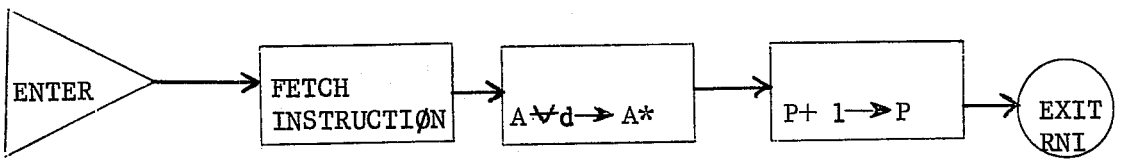
Example:
A = 001110101011001001
d = 001010
001110101011000011

RNI @ P+1

REFERENCES :

LMN

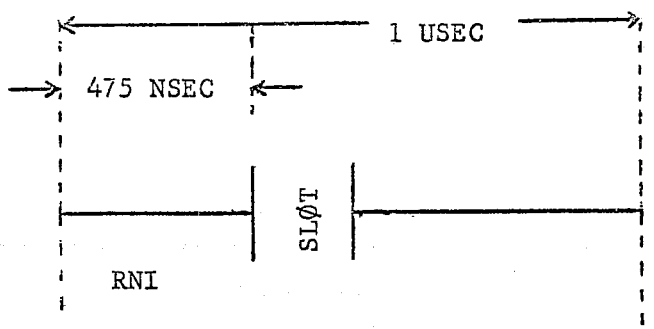
FLOW DIAGRAM



TIMING

6400/6600

1 USEC

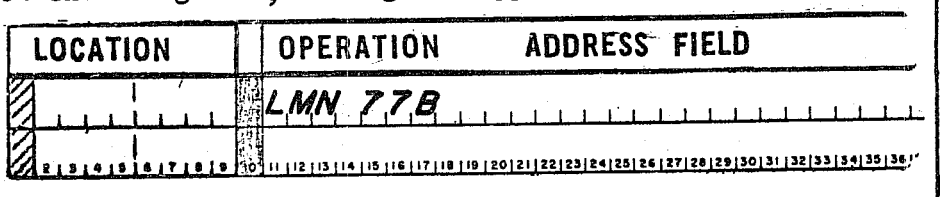


SLØT TIME = 100 NSEC

EXAMPLES

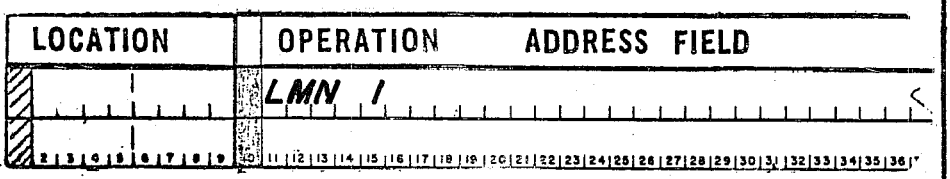
Exercise #1 - Code an instruction that would complement the lower 6-bits of the A-Register, leaving the upper 12-bits unaltered.

ANSWER



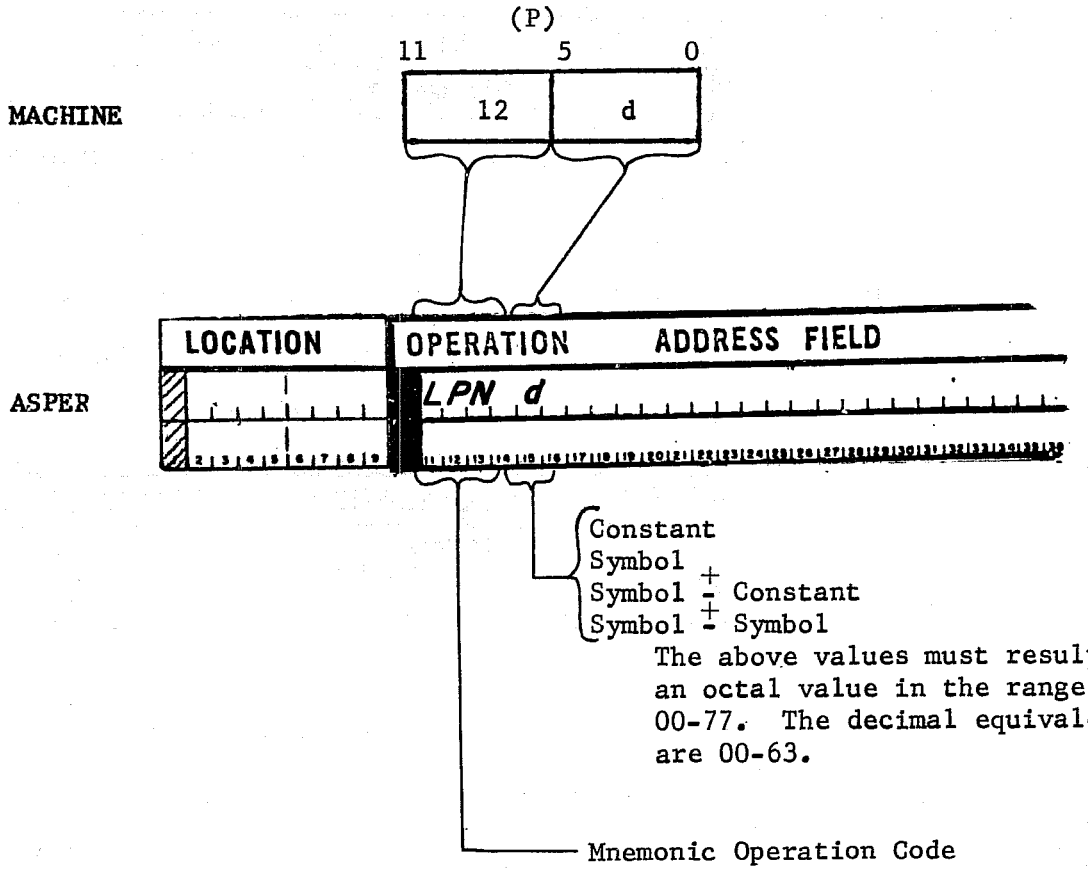
Exercise #2 - Assume the A-Register is zero, code an instruction that will set the 2^0 bit position to a one for a flag indicator.

ANSWER



FORMATS

LØGICAL PRØDUCT
NØ ADDRESS



DESCRIPTION

This instruction forms in the A-Register the bit by bit logical product of d and the lower 6 bits of A. In the final result the upper 12 bits of A are zero.

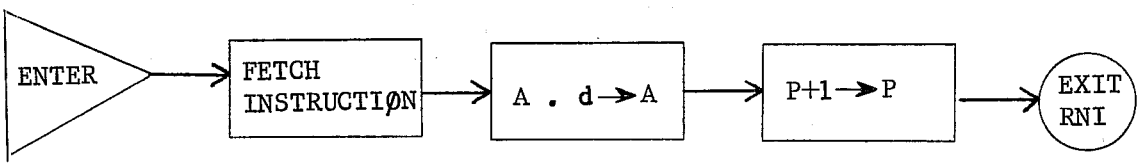
Example: A = 001110101011001001
d = 001010
000000000000001000

RNI @ P+1

REFERENCES :

LPN

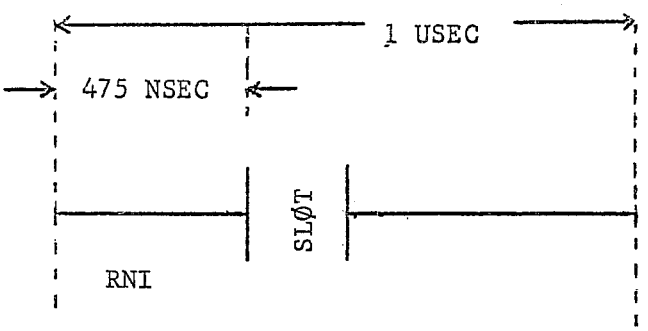
FLOW DIAGRAM



TIMING

6400/6600

1 USEC

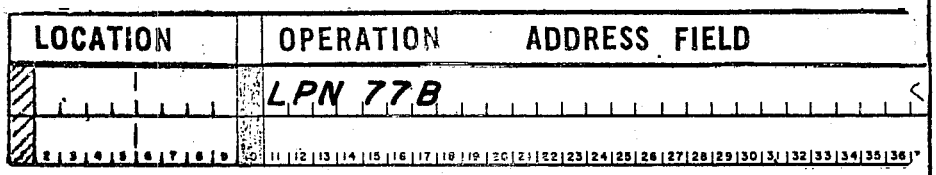


SLØT TIME = 100 NSEC

EXAMPLES

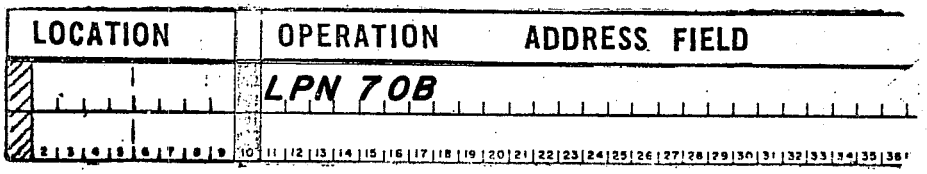
Exercise #1 - Code an instruction that saves just the lower 6-bits of the A-Register, setting the upper 12 bits to zero.

ANSWER



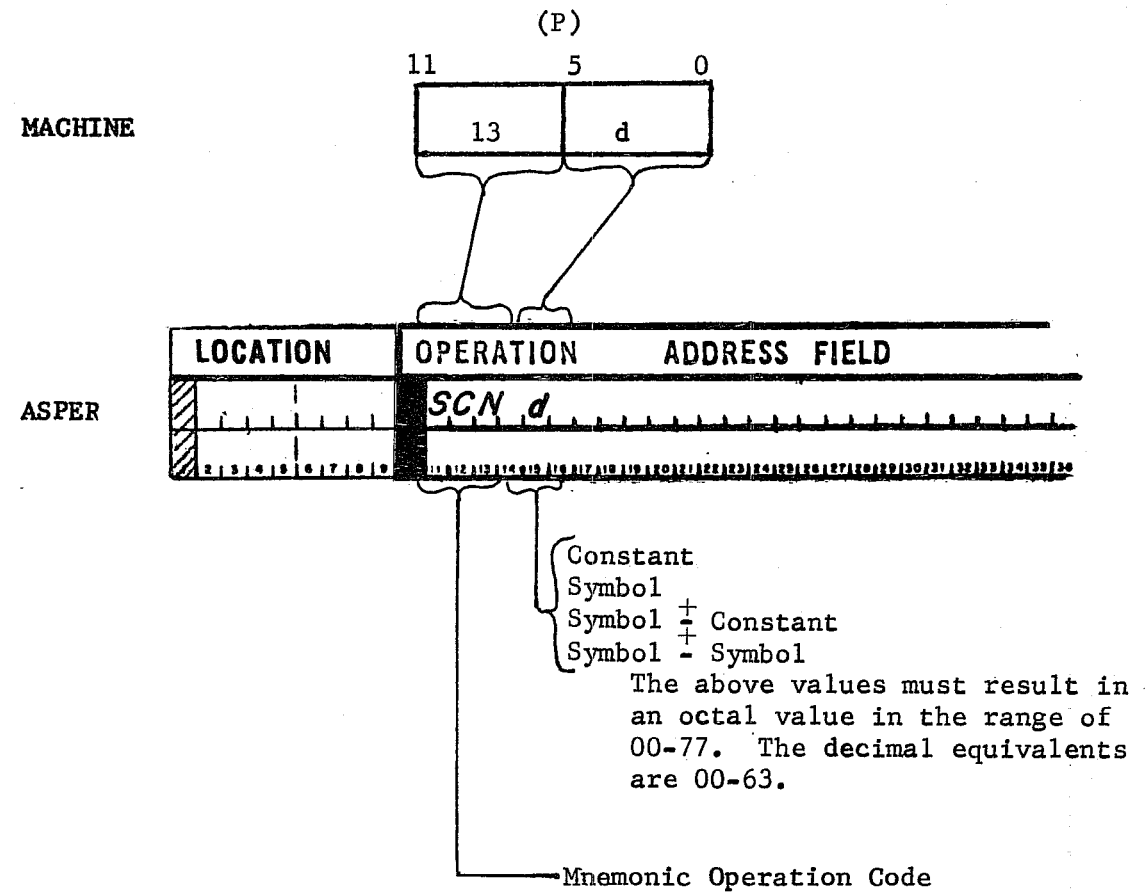
Exercise #2 - Code an instruction that saves the 2^3 through the 2^5 bits of the A-Register, setting the rest to zeros.

ANSWER



FORMATS

SELECTIVE CLEAR
NØ ADDRESS



DESCRIPTION

This instruction clears any of the lower 6 bits of the A-Register where corresponding bits of d are one. The upper 12 bits of A are not altered.

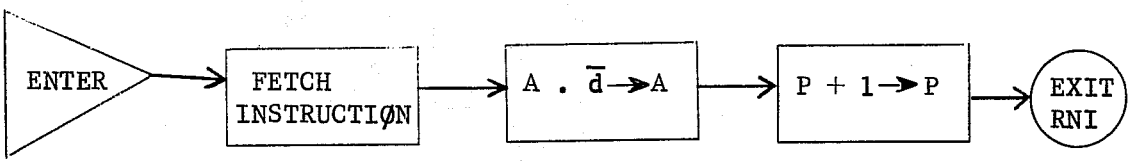
Example:
A = 001110101011001001
d = 001010
001110101011000001

RNI @ P+1

REFERENCES :

SCN

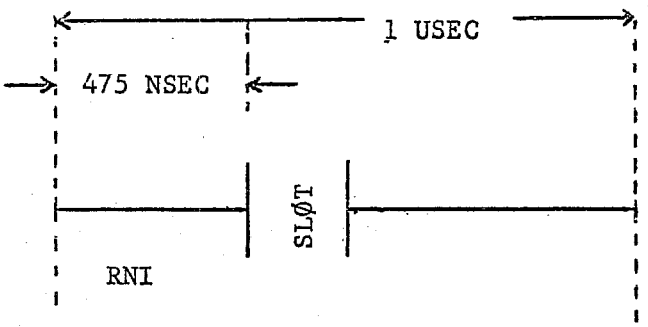
FLOW DIAGRAM



TIMING

6400/6600

1 USEC

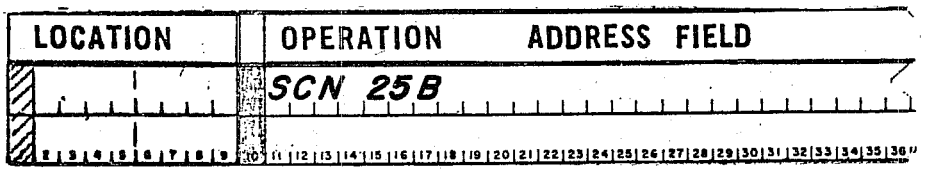


SLØT TIME = 100 NSEC

EXAMPLES

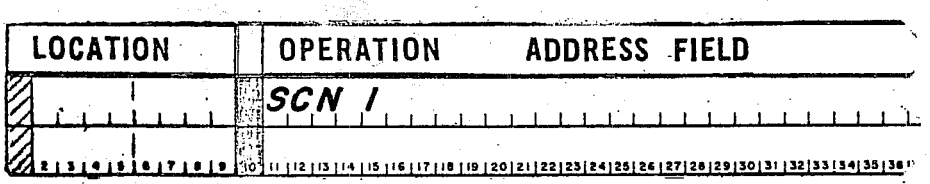
Exercise #1 - Code an instruction that clears any of the lower 6-bits of A that correspond to the 2^0, 2^2, 2^4 bit positions.

ANSWER



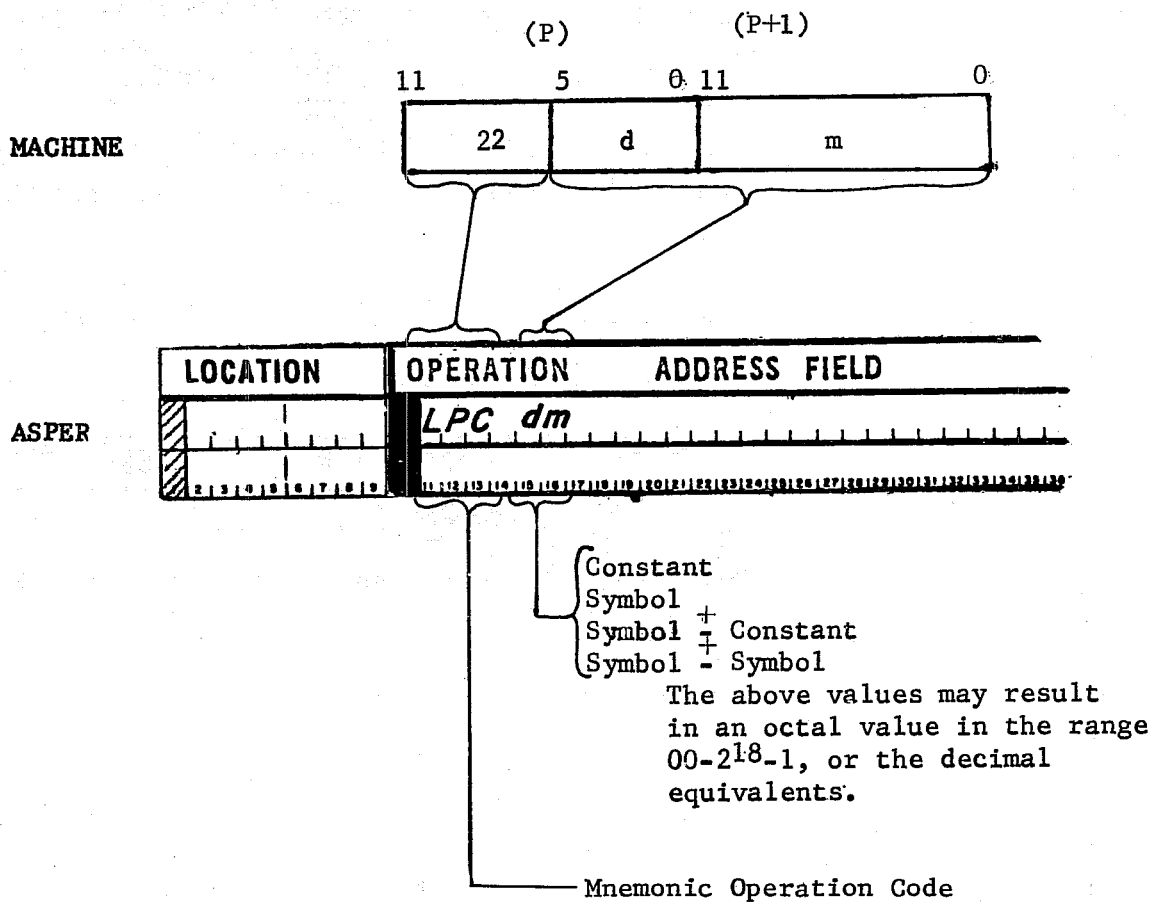
Exercise #2 - Code an instruction that clears Flag Bit 2^0 and leaves the other flag-bits unaltered.

ANSWER



FORMATS

LØGICAL PRØDUCT
CØNSTANT



DESCRIPTION

This instruction forms in the A-Register the bit by bit logical product of the contents of A and the 18-bit quantity dm, consisting of d as the upper 6 bits and m as the lower 12 bits.

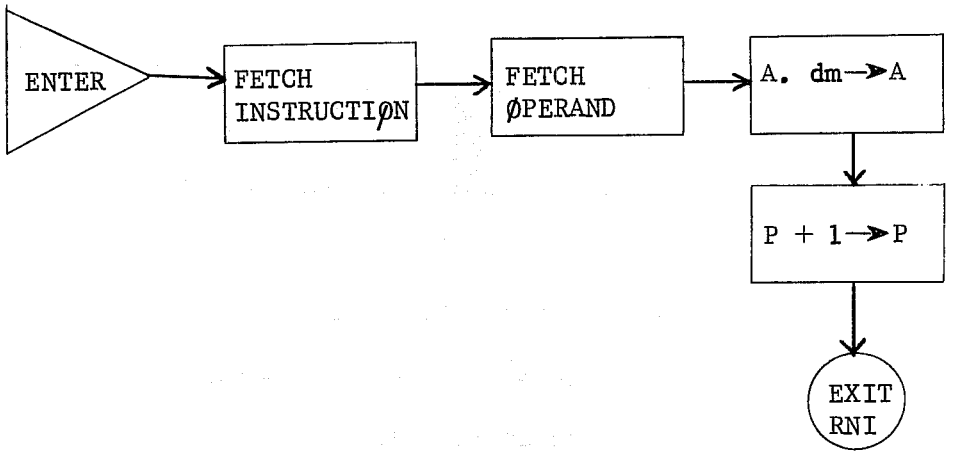
Example:
A = 001110101011001001
dm = 001110000011001010
001110000011001000

RNI @ P+2

REFERENCES :

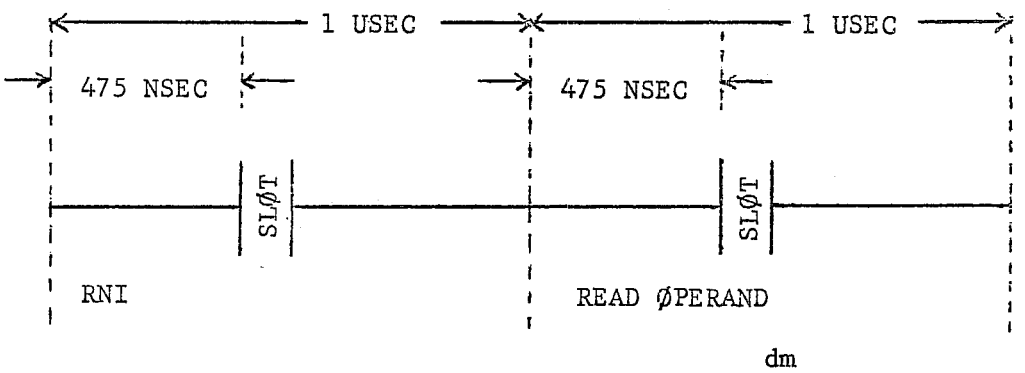
LPC

FLOW DIAGRAM



TIMING

6400/6600 2 USEC

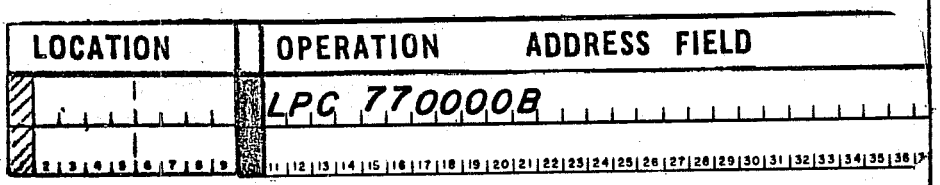


SLØT TIME = 100 NSEC

EXAMPLES

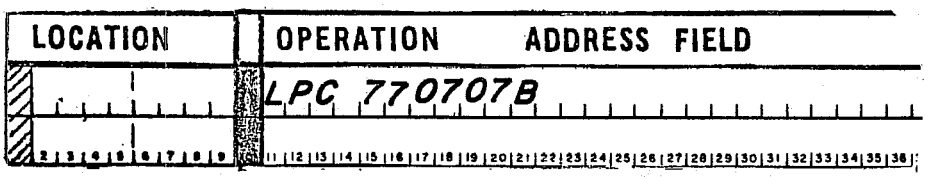
Exercise #1 - Code an instruction that saves the upper 6-bits of the A-Register and sets the lower 12 bits to zeros.

ANSWER



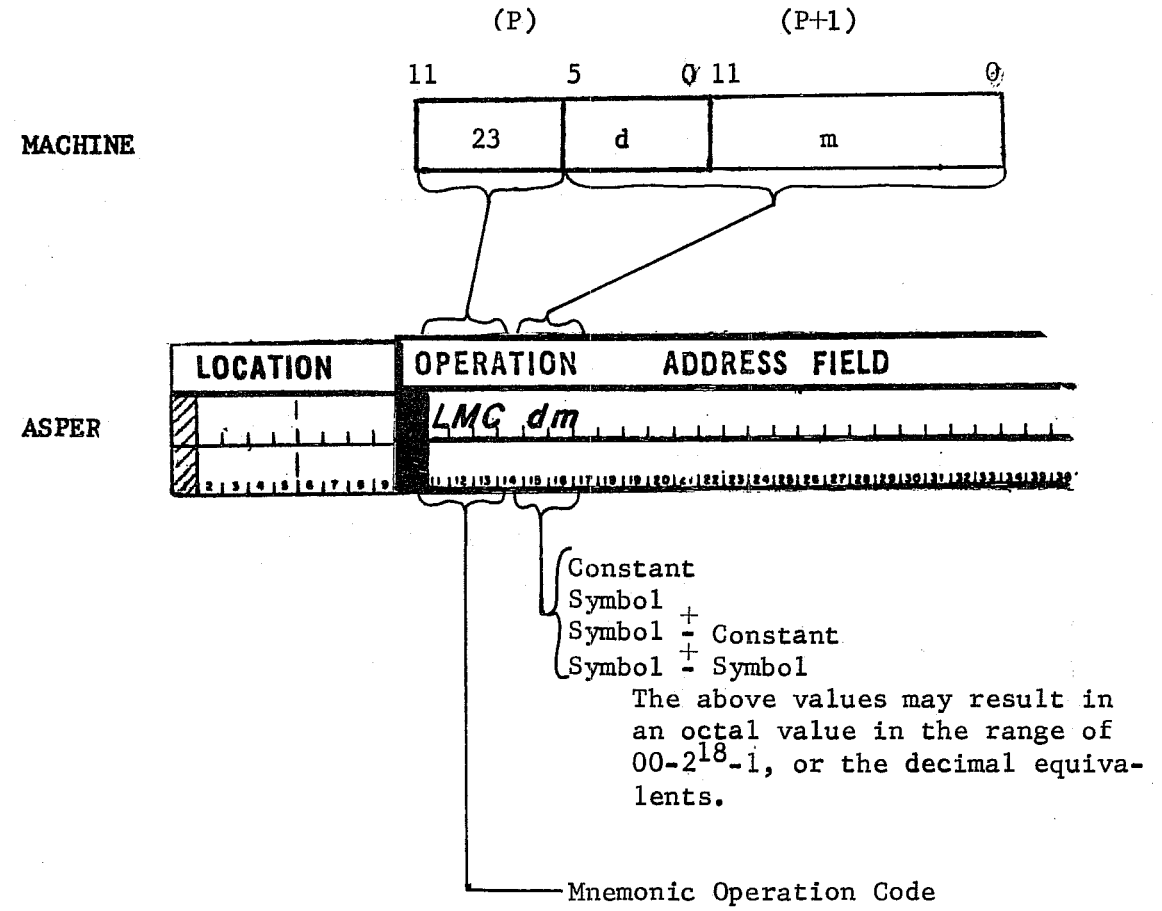
Exercise #2 - Code an instruction that clears the 2^3 through 2^5 and 2^9 through 2^{11} bits in the A-Register, and leaves the other bits unaltered.

ANSWER



FORMATS

LØGICAL DIFFERENCE CØNSTANT



DESCRIPTION

This instruction forms in the A-Register the bit by bit logical difference of the contents of A and the 18-bit quantity dm, consisting of d as the upper 6-bits and m as the lower 12 bits.

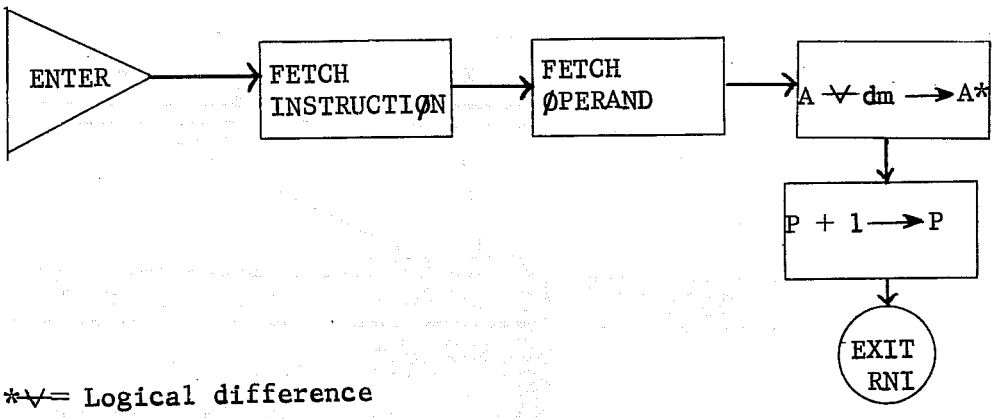
Example:
A = 001110101011001001
dm = 000010000000001010
001100101011000011

RNI @ P+2

REFERENCES :

LMC

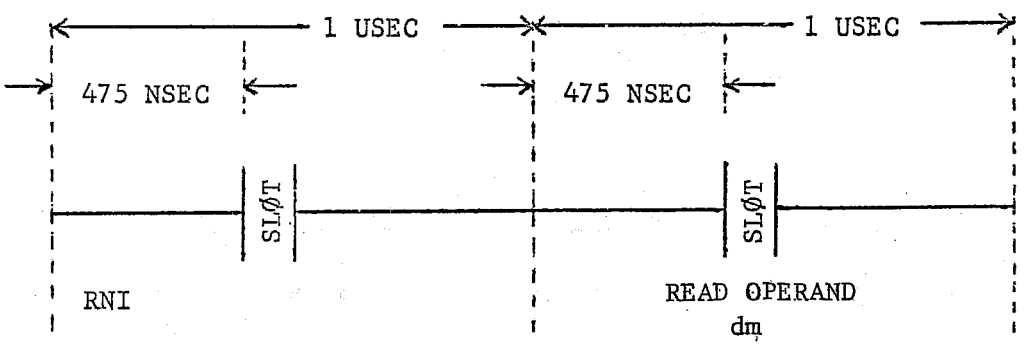
FLOW DIAGRAM



*∨ = Logical difference

TIMING

6400/6600 2 USEC

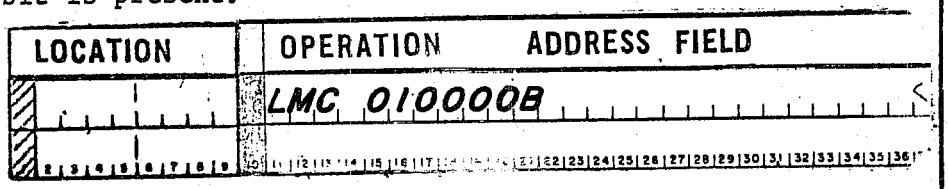


SLØT TIME = 100 NSEC

EXAMPLES

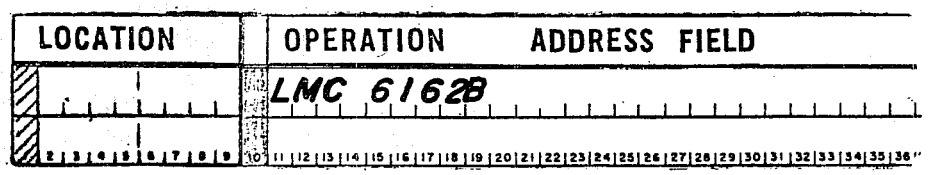
Exercise #1 - The 2^{12} bit in the A-Register may be set, the others will be zero if the results from previous were correct. Code the 1st of two instructions that could test to see if the bit is present.

ANSWER



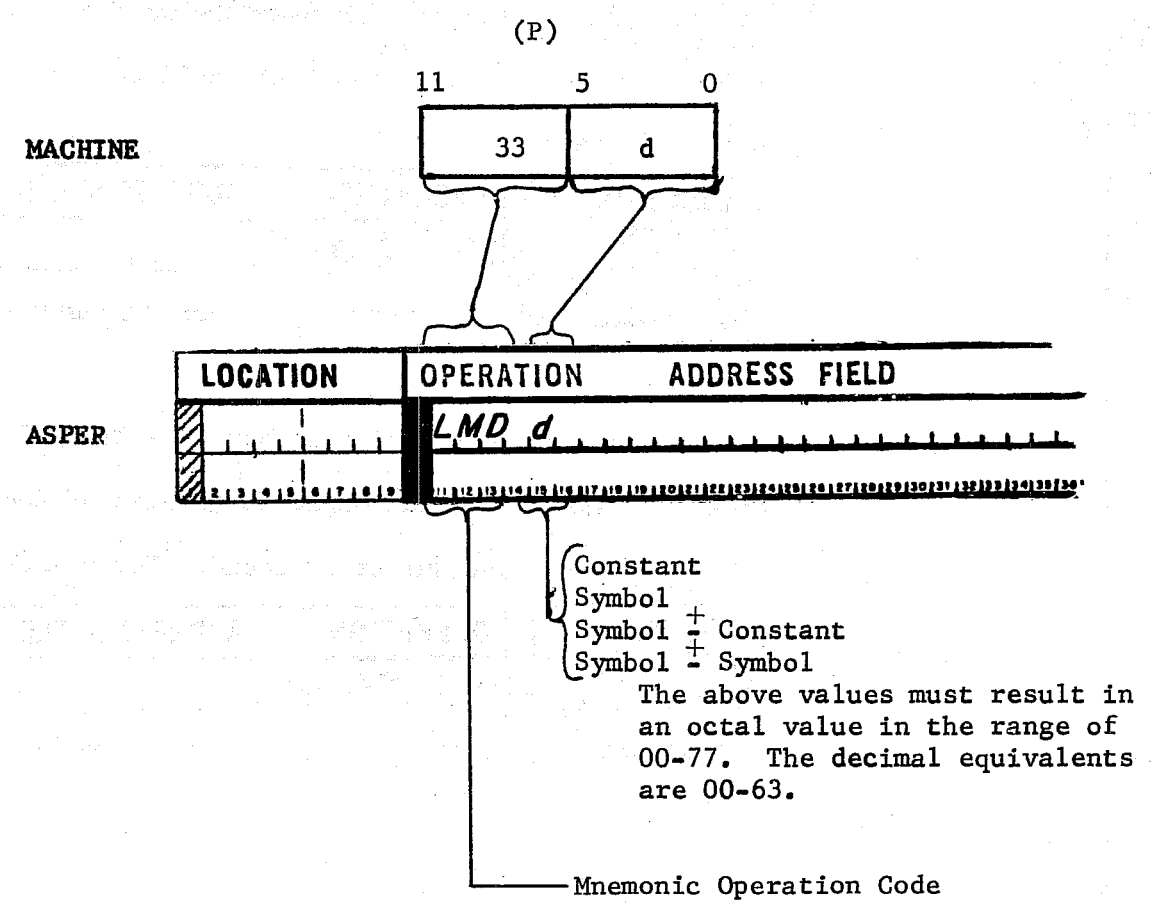
Exercise #2 - In looking for a particular number in a list of numbers. Code an instruction that might be the 1st of two that tests to see if the number is found. The number is 6162.

ANSWER



FORMATS

LØGICAL DIFFERENCE
DIRECT



DESCRIPTION

This instruction forms in the A-Register the bit by bit logical difference of the lower 12 bits of A and the contents of location d (00dd) . This is equivalent to complementing individual bits of A which correspond to one bits in the contents of location d. The upper 6-bits of A are not altered.

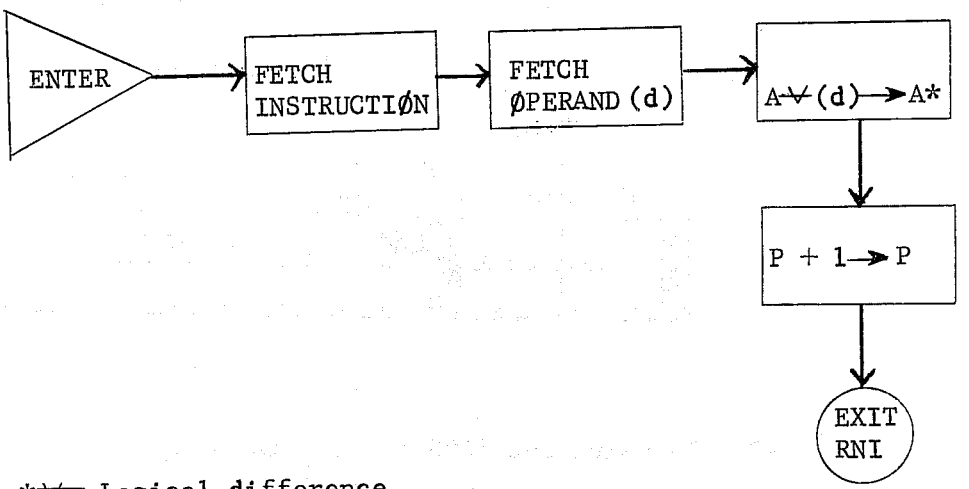
Example: A = 001110101011001001
(d)= 010100001010
001110111111000011

RNI @ P+1

REFERENCES :

LMD

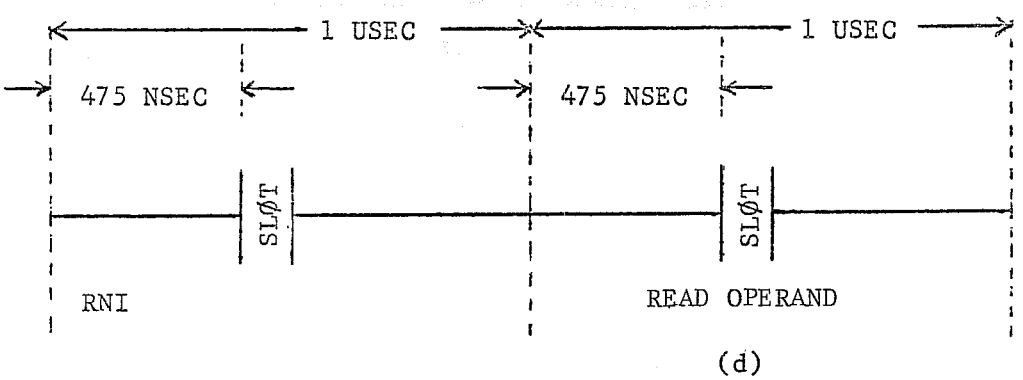
FLOW DIAGRAM



*∀= Logical difference

TIMING

6400/6600 2 USEC

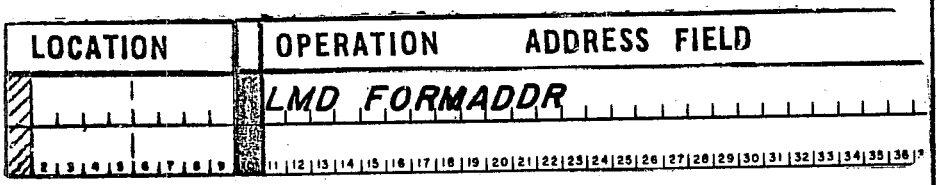


SLØT TIME = 100 NSEC

EXAMPLES

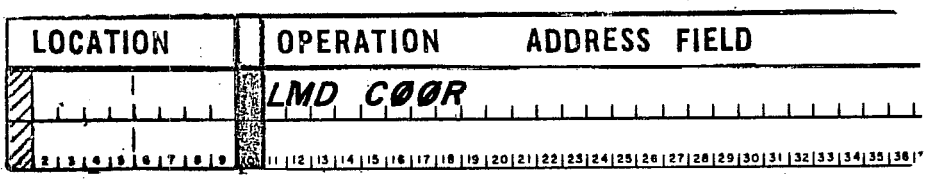
Exercise #1 - The A-Register contains 770000. Code an instruction to set a 12-bit number in memory in A, which may have another 12-bit number added to it. Let d, which is some value 00-77, be called FØRMADDR.

ANSWER



Exercise #2 - In using the 6600 scope (display), A contains the X-coordinate digit (bits 2^9-2^{11}). Code an instruction that will set the coordinate position in the A-Register along with the coordinate. The 1st reference is to CØØR which is some value 00-77.

ANSWER

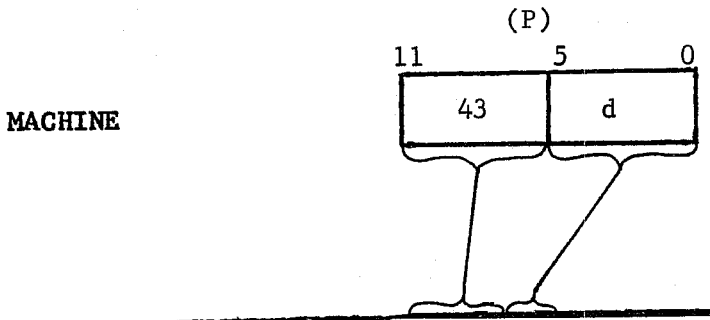


Note: Assume the operand is some value 0XXX.

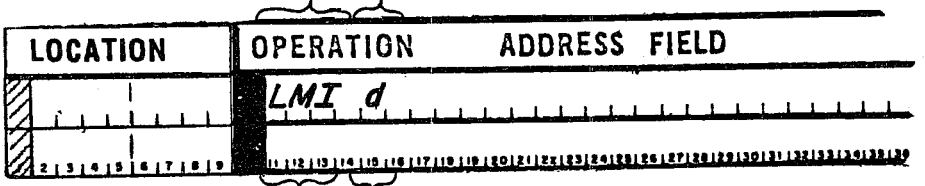
FORMATS

LØGICAL DIFFERENCE
INDIRECT

MACHINE



ASPER



Constant
Symbol
Symbol $\pm$ Constant
Symbol - Symbol

The above values must result in an octal value in the range of 00-77. The decimal equivalents are 00-63.

Mnemonic Operation Code

DESCRIPTION

This instruction forms in the A-Register the bit by bit logical difference of the lower 12-bits of A and the 12-bit operand obtained by indirect addressing. This is equivalent to complementing individual bits of A which correspond to one bits in the operand. The upper 6 bits of A are not altered.

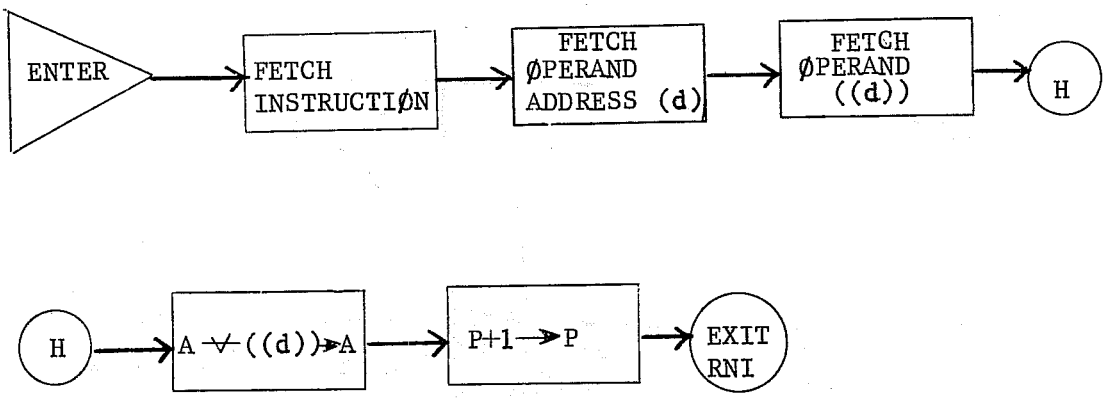
Example: A = 001110101011001001
((d)) = 010100001010
001110111111000011

RNI @ P+1

REFERENCES :

LMI

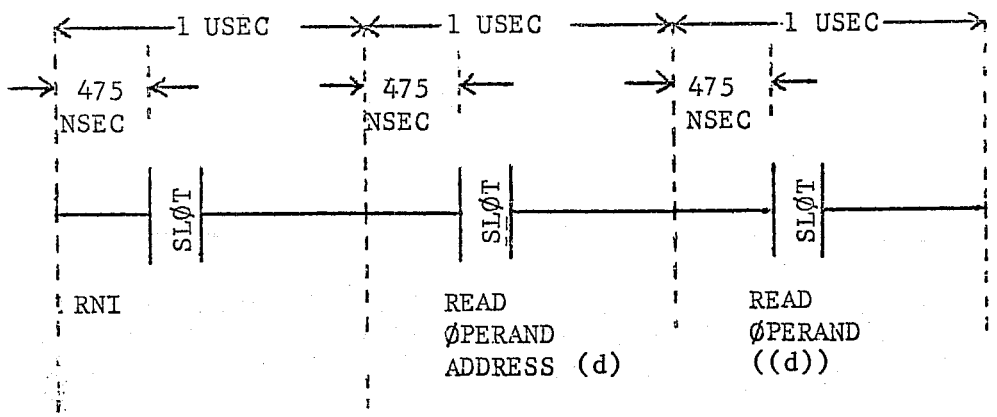
FLOW DIAGRAM



TIMING

6400/6600

3 USEC

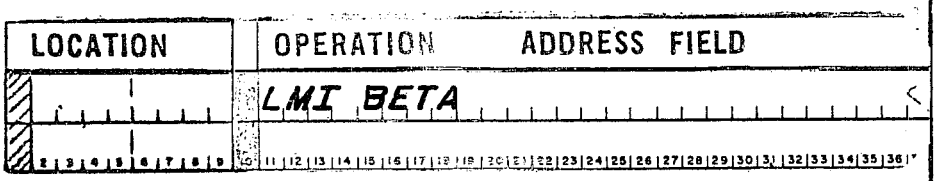


SLØT TIME = 100 NSEC

EXAMPLES

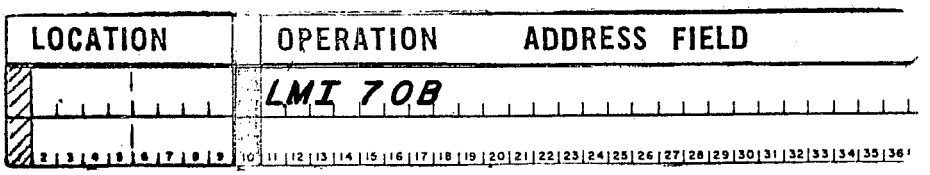
Exercise #1 - In building a word to send out to the display unit, which must be a number 0X0X, we find A contains 0X00. Code an instruction to set in the lower 6-bits. d is the 1st reference called BETA and the contents of BETA might be called GAMMA, which is the address of the operand.

ANSWER



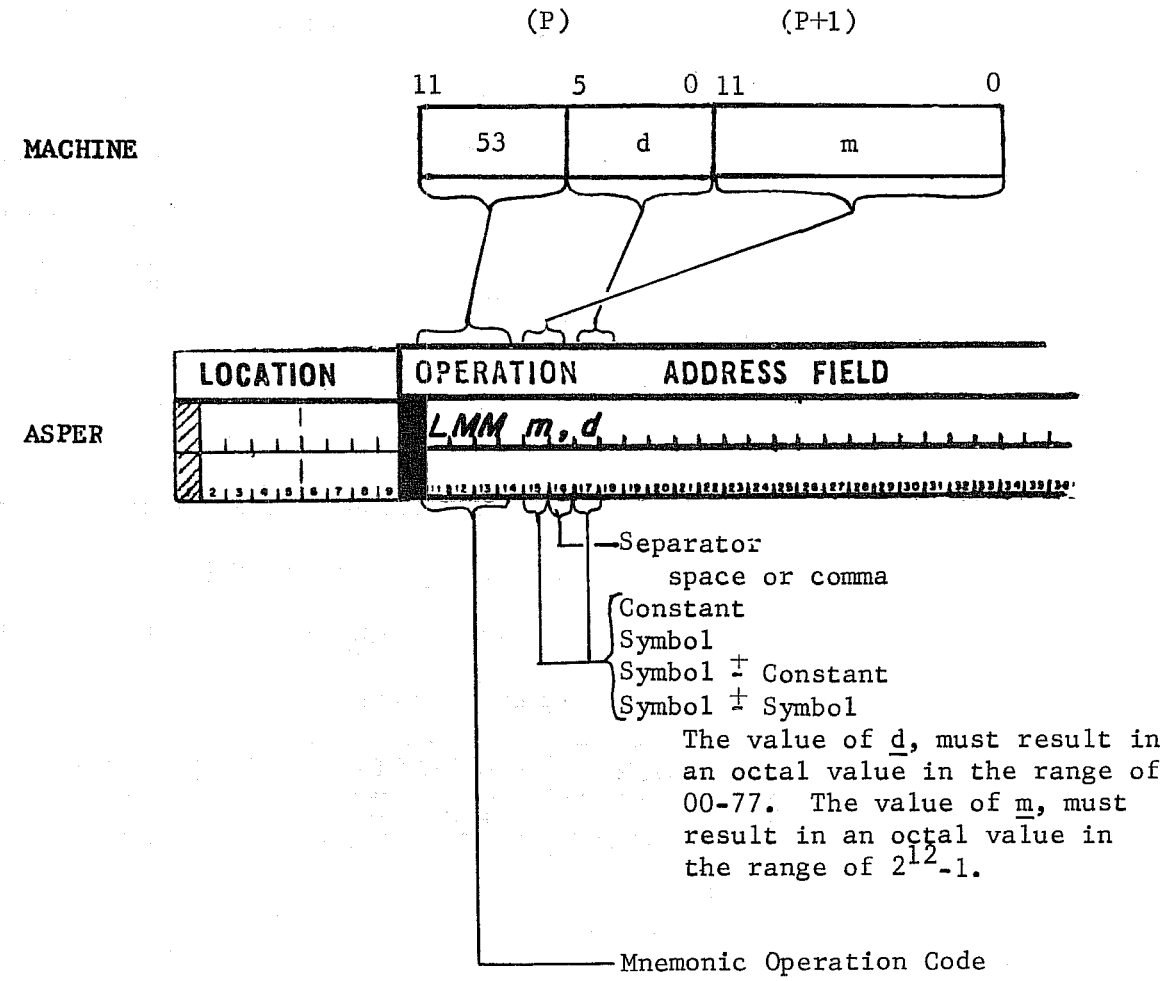
Exercise #2 - Code a one word instruction that makes three memory references and forms the logical difference with the contents of A. Let d = 70_8, and the (70) of = LENGTH, which is the operand address.

ANSWER



FORMATS

LØGICAL DIFFERENCE
MEMØRY INDEX



DESCRIPTION

This instruction forms in the A-Register the bit by bit logical difference of the lower 12 bits of A and a 12 bit operand obtained by indexed addressing. This is equivalent to complementing individual bits of A which correspond to one bits in the operand. The upper 6 bits of A are not altered.

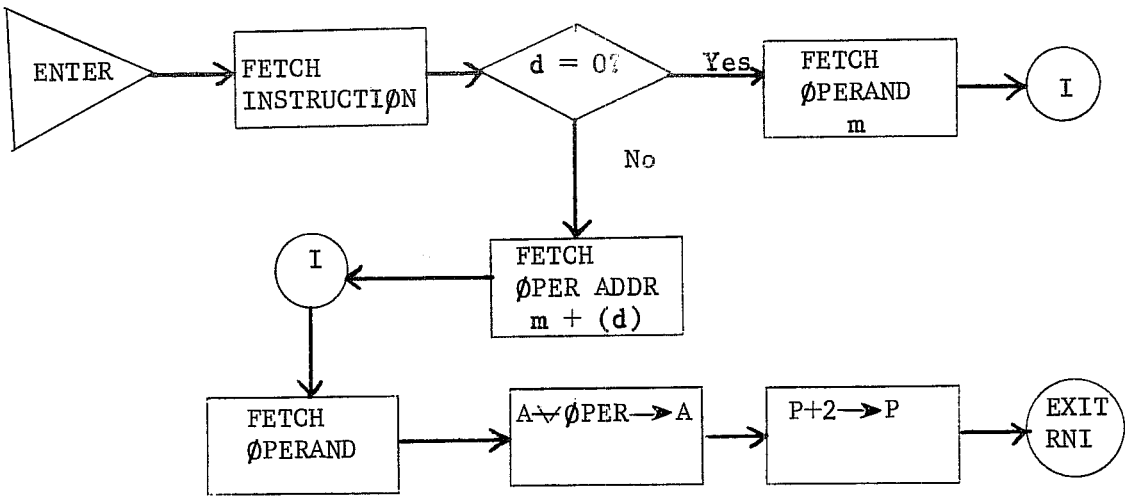
Note: If $d = 0$, the operand address is simply m.
If $d \neq 0$, then m plus the contents of location d, m+(d) is the operand address; thus the contents of d may be used as an index quantity to modify operand addresses.

RNI @ P+2.

REFERENCES :

LMM

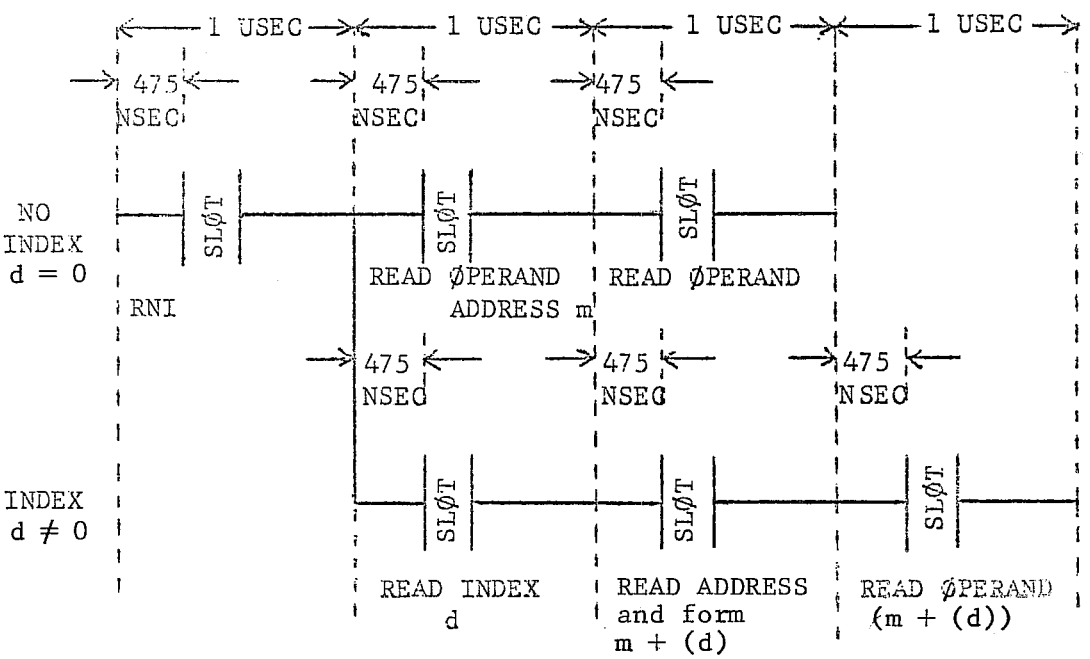
FLOW DIAGRAM



TIMING

6400/6600

3-4 USEC



SLØT TIME = 100 NSEC

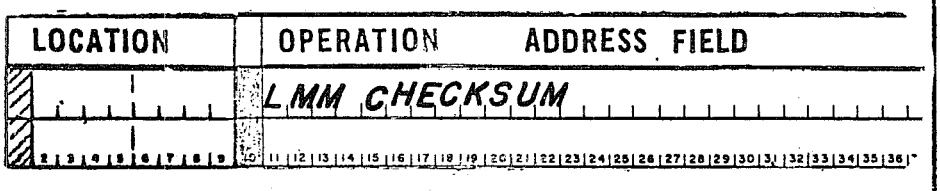
EXAMPLES

Exercise #1 - Code a two word instruction, which makes three memory references and no indexing to perform the logical difference with the contents of A.

Let m = CHECKSUM, and d must = 0.

The contents of CHECKSUM would be the operand.

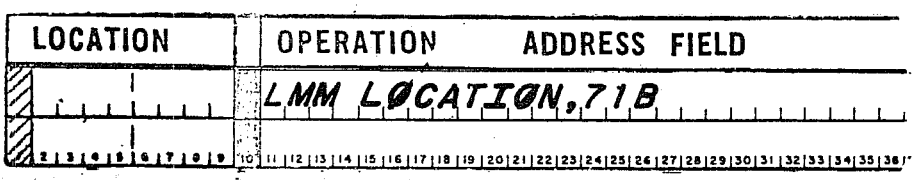
ANSWER



Exercise #2 - Code a two word instruction which makes four memory references, which includes indexing, to perform the logical difference with the contents of A.

Let m = LØCATIØN, which is the operand address, and d = 71_8 (the index location).

ANSWER

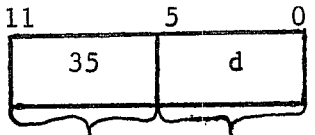


REPLACE ADD DIRECT

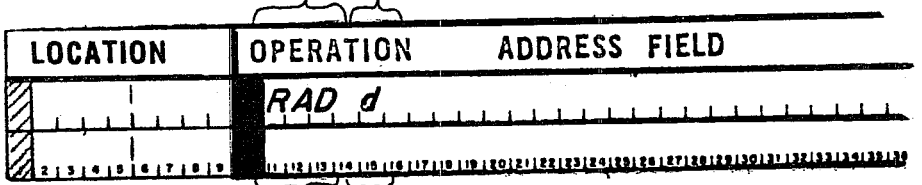
FORMATS

(P)

MACHINE



ASPER



Constant
Symbol
Symbol + Constant
Symbol - Symbol

The above values must result in an octal value in the range of 00-77. The decimal equivalents are 00-63.

Mnemonic Operation Code

DESCRIPTION

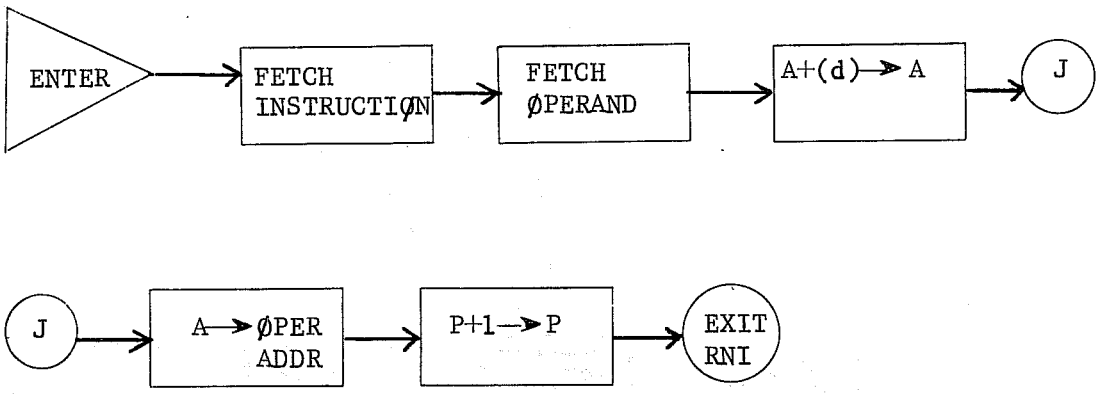
This instruction adds the 12 bit quantity in location d to the contents of the A-Register and stores the lower 12-bits of the result back in location d. The result is also left in the A-Register at the end of the operation.

RNI @ P+1.

REFERENCES :

RAD

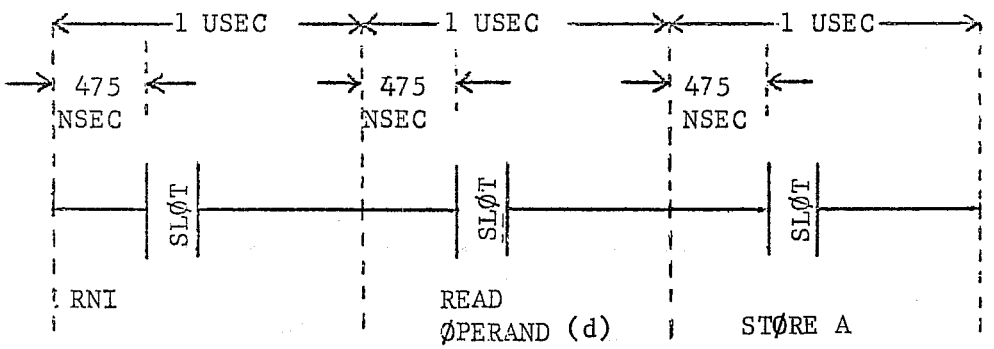
FLOW DIAGRAM



TIMING

6400/6600

3 USEC



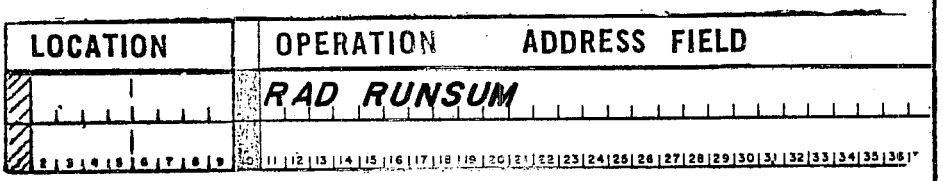
SLØT TIME = 100 NSEC

EXAMPLES

Exercise #1 - Code an instruction to add an operand to the contents of the A-Register and store the results.

Let d = RUNSUM, which is the operand address and has a value in the range of 00-77.

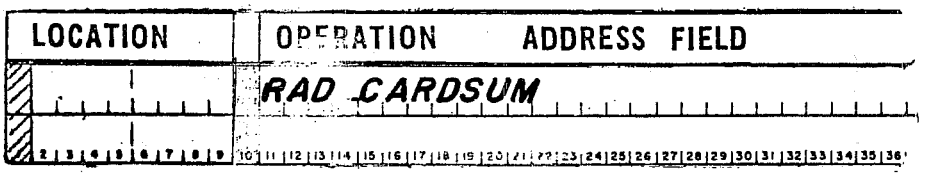
ANSWER



Exercise #2 - Code an instruction to form the checksum of all the information on a 80 column card just read into memory.

Let d = CARDSUM, which is the operand address, and takes on a value in the range of 00-77. The operand in location CARDSUM would start out = to zero.

ANSWER

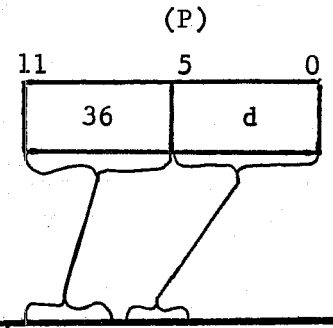


AØD

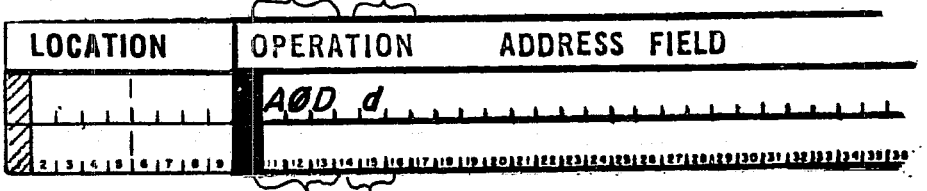
REPLACE ADD ØNE DIRECT

FORMATS

MACHINE



ASPER



Constant
Symbol
Symbol ± Constant
Symbol ± Symbol

The above value must result in an octal value in the range 00-77. The decimal equivalents are 00-63.

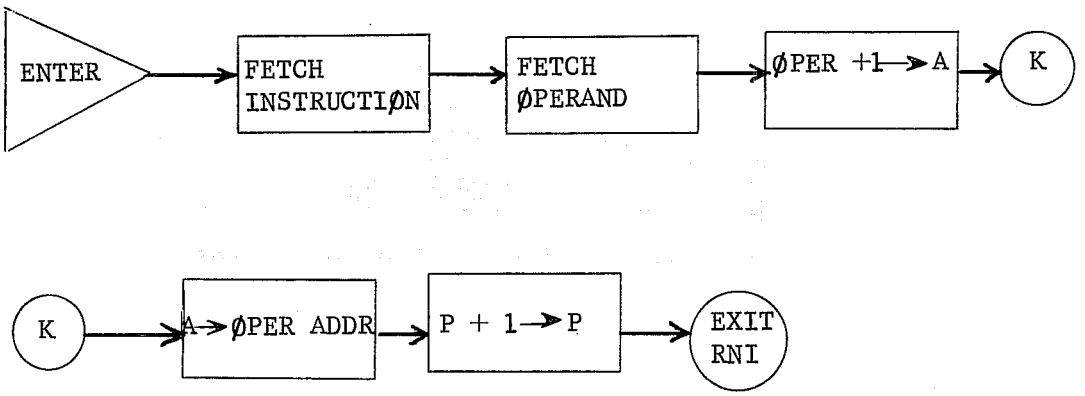
Mnemonic Operation Code

DESCRIPTION

This instruction adds one to the original value in location d and stores the result back in location d. The result is also left in the A-Register at the end of the operation. The original contents of A are destroyed. RNI @ P+1.

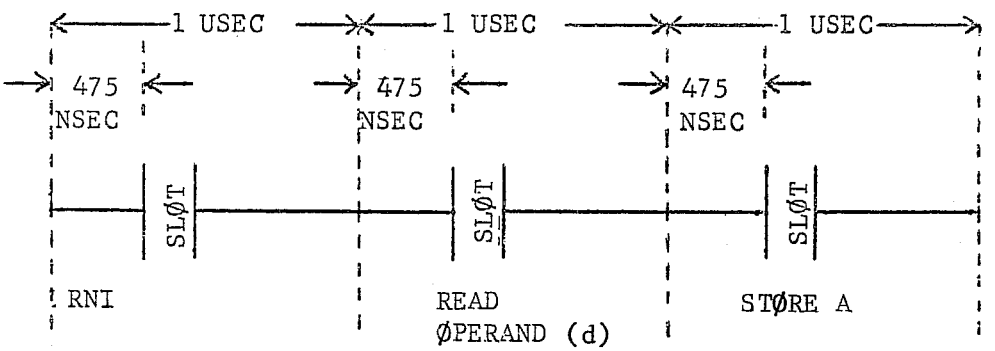
REFERENCES :

FLOW DIAGRAM



TIMING

6400/6600 3 USEC

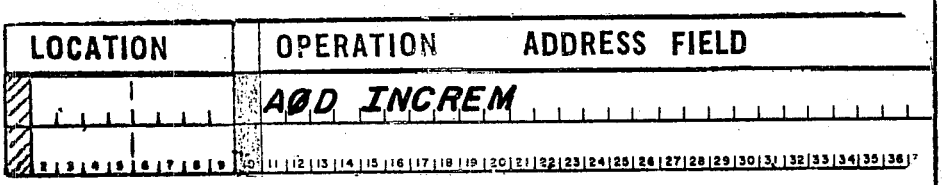


SLØT TIME = 100 NSEC

EXAMPLES

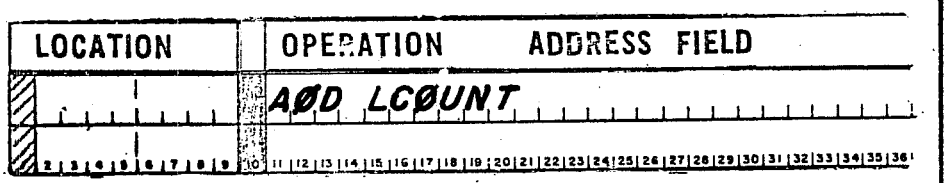
Exercise #1 - Code an instruction to update an index location. Let d, the index location, be called INCREM, which is an address with a value in the range of 00-77.

ANSWER



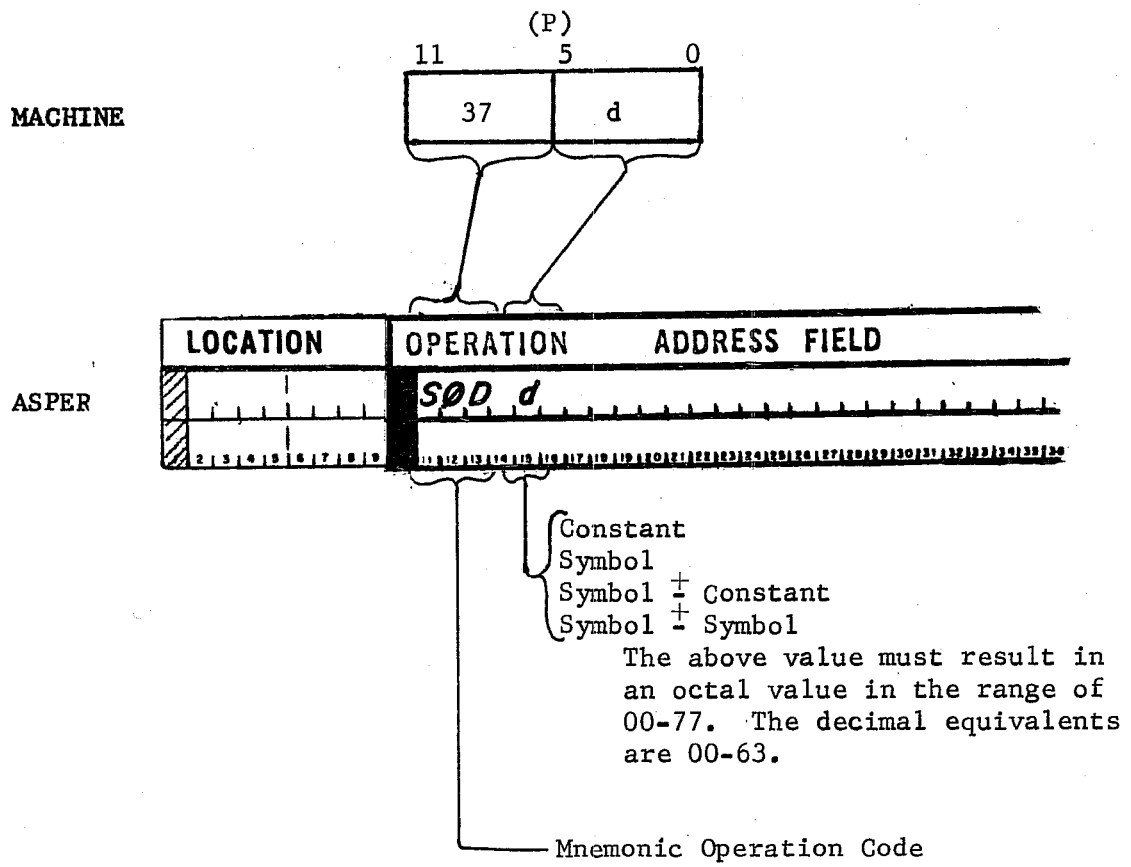
Exercise #2 - Code an instruction to update a loop counter which would leave the result in the A-Register so it could next be tested. Let d = LCØUNT.

ANSWER



FORMATS

REPLACE SUBTRACT ØNE DIRECT



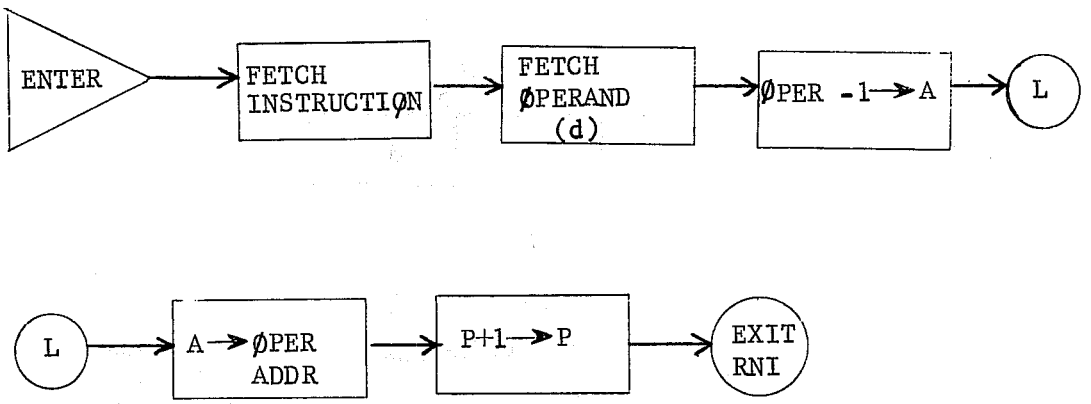
DESCRIPTION

This instruction subtracts one from the original value in location d and stores the result back in location d. The result is also left in the A Register at the end of the operation. The original contents of A are destroyed. RNI @ P+1.

REFERENCES :

SØD

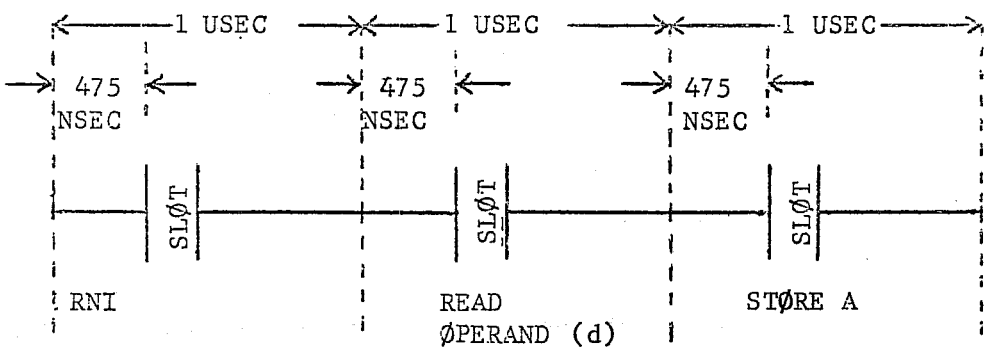
FLOW DIAGRAM



TIMING

6400/6600

3 USEC



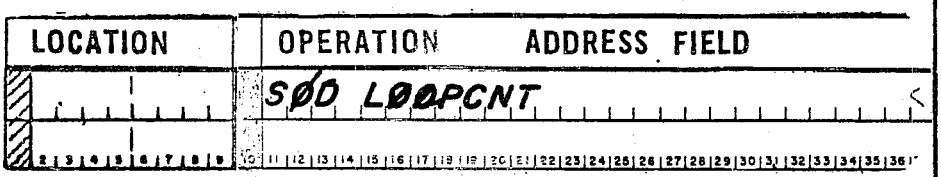
SLØT TIME = 100 NSEC

EXAMPLES

Exerçise #1 - Code an instruction that would decrement a loop counter (by one until it equals zero,) leaving the result in the A-Register each time to be tested.

Let d = LØØPCNT

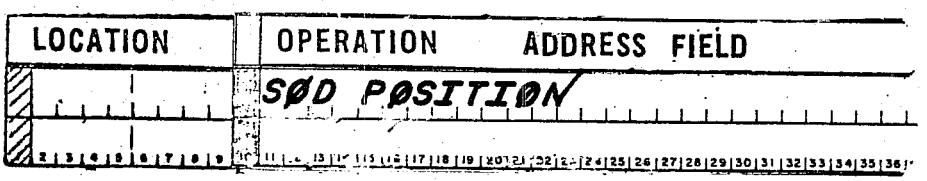
ANSWER



Exercise #2 - Code an instruction that would decrement a Y-coordinate position in Dot Mode using the display unit.

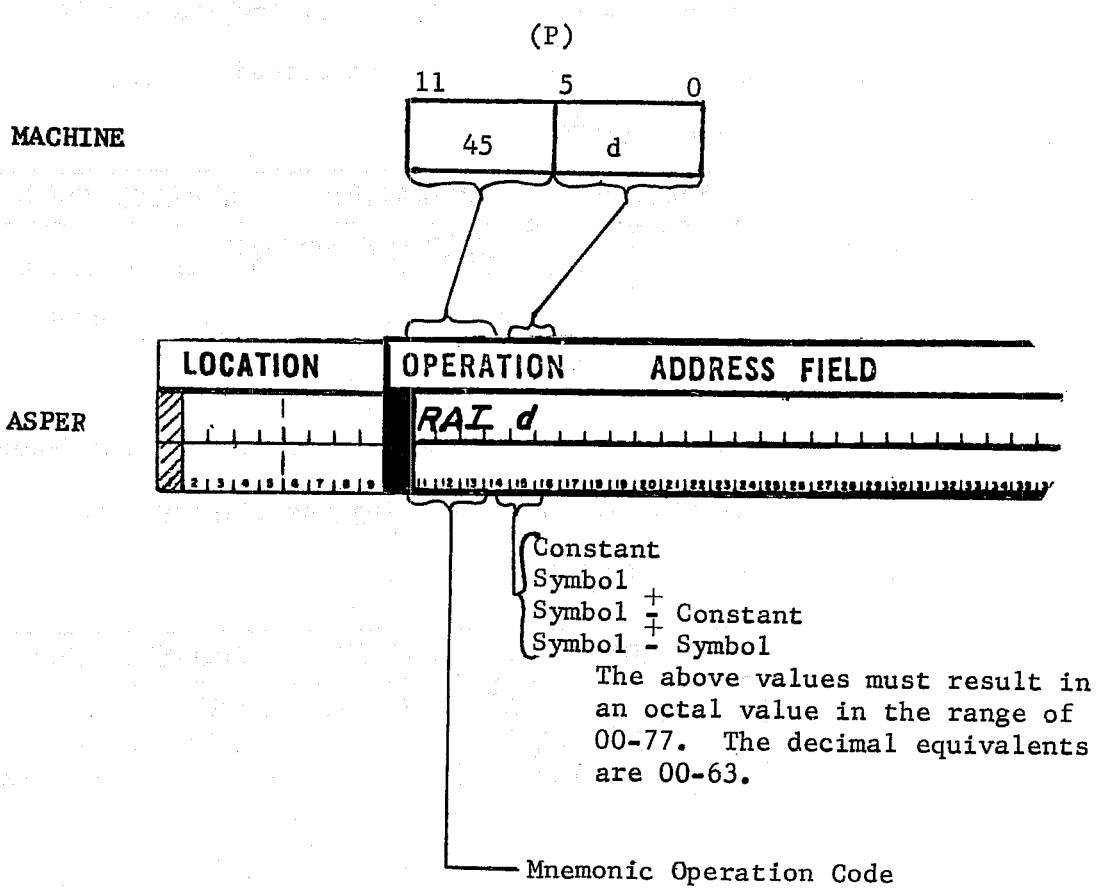
Let d = PØSITIØN

ANSWER



FORMATS

REPLACE ADD INDIRECT



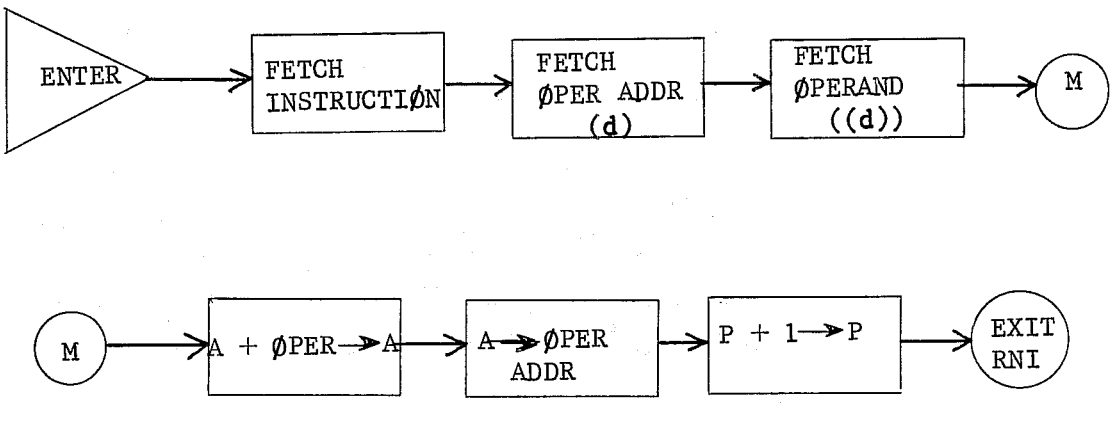
DESCRIPTION

This instruction adds to the contents of the A-Register the operand obtained from indirect addressing. ØPERAND = ((d)) . The resultant sum is left in the A-Register at the end of the operation and the lower 12 bits of A replace the original operand in memory. RNI @ P+1

REFERENCES :

RAI

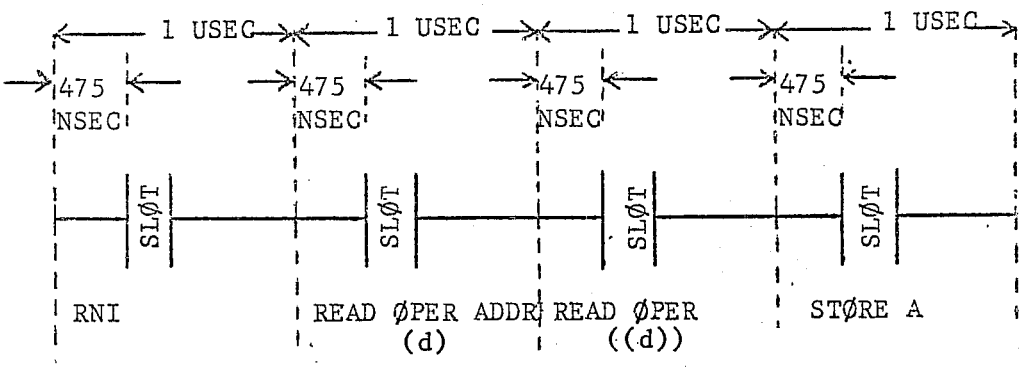
FLOW DIAGRAM



TIMING

6400/6600

4 USEC



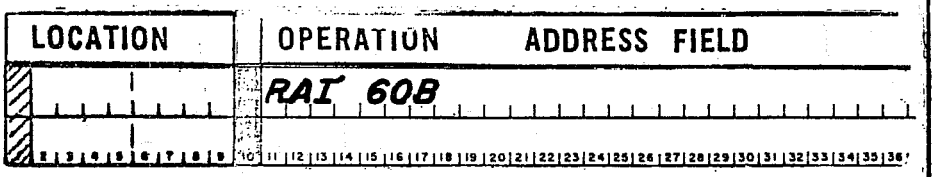
SLØT TIME = 100 NSEC

EXAMPLES

Exercise #1 - Code an instruction to add the contents of A to an operand and replace the original operand with the results using the indirect method.

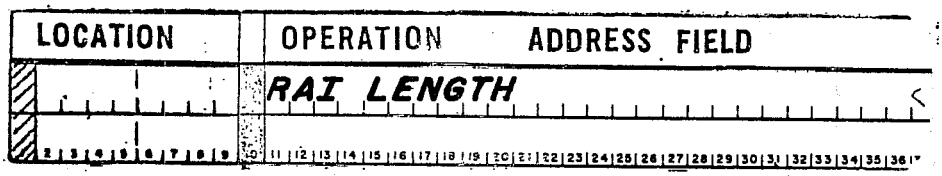
Let d = 60_8, and the contents of 60 = MESSAGE, which is

ANSWER



Exercise #2 - Code a replace add instruction which 1st references location 50_8, called LENGTH, and 2nd location 5000_8, called BUFFER.

ANSWER

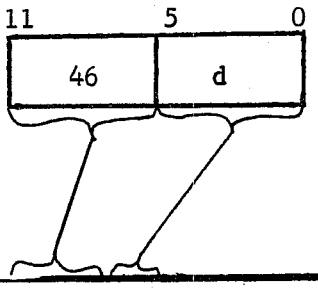


FORMATS

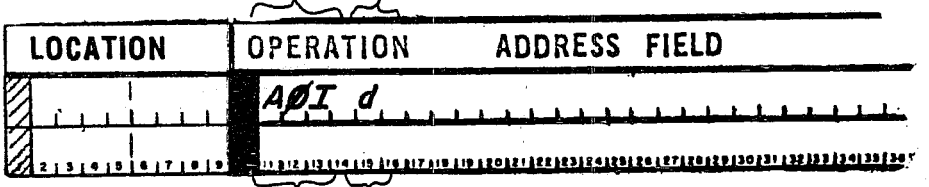
REPLACE ADD ØNE INDIRECT

(P)

MACHINE



ASPER



Constant
Symbol
Symbol ± Constant
Symbol ± Symbol

The above values must result in an octal value in the range of 00-77. The decimal equivalents are 00-63.

Mnemonic Operation Code

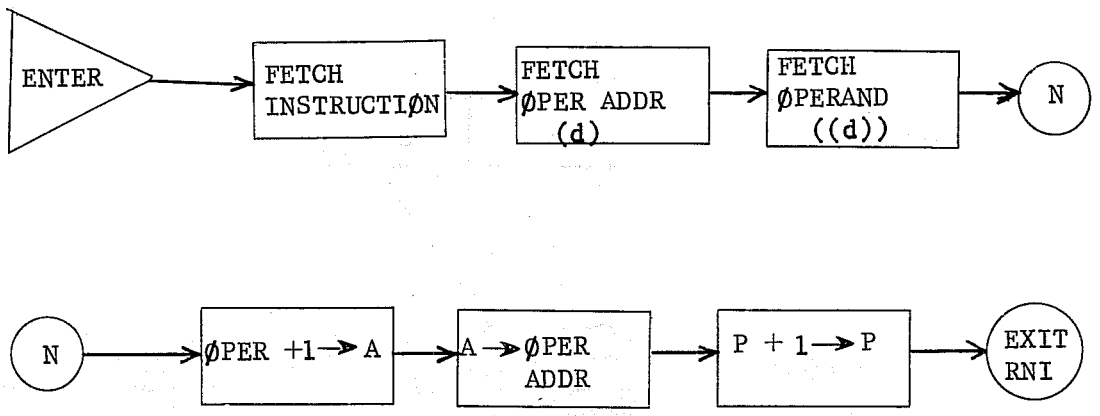
DESCRIPTION

The instruction adds one to the operand obtained from indirect addressing. OPERAND = ((d)) . The resultant sum is left in the A-Register at the end of the operation and the lower 12 bits of A replace the original operand in memory. The original contents of A are destroyed. RNI @ P+1

REFERENCES :

AØI

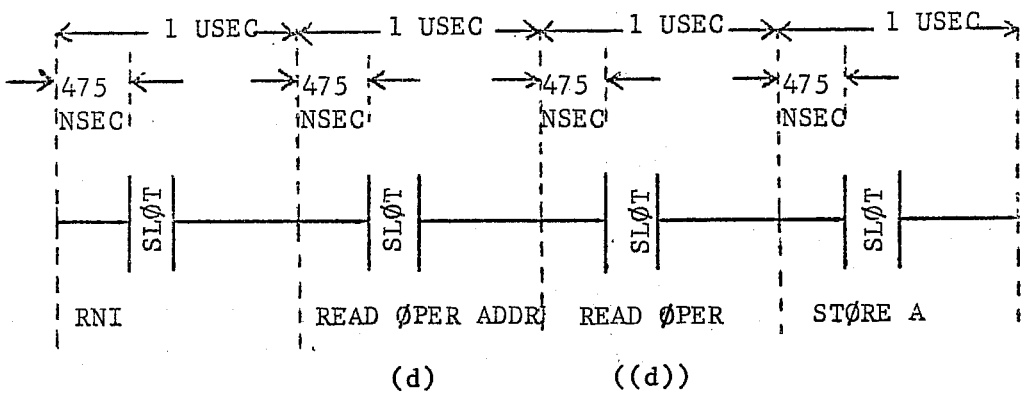
FLOW DIAGRAM



TIMING

6400/6600

4 USEC

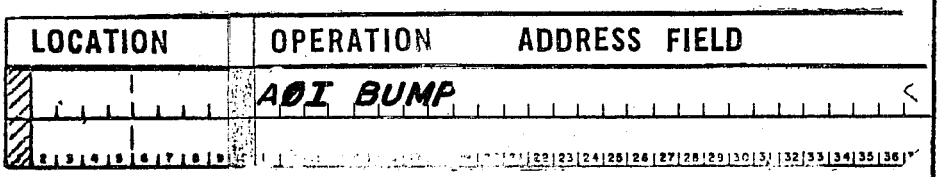


SLØT TIME = 100 NSEC

EXAMPLES

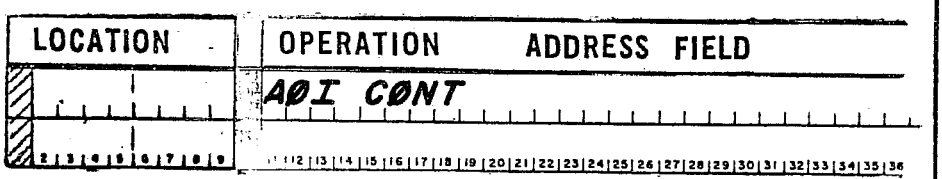
Exercise #1 - Code an instruction to update an operand that is updated from various parts of a program. All 1st references are made to location BUMP and 2nd references to location LAYØUT.

ANSWER



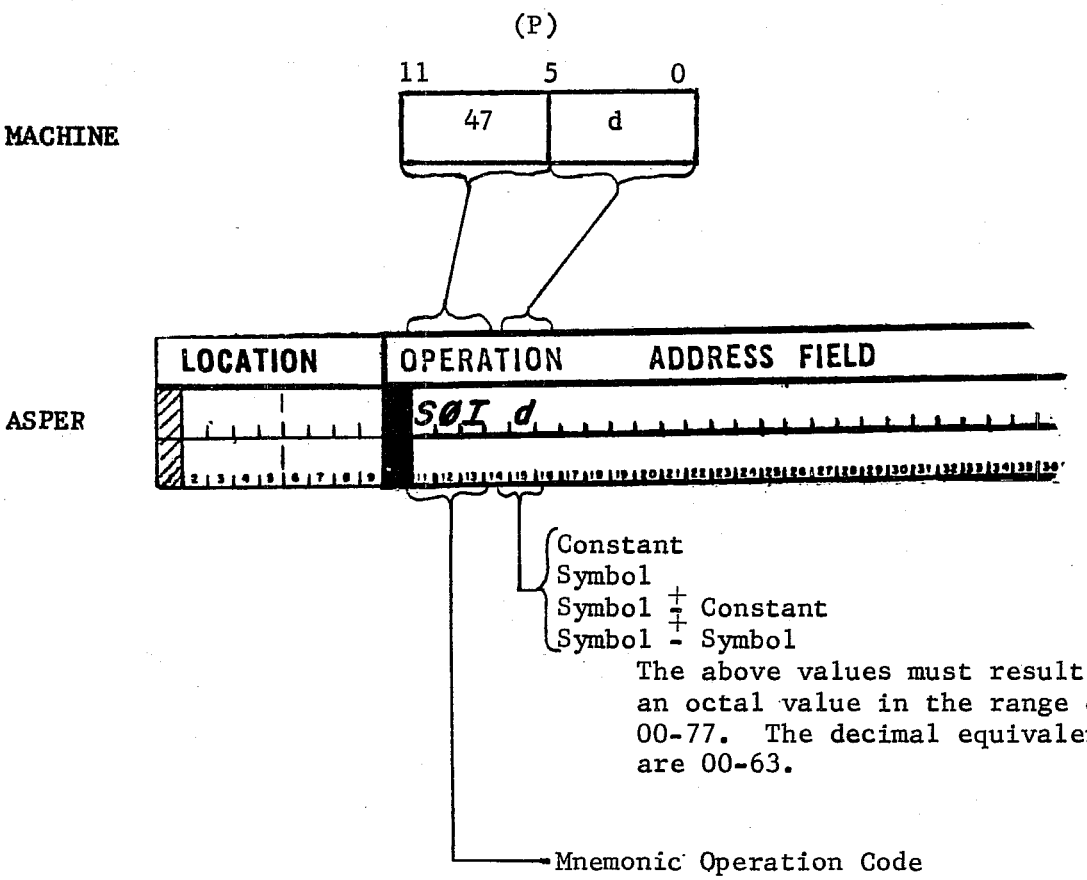
Exercise #2 - Code an instruction which updates a clock count each time the RTC goes through its cycle. Let d = CØNT and the 2nd reference = CLØCK.

ANSWER



FORMATS

REPLACE SUBTRACT ØNE
INDIRECT



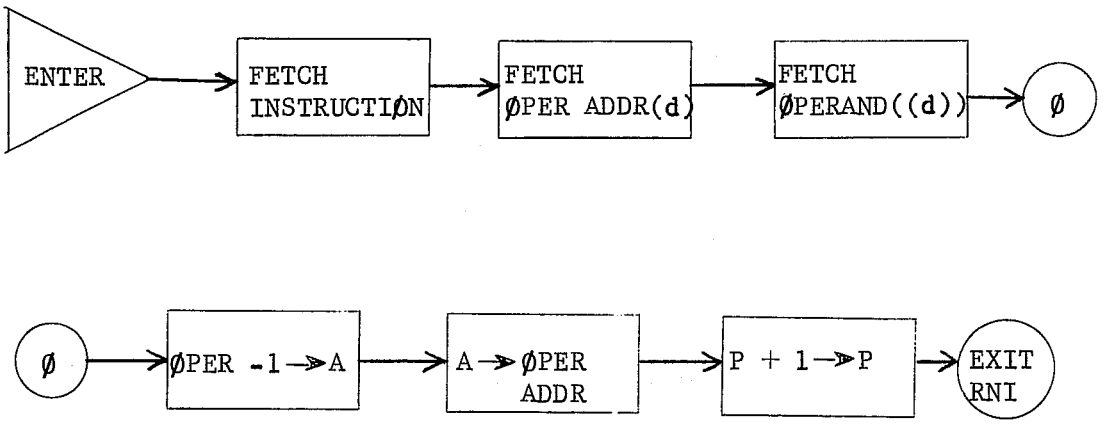
DESCRIPTION

This instruction subtracts one from the operand obtained from indirect addressing, ØPERAND = ((d)) . The resultant difference is left in the A-Register at the end of the operation and the lower 12 bits of A replaces the original operand in memory. The original contents of A are destroyed . RNI @ P+1.

REFERENCES :

SØI

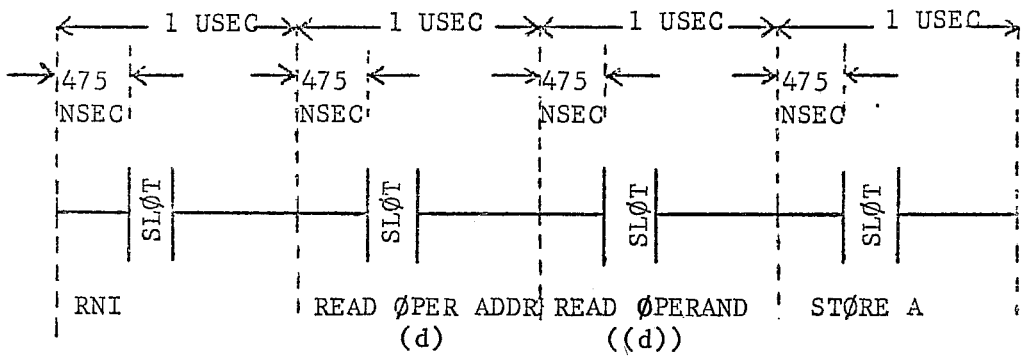
FLOW DIAGRAM



TIMING

6400/6600

4 USEC

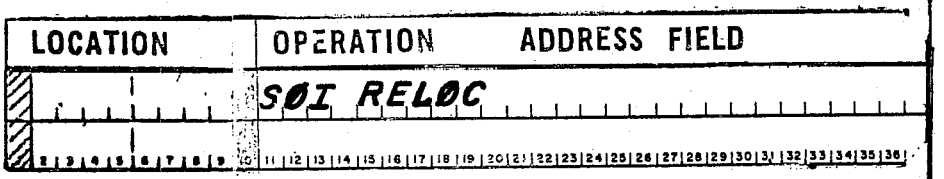


SLØT TIME = 100 NSEC

EXAMPLES

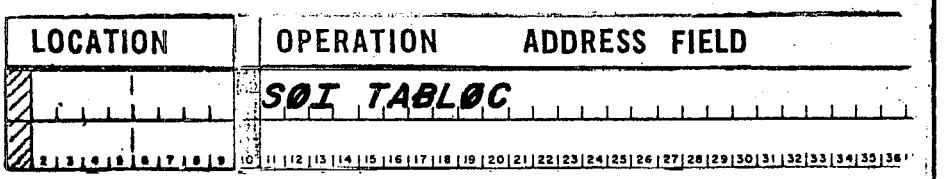
Exercise #1 - Code an instruction which decrements a record count and makes a 1st reference to location RELØC and a 2nd reference to location RECØRD.

ANSWER



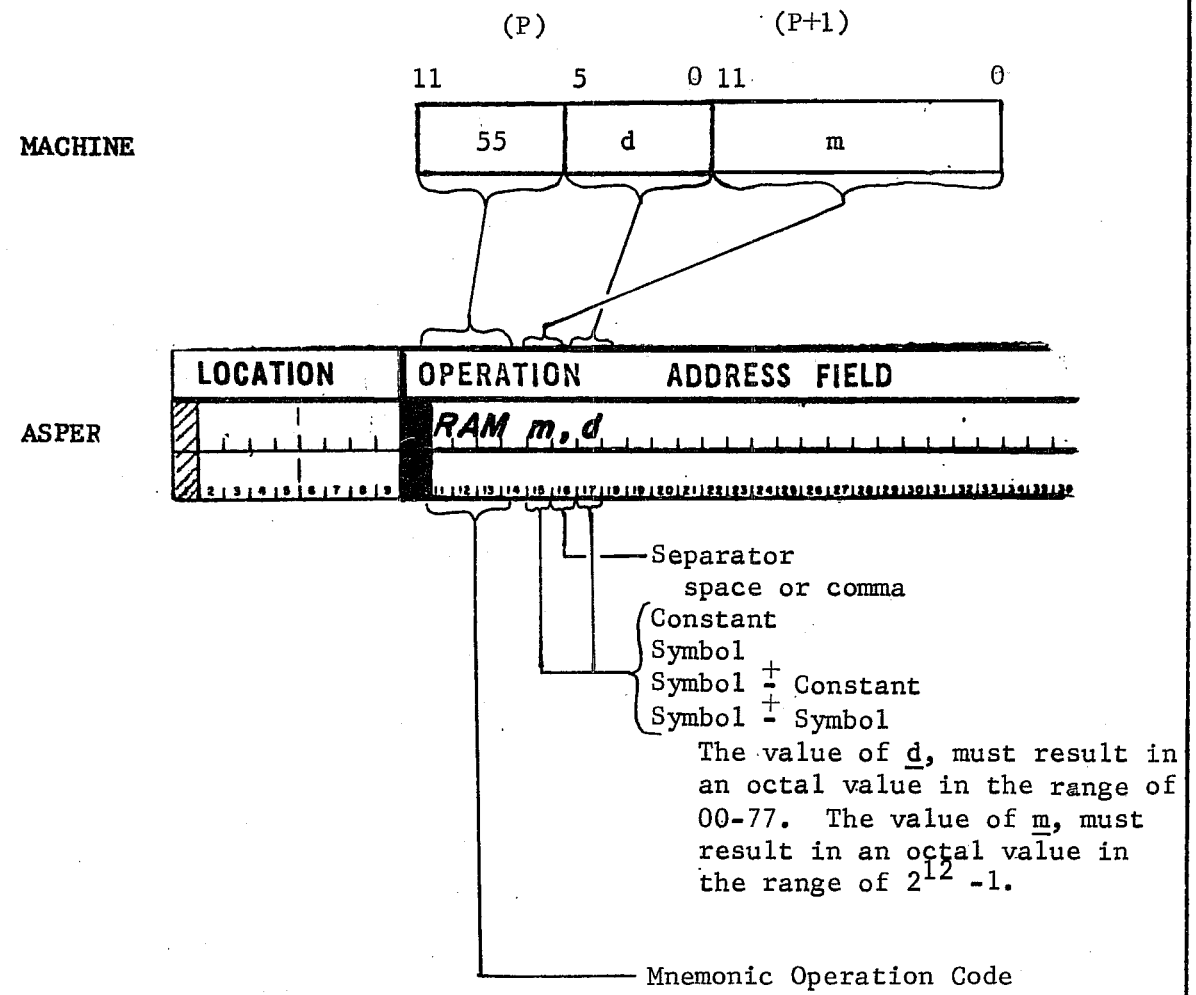
Exercise #2 - Code an instruction which subtracts one from an available table. The 1st reference is to TABLØC, and the 2nd reference to AVAILTAB.

ANSWER



FORMATS

REPLACE ADD MEMORY INDEX



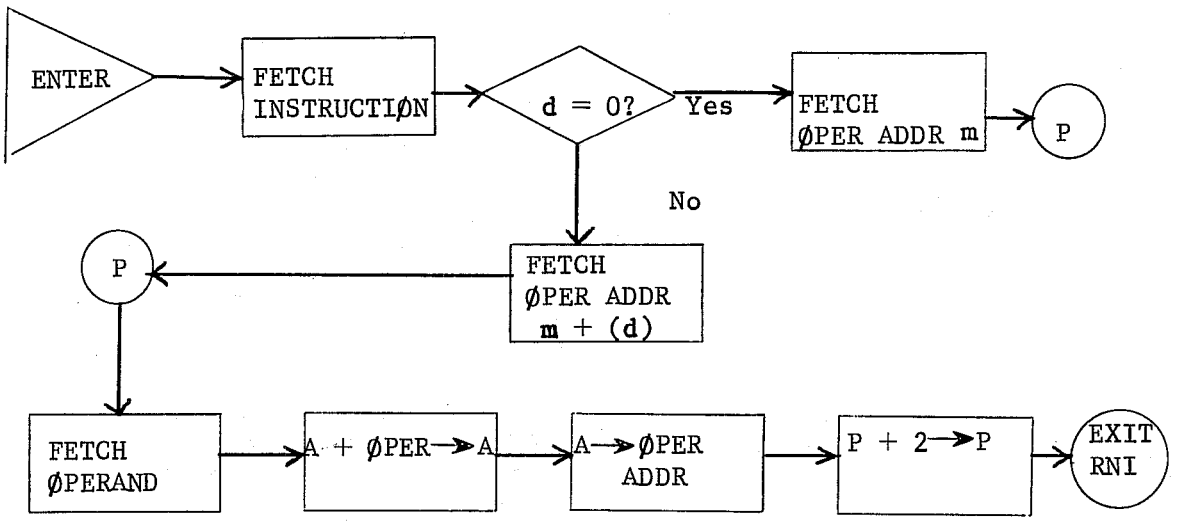
DESCRIPTION

This instruction adds the contents of the A-Register to the operand obtained from indexed addressing. The resultant sum is left in the A-Register at the end of the operation and the lower 12-bits of A replace the original operand in memory. Note: If $d = 0$, the operand address is simply m. If $d \neq 0$, then m plus the contents of location d, $m + (d)$ is the operand address. Thus the contents of d may be used as an index quantity to modify operand addresses. RNI @ P+2.

REFERENCES :

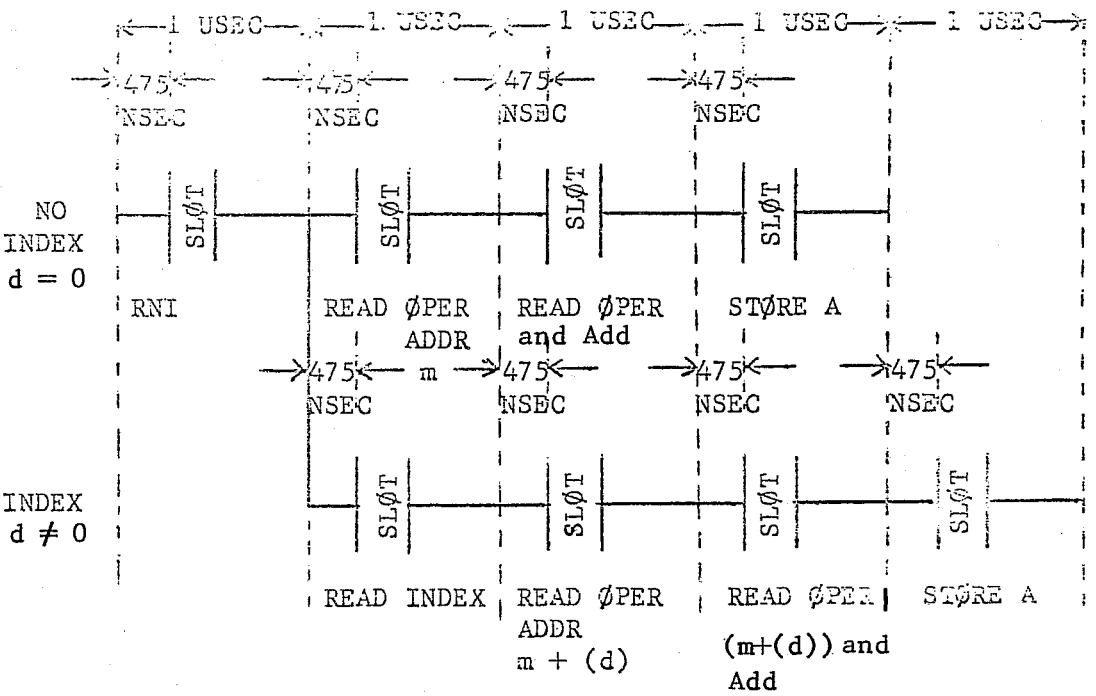
RAM

FLOW DIAGRAM



TIMING

6400/6600 4-5 USEC

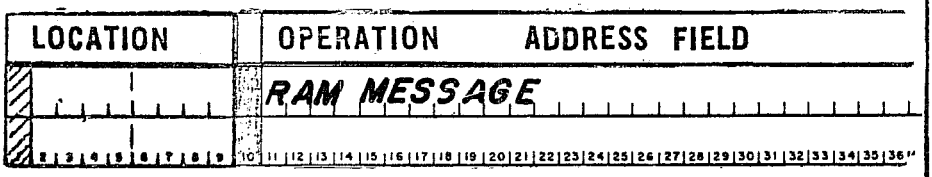


SLØT TIME = 100 NSEC

EXAMPLES

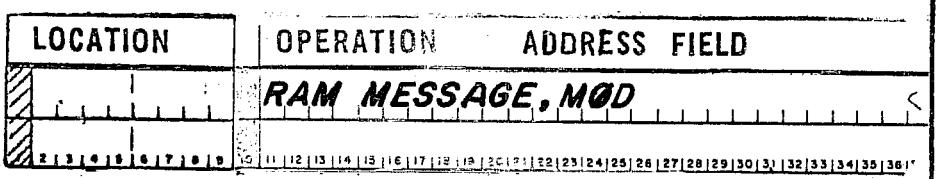
Exercise #1 - Code an instruction to add an operand and the A-Register and replace the result using the memory index instruction with no indexing. Let m = MESSAGE, and d must = 0.

ANSWER



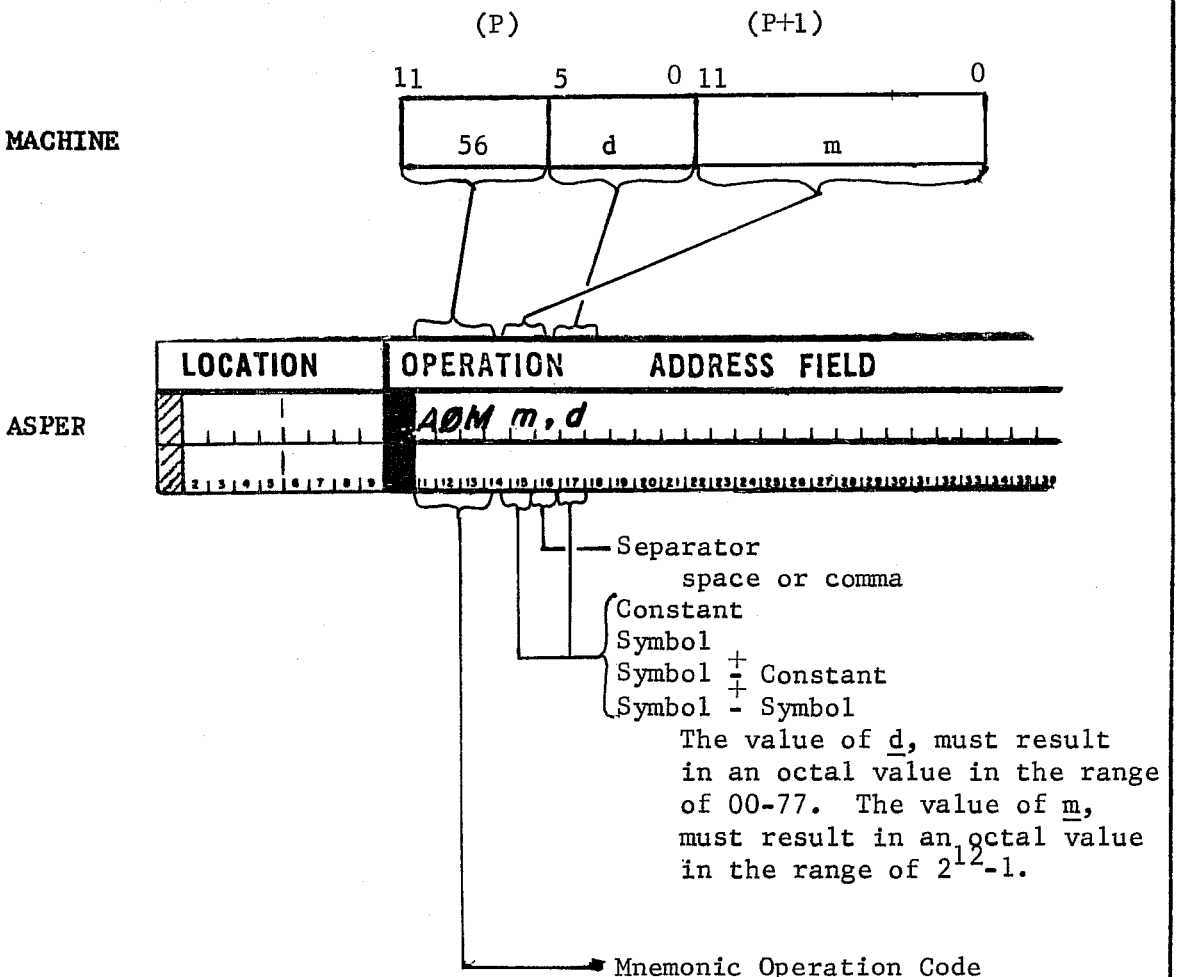
Exercise #2 - Using the same exercise as above, code an instruction using memory index with index modification. Let d = MØD.

ANSWER



FORMATS

REPLACE ADD ØNE MEMORY INDEX



DESCRIPTION

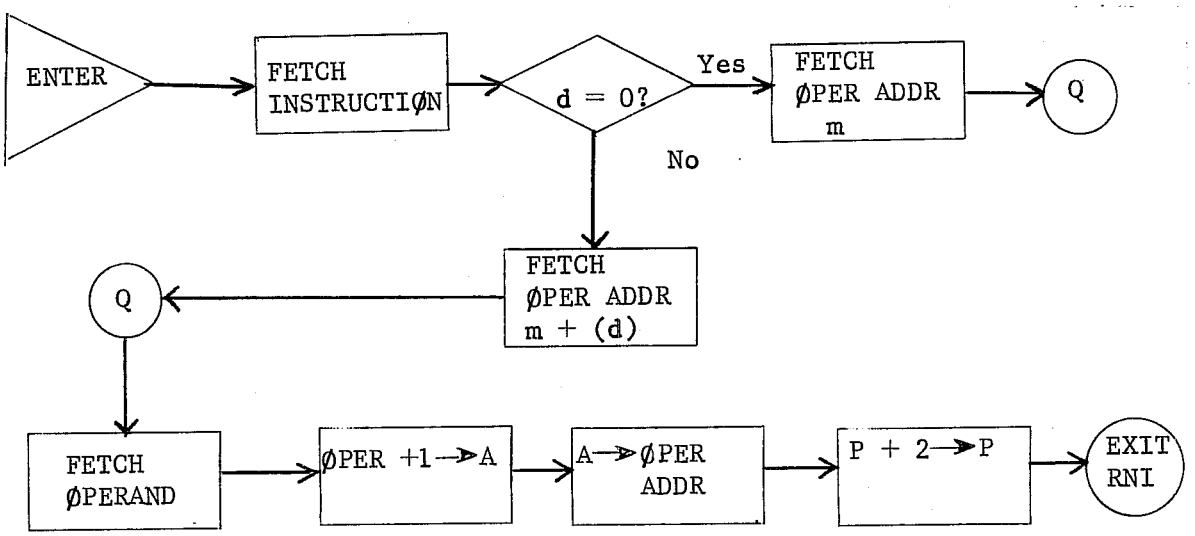
This instruction adds one to the operand obtained from indexed addressing. The resultant sum is left in the A-Register at the end of the operation and the lower 12-bits of A replace the original operand in memory. The original contents of A are destroyed.

Note: If $d = 0$, the operand address is simply m.
If $d \neq 0$, then m plus the contents of location d, m + (d) is the operand address. Thus the contents of d may be used as an index quantity to modify operand addresses. RNI @ P+2.

REFERENCES :

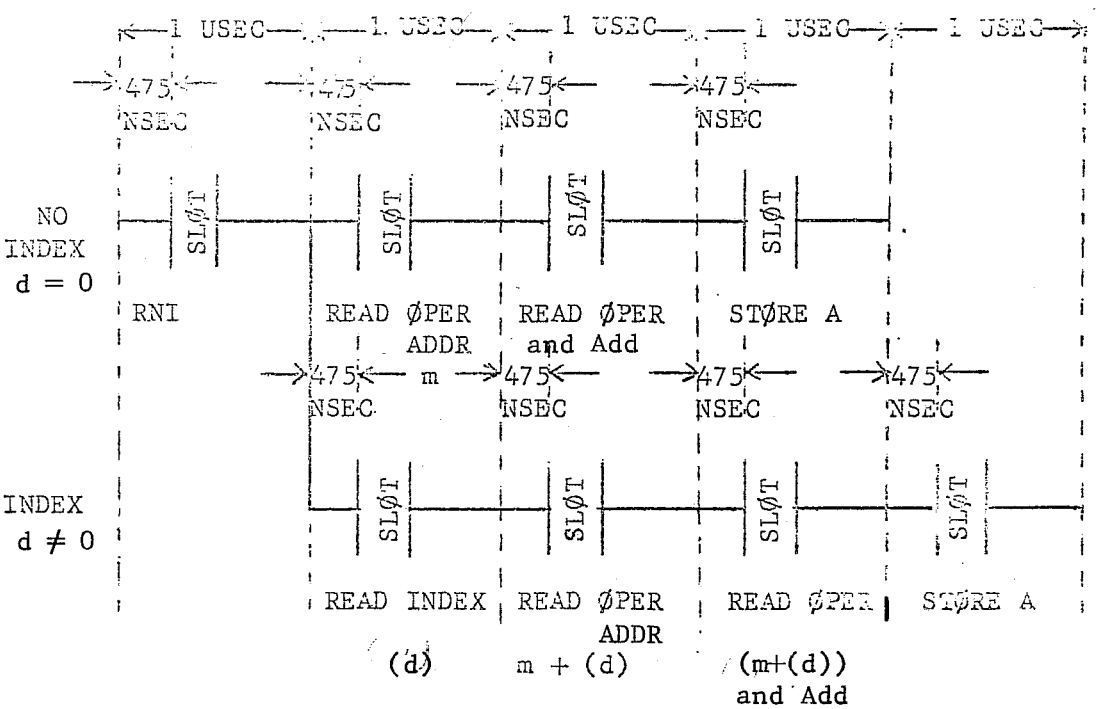
AØM

FLOW DIAGRAM



TIMING

6400/6600 4-5 USEC

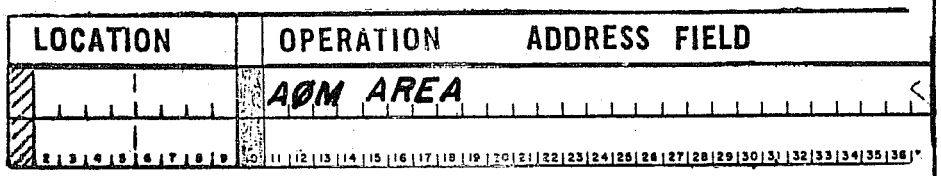


SLØT TIME = 100 NSEC

EXAMPLES

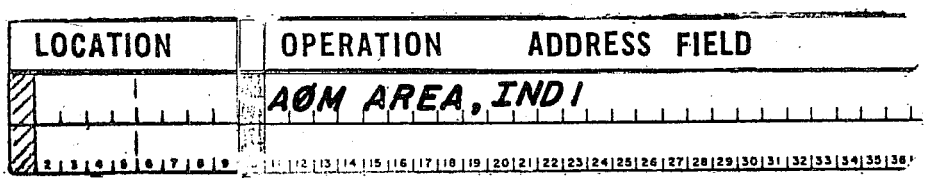
Exerçise #1 - Code an instruction to update an operand by one. Use memory index with no index modificatiǒn. Let m = AREA.

ANSWER



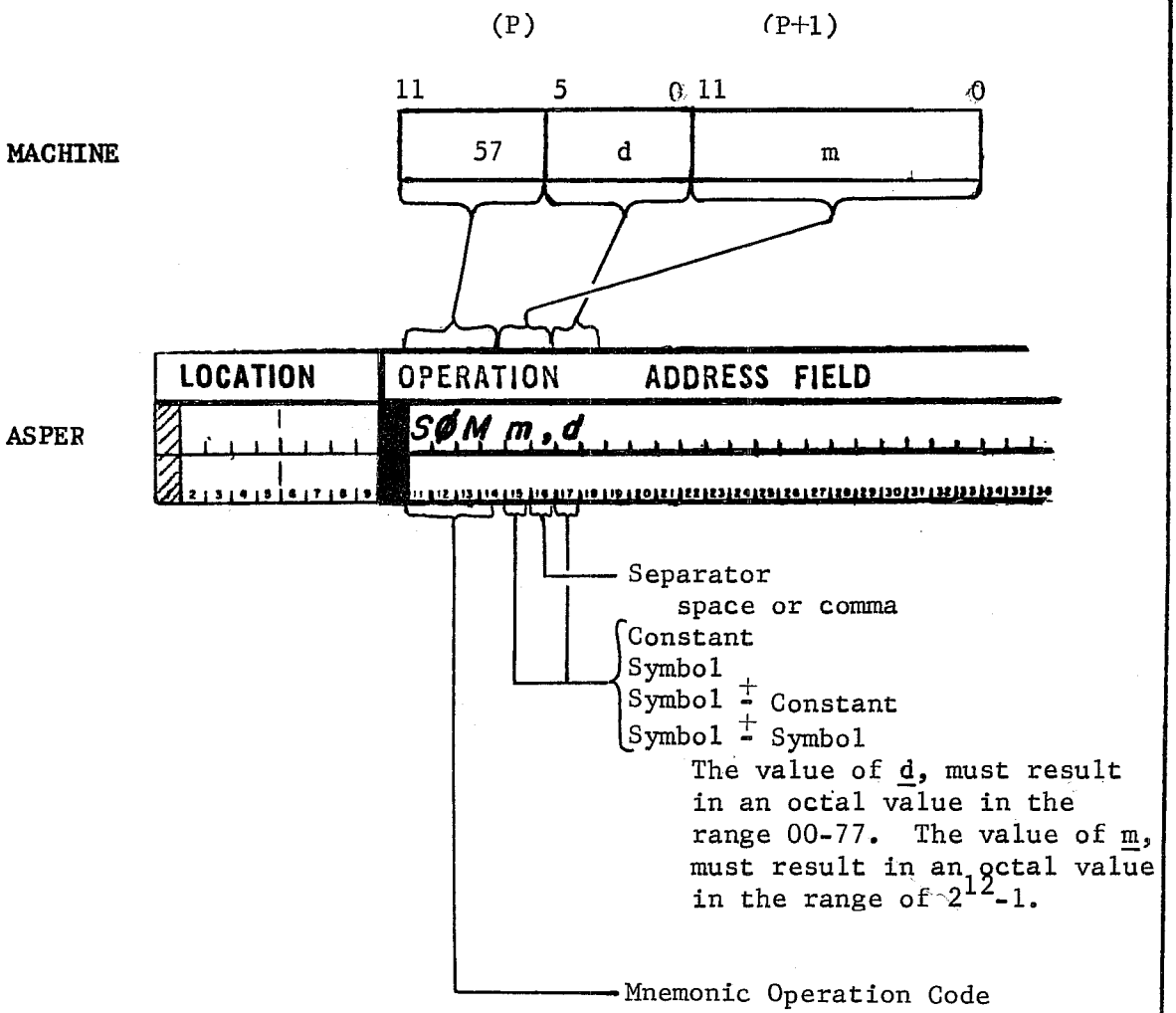
Exercise #2 - Using the example above,code the instruction using the index modification, and let d = IND1.

ANSWER



FORMATS

REPLACE SUBTRACT ØNE
MEMØRY INDEX



DESCRIPTION

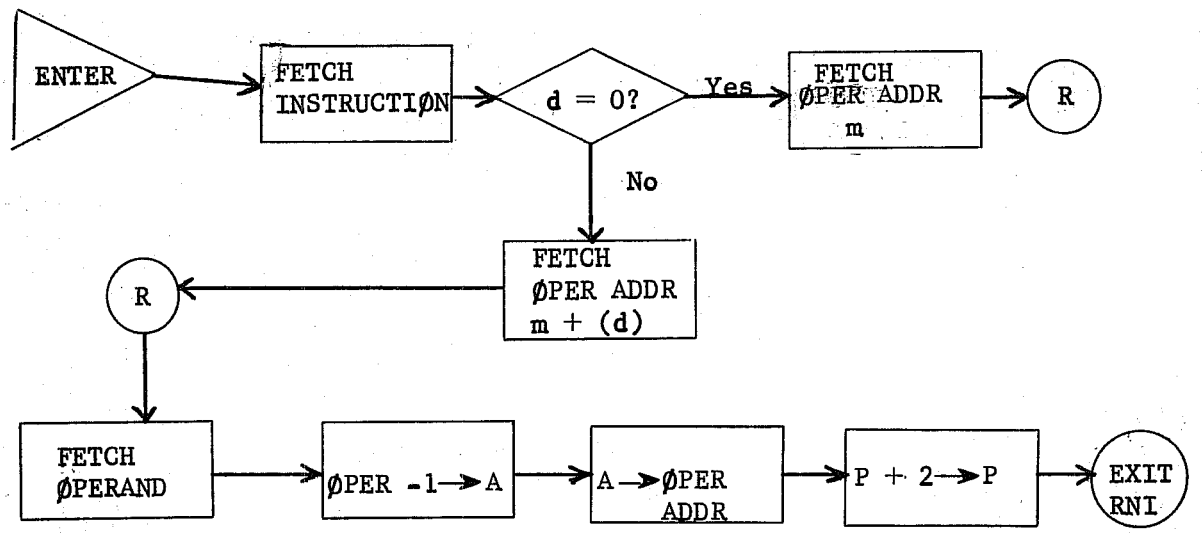
This instruction subtracts one from the operand obtained from indexed addressing. The resultant difference is left in the A-Register at the end of the operation and the lower 12-bits of A replace the original operand in memory. The original contents of A are destroyed.

Note: If $d = 0$, the operand address is simply m.
If $d \neq 0$, then m plus the contents of location d, $m + (d)$ is the operand address. Thus the contents of d may be used as an index quantity to modify operand addresses. RNI @ P+2.

REFERENCES :

SØM

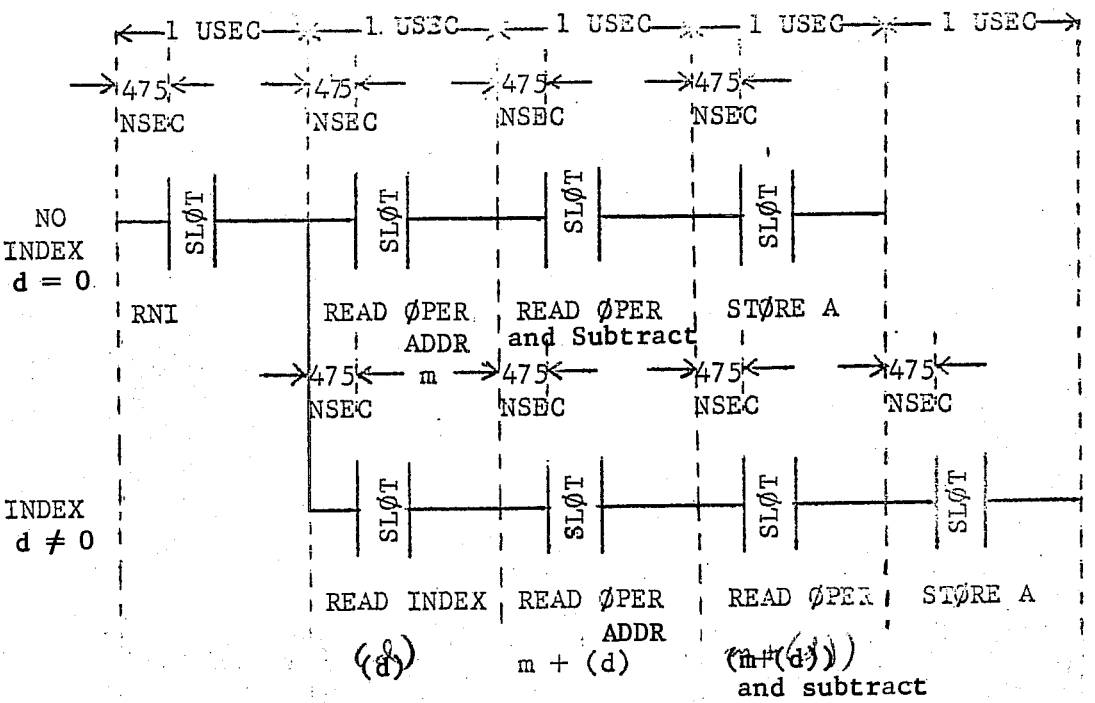
FLOW DIAGRAM



TIMING

6400/6600

4-5 USEC

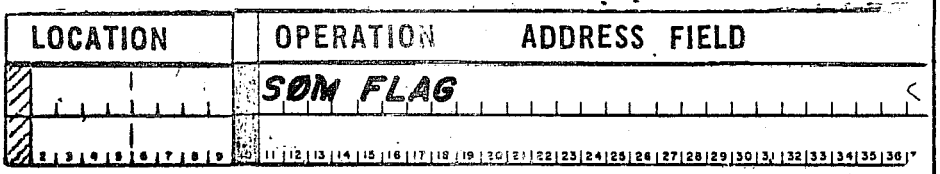


SLØT TIME = 100 NSEC

EXAMPLES

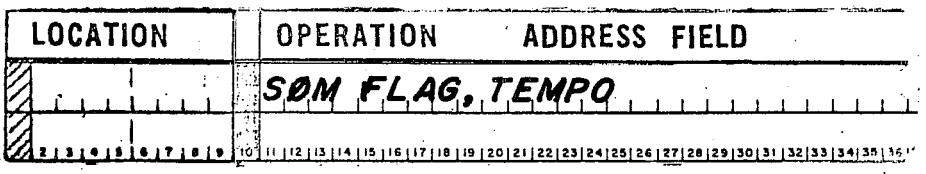
Exercise #1 - Code an instruction to decrement an operand by one, using memory index with no index modification. Let m = FLAG.

ANSWER



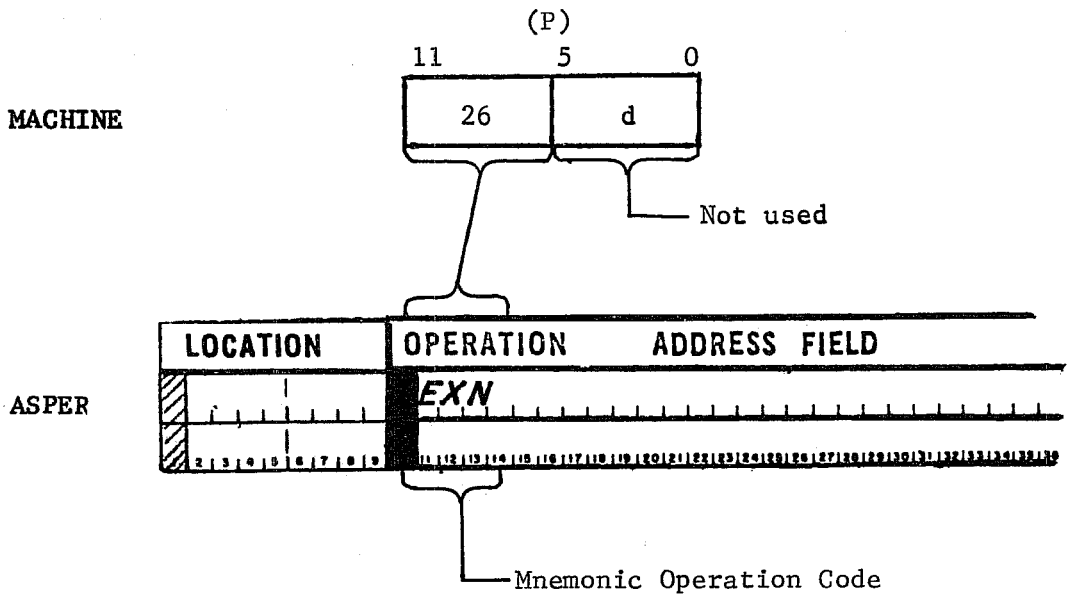
Exercise #2 - Using the above exercise code the instruction using indexing. Let d = TEMPO.

ANSWER



FORMATS

EXCHANGE JUMP



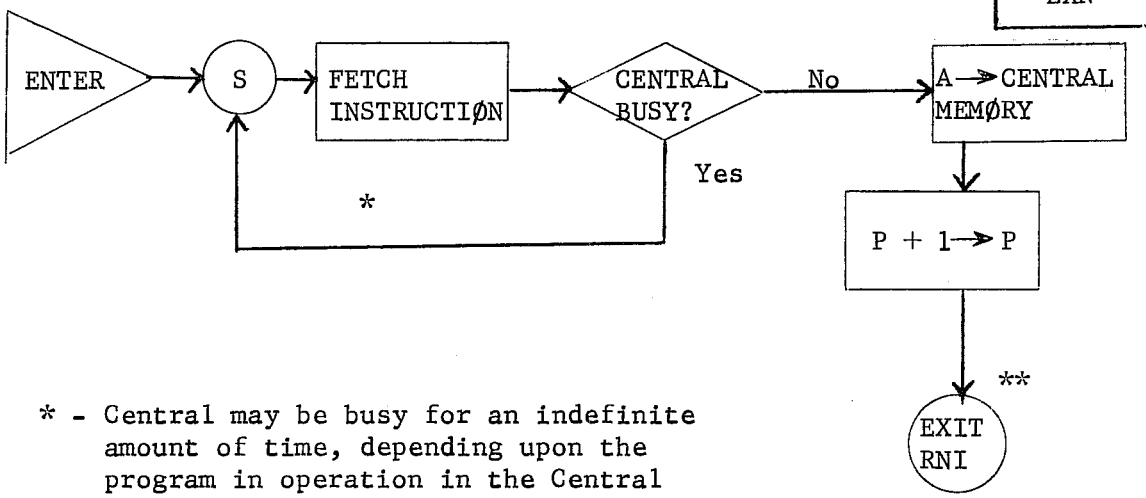
DESCRIPTION

This instruction transmits an 18-bit address from the A-Register to the central processor with a signal which tells the central processor to perform an exchange jump, with the address in A as the starting location of a file of 16 words containing information about the CP Program to be executed. The 18 bit initial address must be entered in A before this instruction is executed. The central processor replaces the file with similar information from the interrupted CP Program. The PP Program is not interrupted. RNI @ P+1.

REFERENCES :

FLOW DIAGRAM

EXN



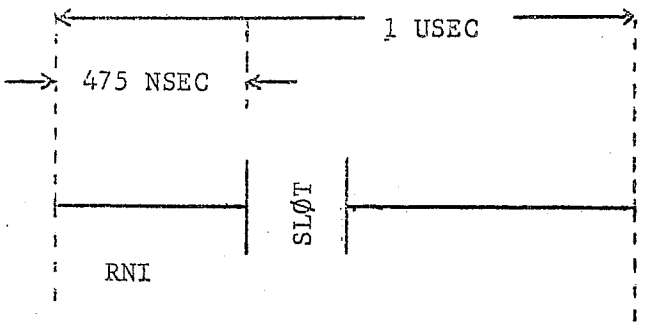
* - Central may be busy for an indefinite amount of time, depending upon the program in operation in the Central Processor.

** - When the PP sets the Central Busy condition, it will stay set for a minimum of 2 usec. However, the PP is free to continue with the next instruction.

TIMING

6400/6600

1 USEC

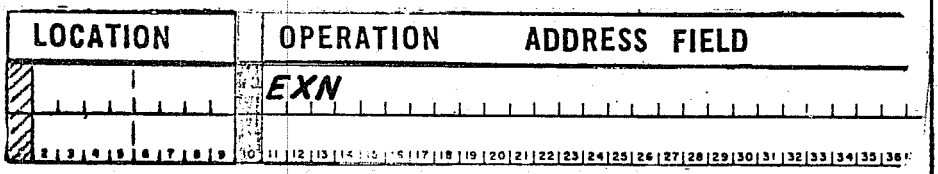


SLØT TIME = 100 NSEC

Exercise #1 - Code an instruction to initiate the CP on a new program.

Note: The A-Register must already contain the 18-bit central memory address.

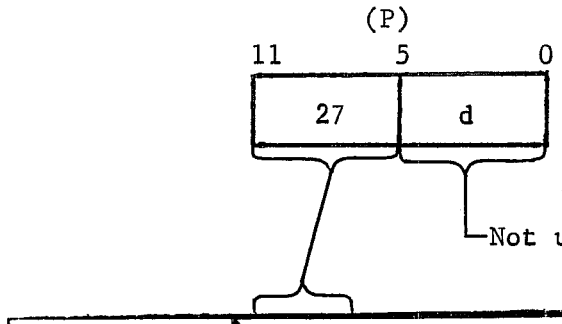
ANSWER



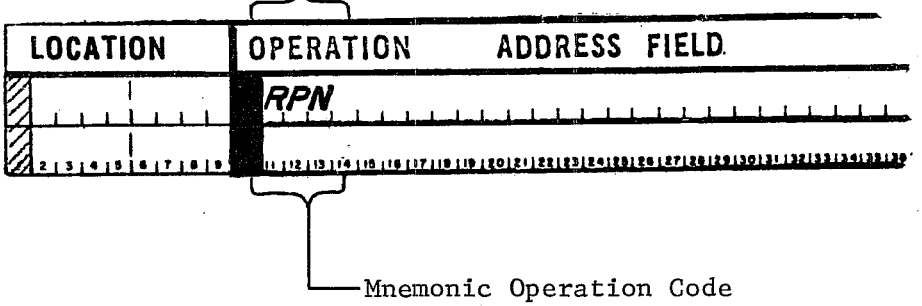
READ PRØGRAM ADDRESS

FORMATS

MACHINE



ASPER



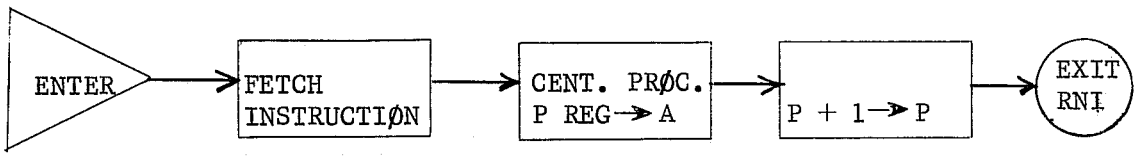
DESCRIPTION

This instruction transfers the contents of the central processor program address register to the peripheral processor A-Register to allow the PP to determine whether the central processor is running.

REFERENCES :

RPN

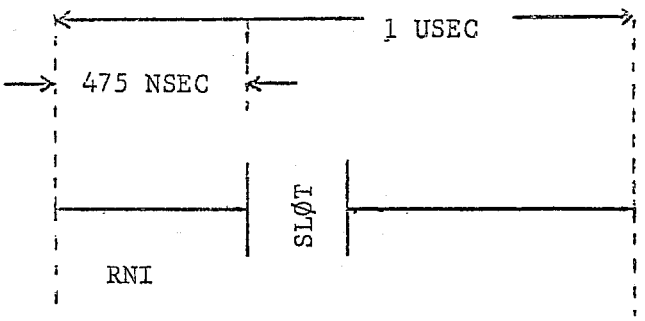
FLOW DIAGRAM



TIMING

6400/6600

1 USEC

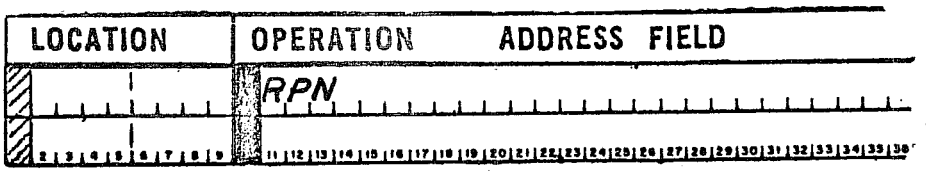


SLØT TIME = 100 NSEC

EXAMPLES

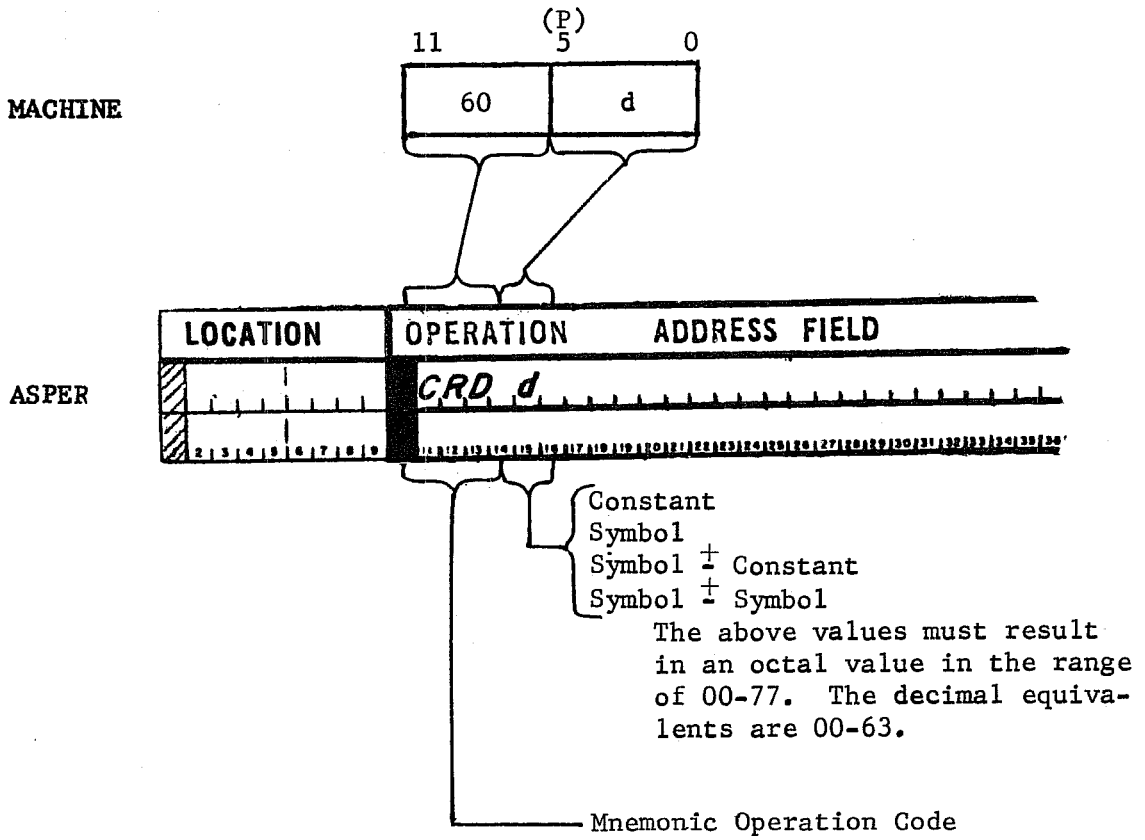
Exercise #1 - Code the instruction to read the contents of the CP P-Register (Program Address Counter)

ANSWER



FORMATS

CENTRAL READ FRØM A

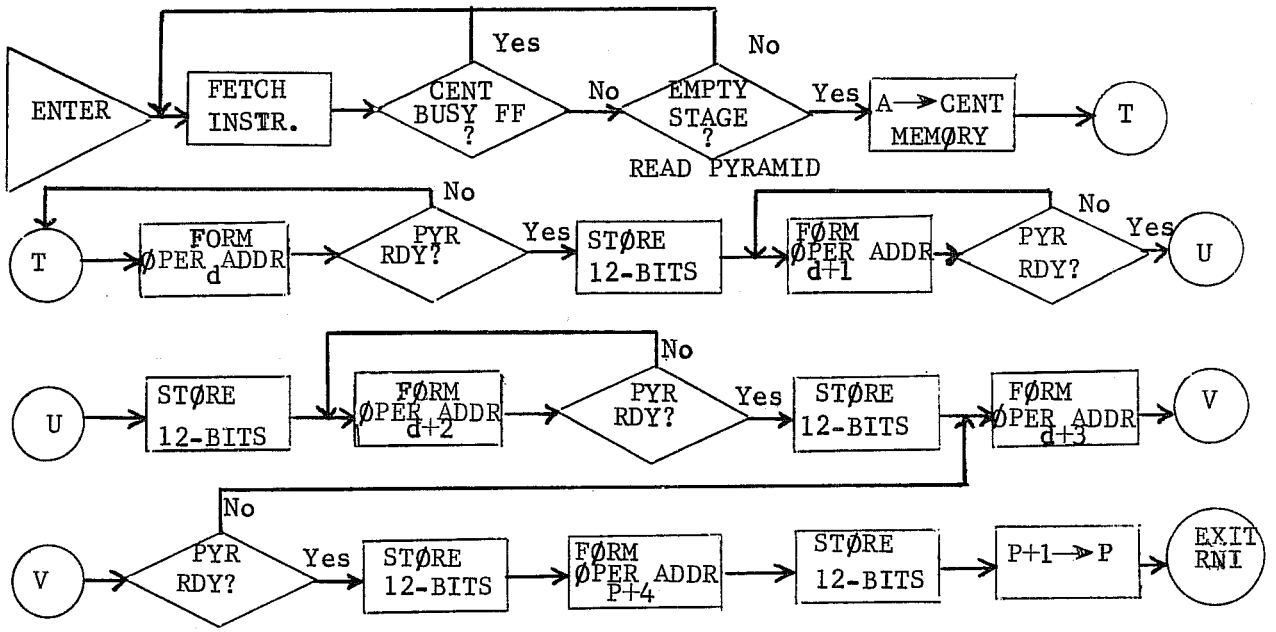


DESCRIPTION

This instruction transfers a 60-bit central memory word to 5 consecutive PP memory locations. The A-Register must contain the 18-bit CM address before the instruction is executed. The 60-bit CM word is disassembled beginning at the left with the location specified by d receiving the left most 12-bit word; d +1, the next 12-bit word, and so on.

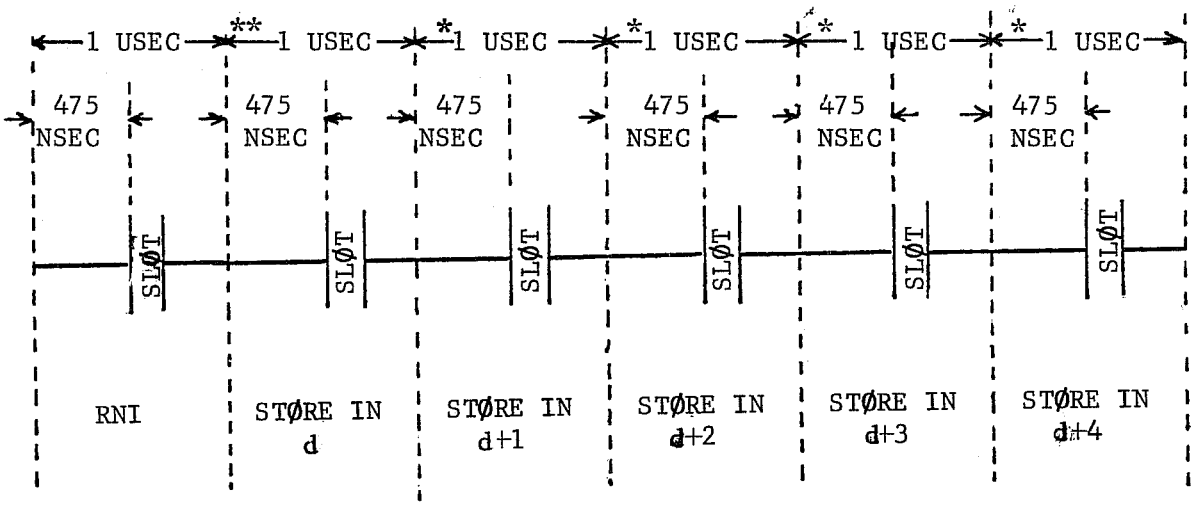
RNI @ P+1.

REFERENCES :



6400/6600

MINIMUM 6 usec



** This cycle will loop until Cent. Busy FF is not being used by another PP, and until there is an empty stage in the Read Pyramid.

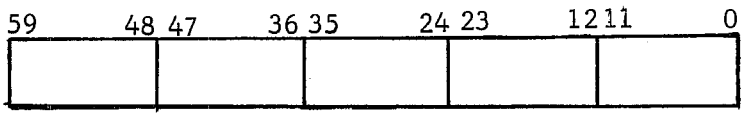
* Each one of these cycles will loop if the next stage in the Read Pyramid is also full, from another processor.

SLØT = 100 nsec.

EXAMPLES

Exercise #1 - Code the instruction to transfer a Cent.Memory Word (60-bits) to 5 PP Mem. locations, (12-Bit Words)

Let d = 76



$2^{48 - 59} \rightarrow 0076$

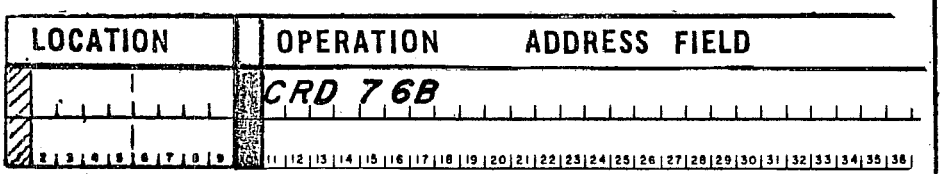
$2^{36 - 47} \rightarrow 0077$

$2^{24 - 35} \rightarrow 0100$

$2^{12 - 23} \rightarrow 0101$

$2^{0 = 11} \rightarrow 0102$

ANSWER

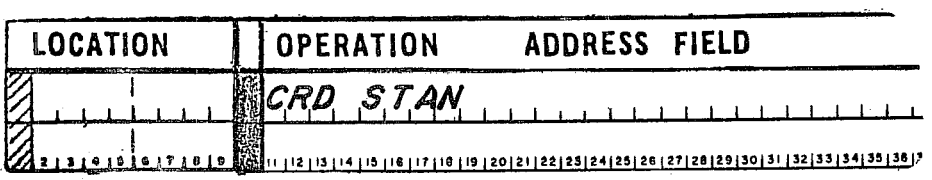


Note: A initially contains the 18-bit Cent. Memory Address

Exercise #2 - Code the instruction to transfer a Cent. Memory Word starting at PP location STAN.

STAN may = any value $00\text{-}77_8$.

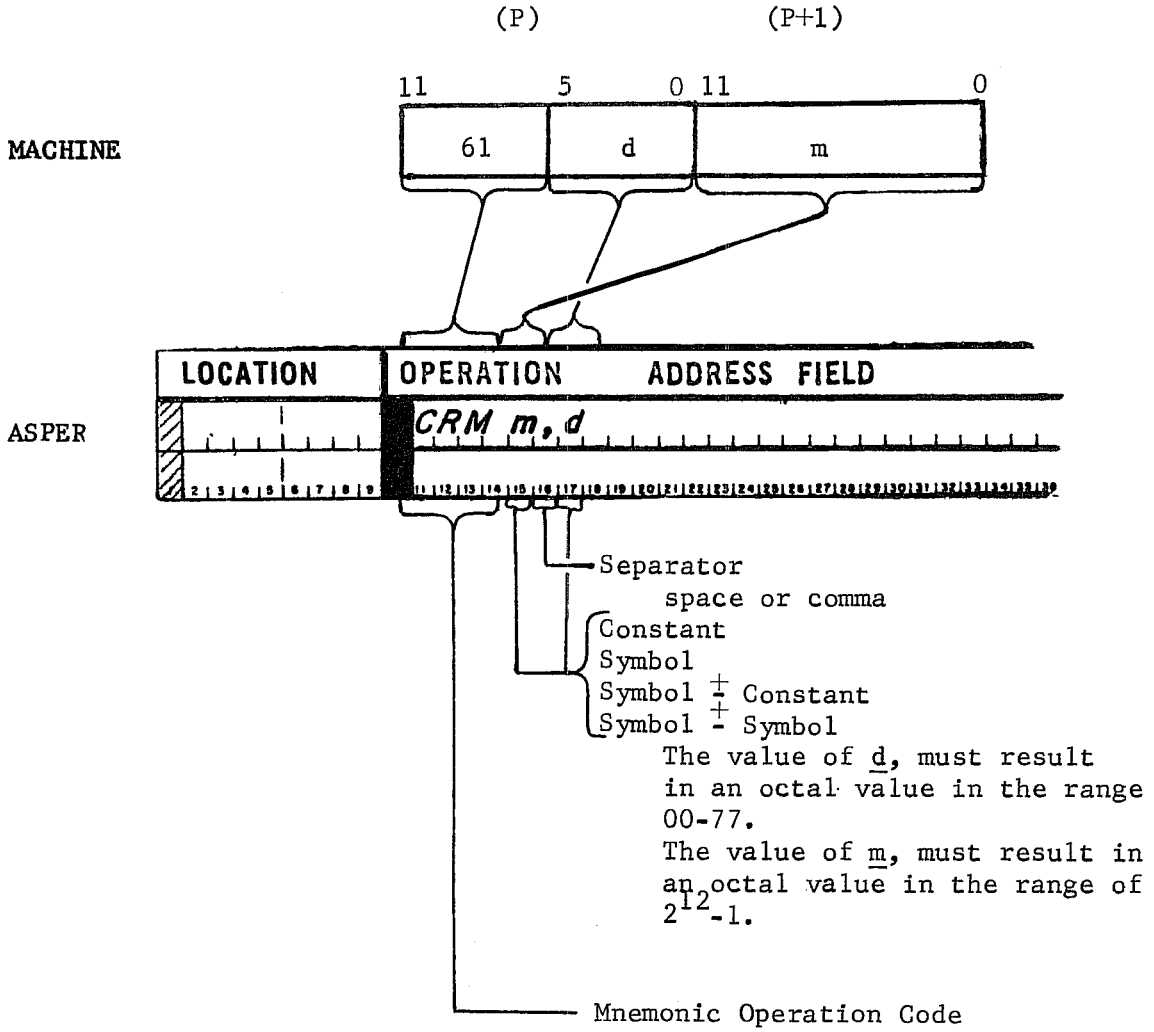
ANSWER



Note: See above note

CENTRAL READ BLØCK

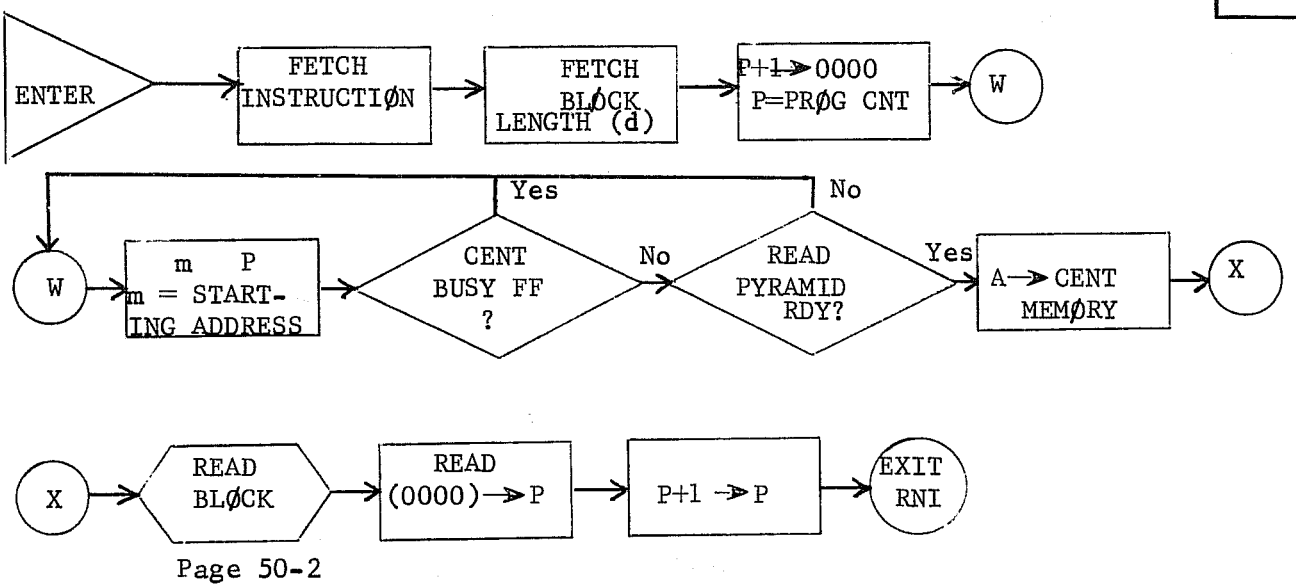
FORMATS



DESCRIPTION

This instruction reads a block of 60-bit words from central memory into PP memory. The A-Register contains the 18-bit CM starting address and must be loaded prior to the execution of this instruction. The contents of A are increased by one as each 60-bit CM word is disassembled and stored. During execution of this instruction, the original contents of the P-Register are stored in PP location 0000 and the address of the first word to be stored into PP memory, m, goes to the P-Register. P is updated for each new address. The original contents of P are restored upon completion. RNI @ P+2. The block length or number of CM words to be read is contained in location d, CM addresses = (A) + (d)-1. The block length also goes to the Q Register where it is reduced by one as each CM word is processed. The transfer is completed when Q=0. The PP memory addresses = m+5(d)-1, where m is the starting address in the PP.

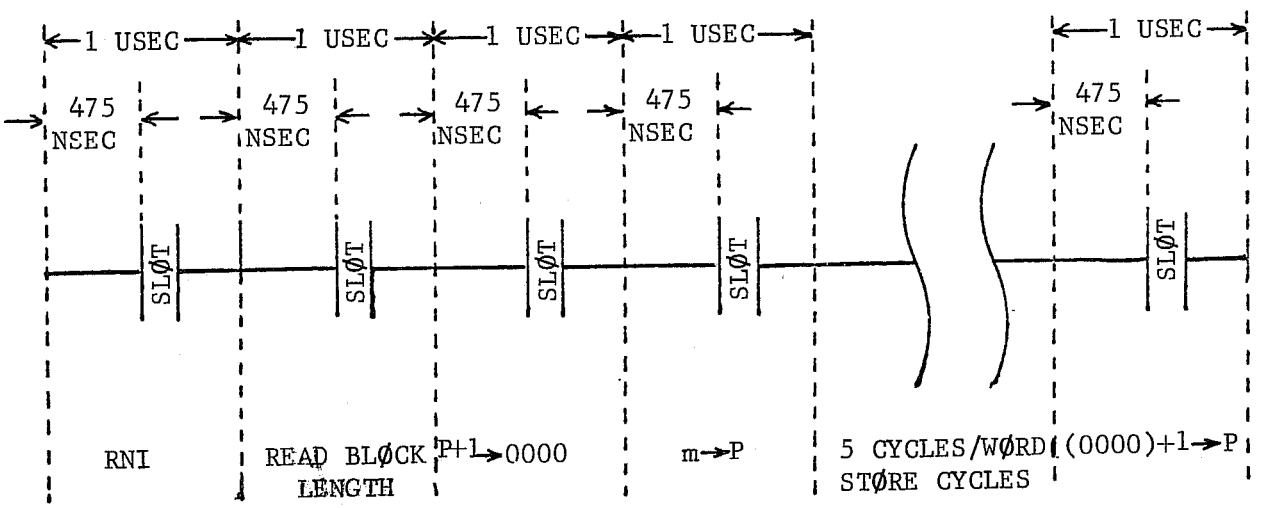
REFERENCES :



Page 50-2

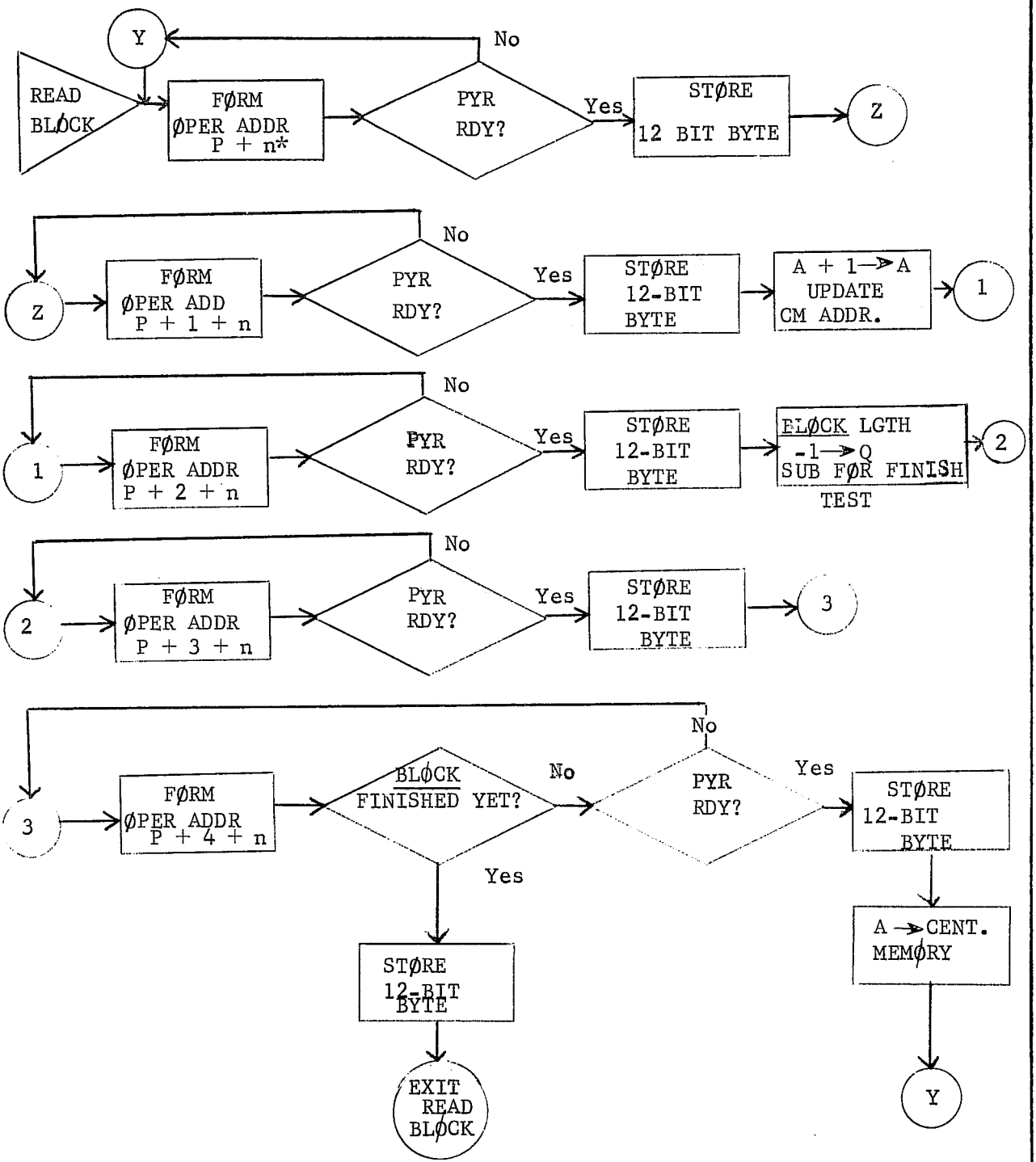
6400/6600

MIN. 5-PLUS 5/WORD usec



SLØT - 100 nsec

FLOW DIAGRAM



* n = 0, on 1st 60-bit word, 5 on 2nd 60 bit word, etc.

EXAMPLES

Exercise #1 - Code an instruction to transfer a block of words from Cent. Memory to PP Memory.

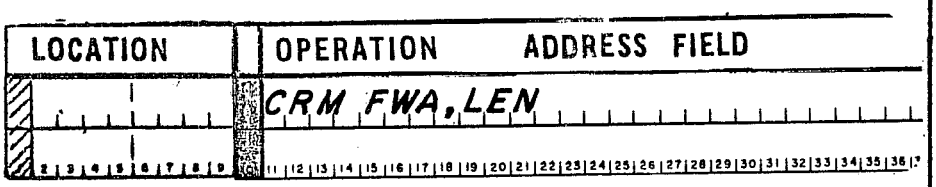
Let d = LEN = $00\text{-}77_8$

(LEN) = # of Words

Let m = FWA, first word address

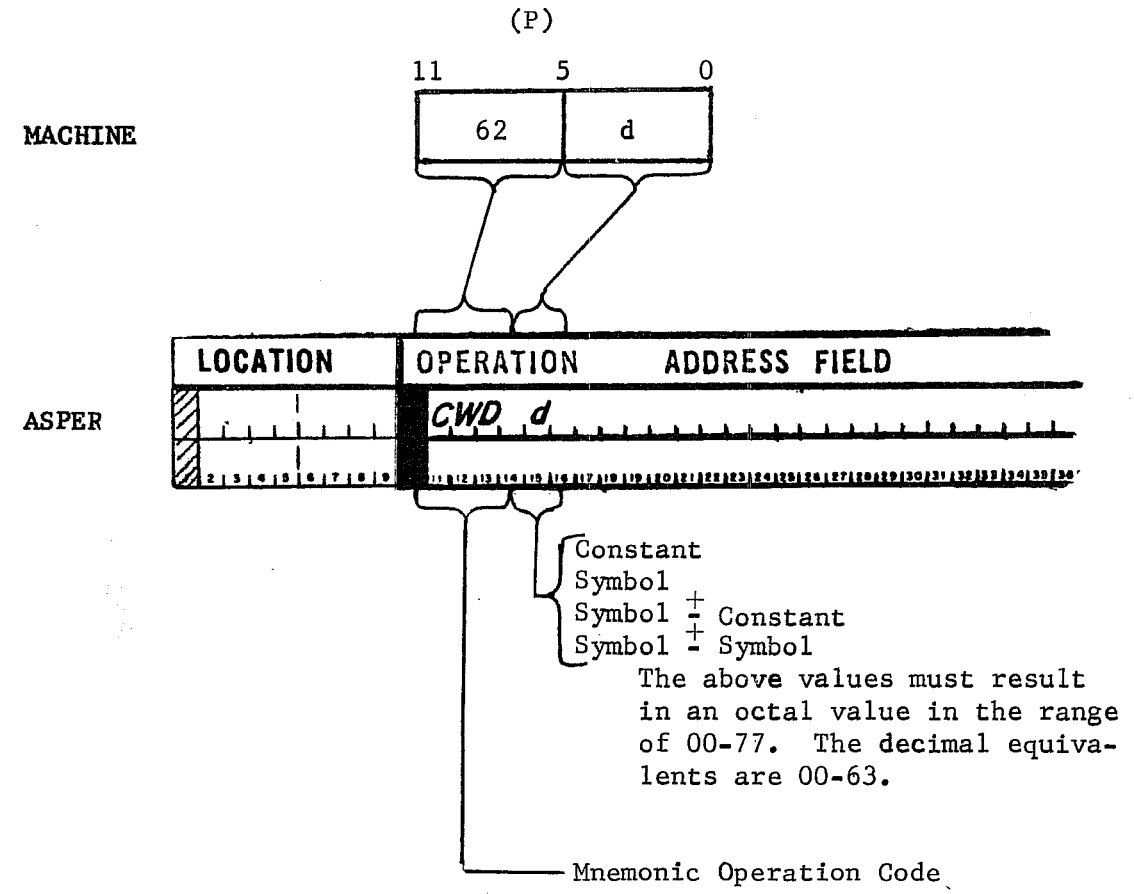
Note: The (A - Register) is the starting address in Cent. Memory.

ANSWER



FORMATS

CENTRAL WRITE TØ A

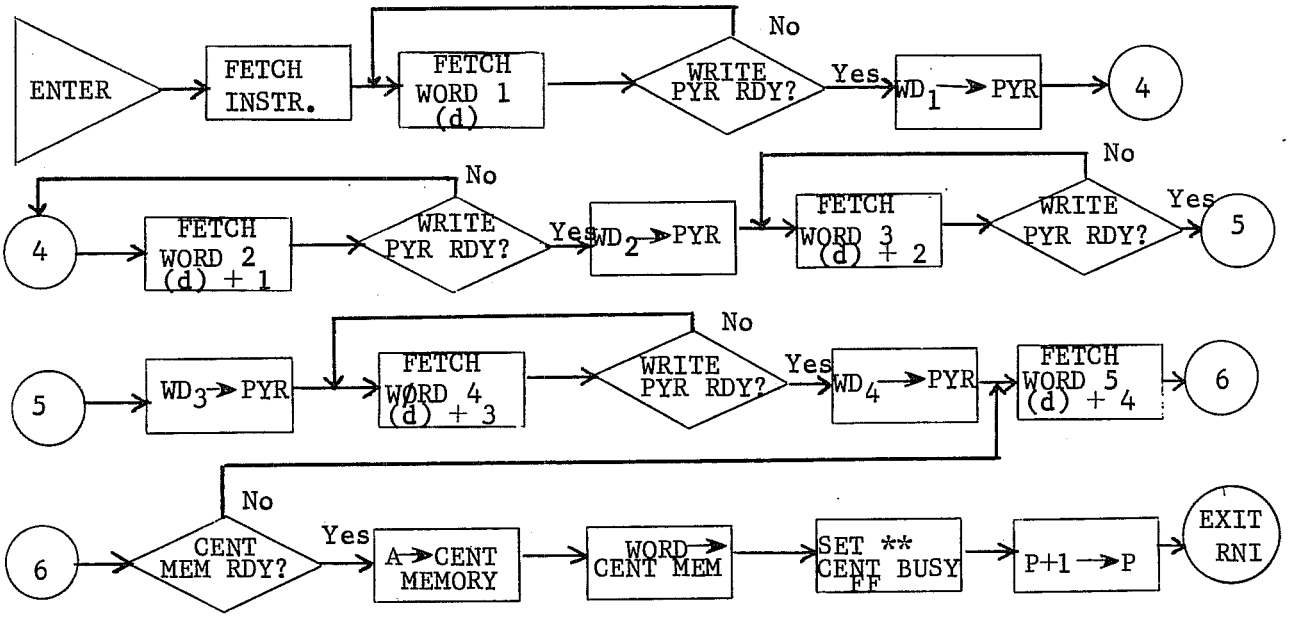


DESCRIPTION

This instruction assembles five successive 12-bit words into a 60-bit word and stores the word in central memory. The 18-bit CM address must be in the A-Register prior to the execution of the instruction. The first word to be read out of PP memory is contained in location d. This word appears as the leftmost 12-bits of the 60-bit word. The remaining 12-bit groups are taken from successive addresses in PP memory. RNI @ P+1.

REFERENCES :

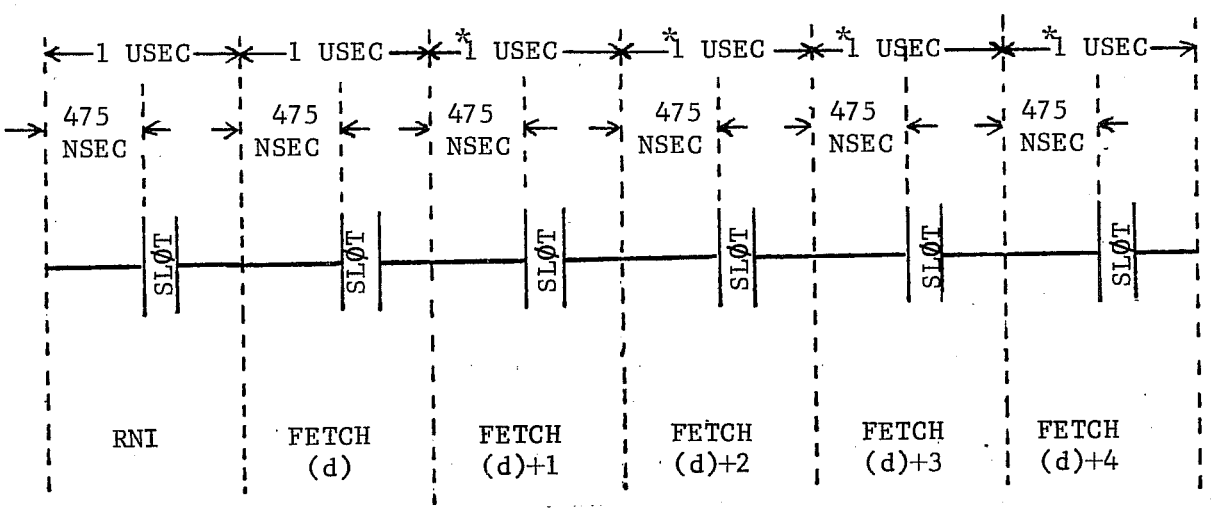
FLOW DIAGRAM



** Central will be busy minimum 600-700 nsec.

TIMING

6400/6600 MINIMUM 6 usec



* Each one of these cycles will loop if the next stage in the write pyramid is also full, from another processor.

SLØT = 100 nsec

EXAMPLES

Exercise #1 - Code an instruction to transfer 5 PP words to Cent. Memory.

Let d = FWAWRITE = 75_8

(0075) → 2^{48} - 59

(0076) → 2^{36} - 47

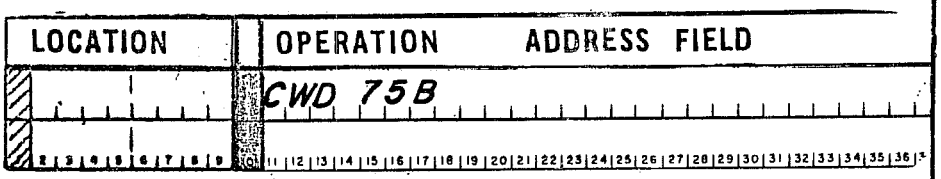
(0077) → 2^{24} - 35

(0100) → 2^{12} - 23

(0101) → 2^{0} - 11

Note: A contains the 18-bit Cent. Memory Address

ANSWER

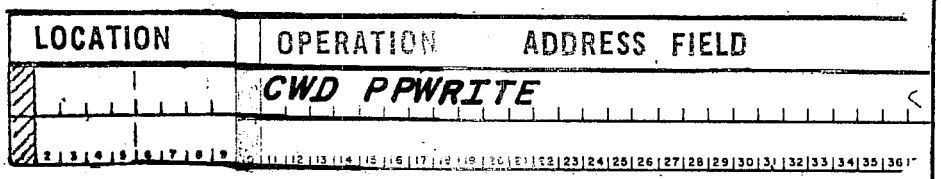


Exercise #2 - Code an instruction to transfer, from PP Mem, a Cent. Mem word to Cent. Mem.

Let d = PPWRITE = a value $00\text{-}77_8$.

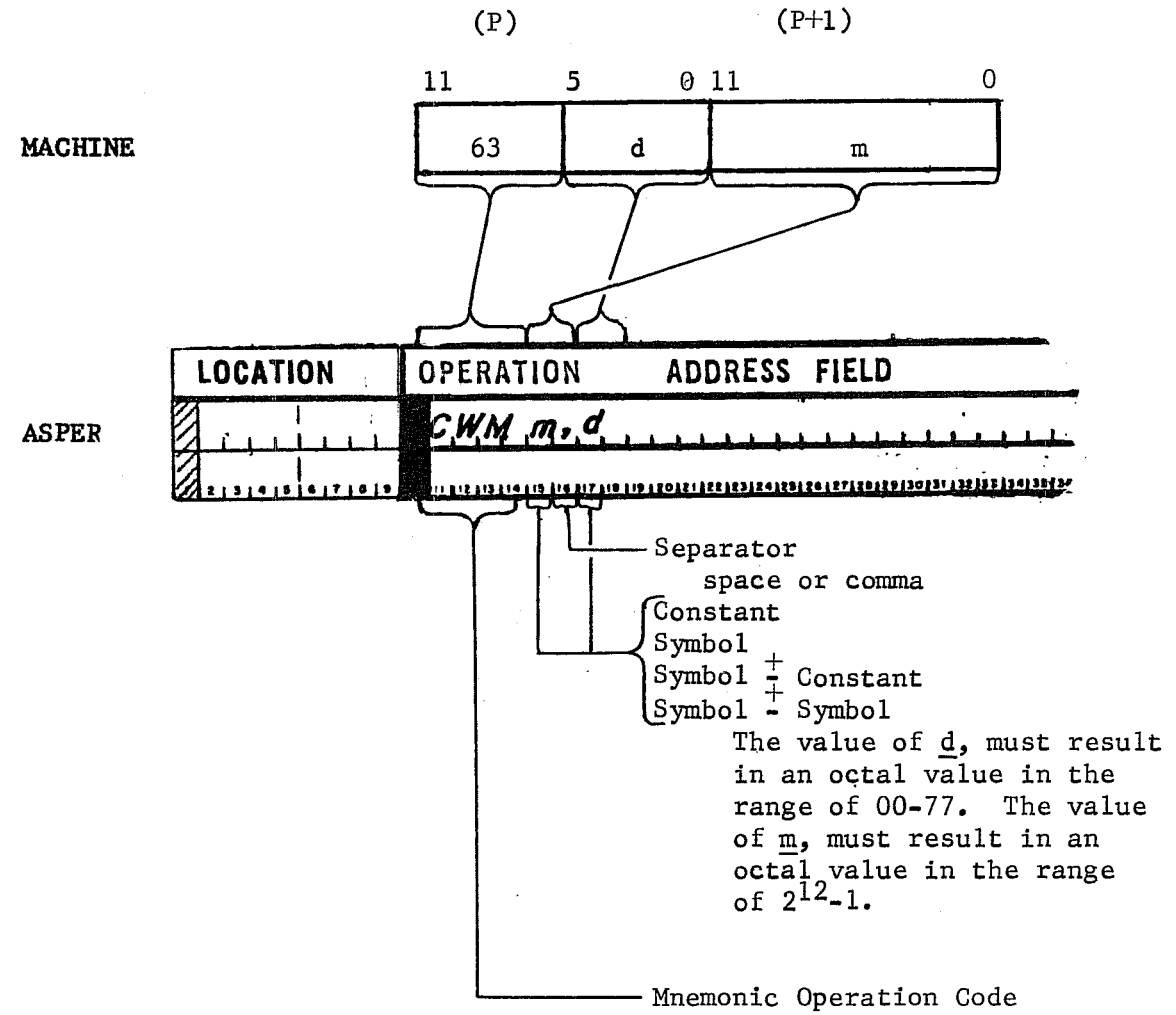
Note: See above note

ANSWER



FORMATS

CENTRAL WRITE BLØCK

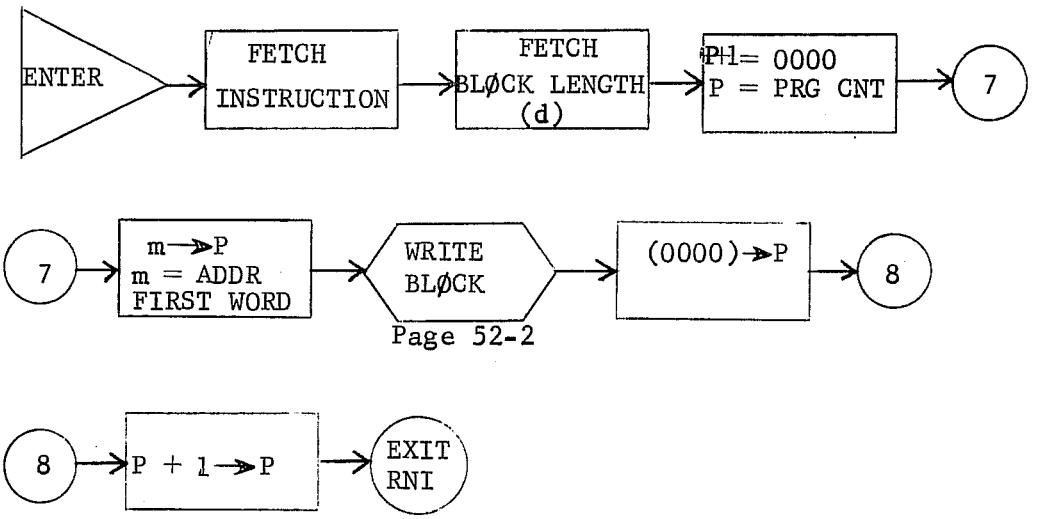


DESCRIPTION

This instruction assembles a block of 60-bit words and writes them in central memory. The A-Register contains the beginning 18 bit CM address and must be loaded prior to the execution of this instruction. Then the address in A is increased by one after each 60-bit word is assembled to provide the next Cm address. The (d)specify the number of 60-bit words to write. This count also goes to the Q-Register where it is reduced by one as each Cm word is assembled. The transfer is completed when Q=0. The Cm Address = (A) + (d)-1. The PP memory addresses = m + 5(d)-1, where m is the starting address in the PP. During execution of this instruction the original contents of the P-Register are are stored in PP location 0000 and the address of the first word to be fetched from PP memory, m, goes to the P-Register. P is updated for each new address. The original contents of P are restored upon completion. RNI @ P+2.

REFERENCES :

FLOW DIAGRAM

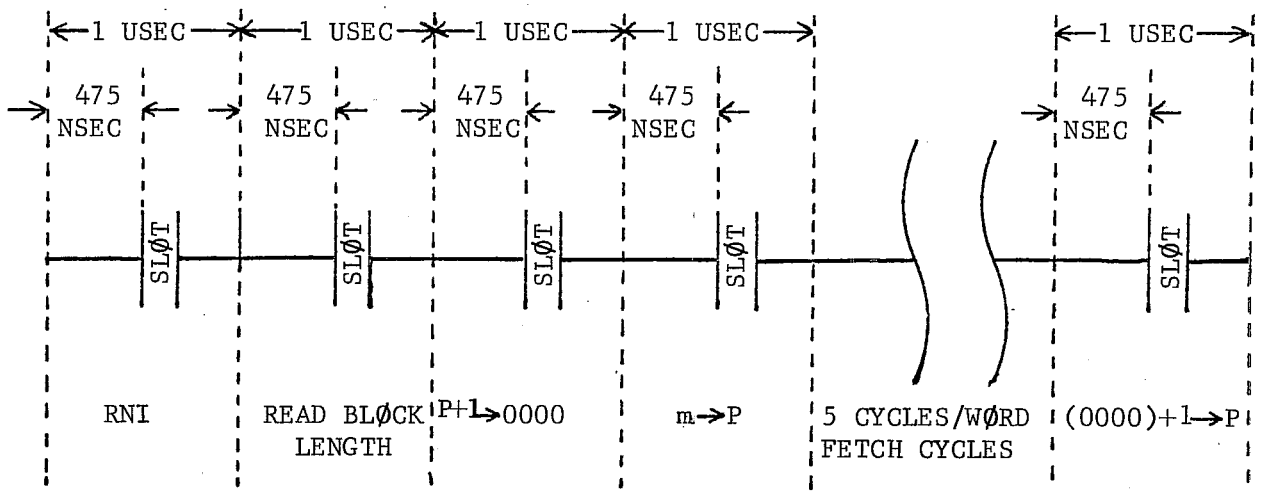


Page 52-2

TIMING

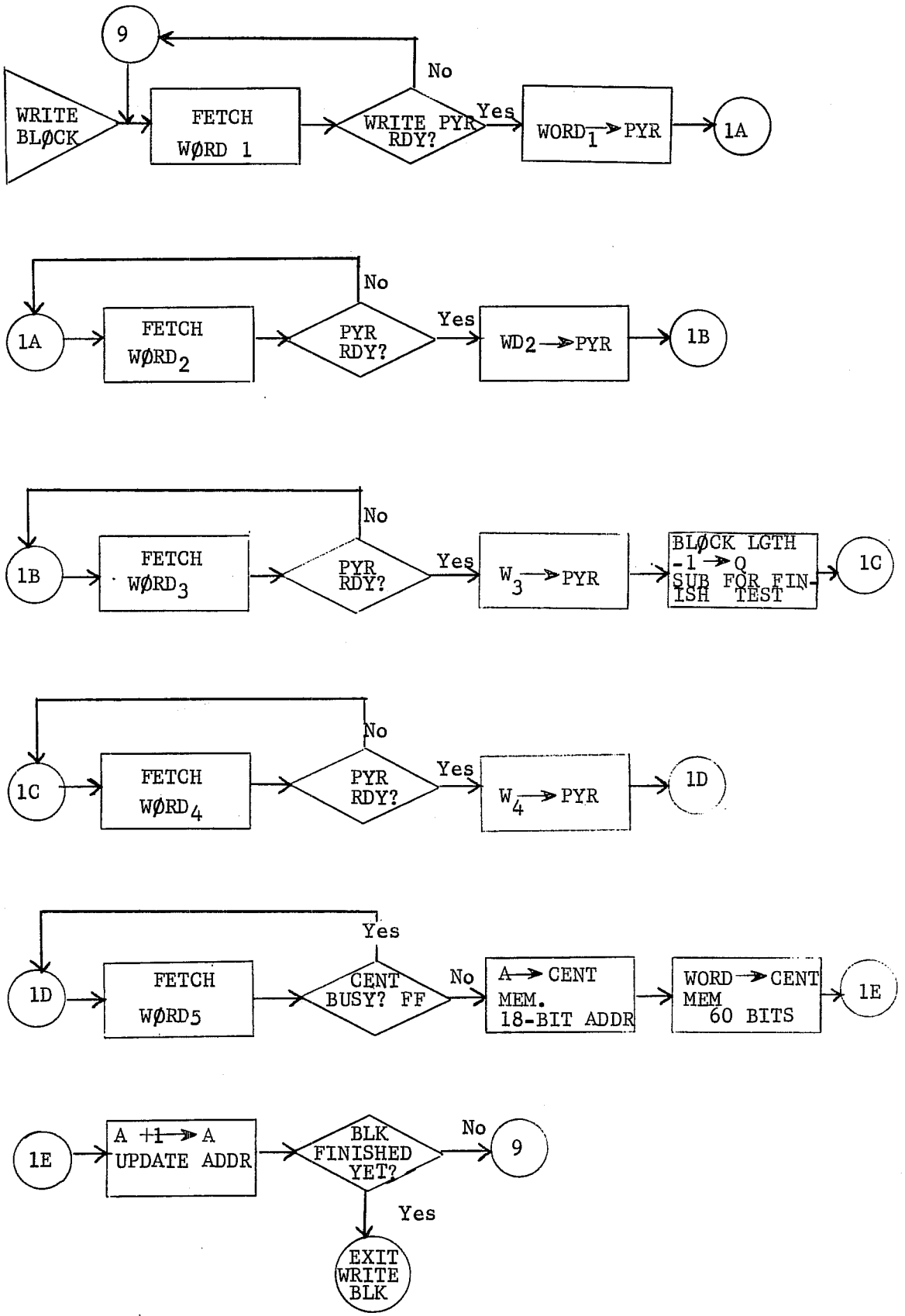
6400/6600

Min. 5-Plus 5/word usec



SLØT = 100 nsec

FLOW DIAGRAM



EXAMPLES

Exercise #1 - Code an instruction to transfer a block of words from PP memory to Central Memory.

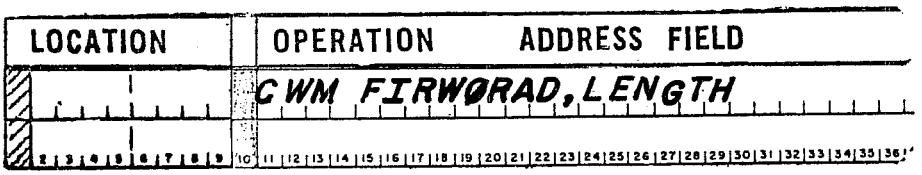
Let d = LENGTH

(LENGTH) = # of words

Let m = FIRWORAD

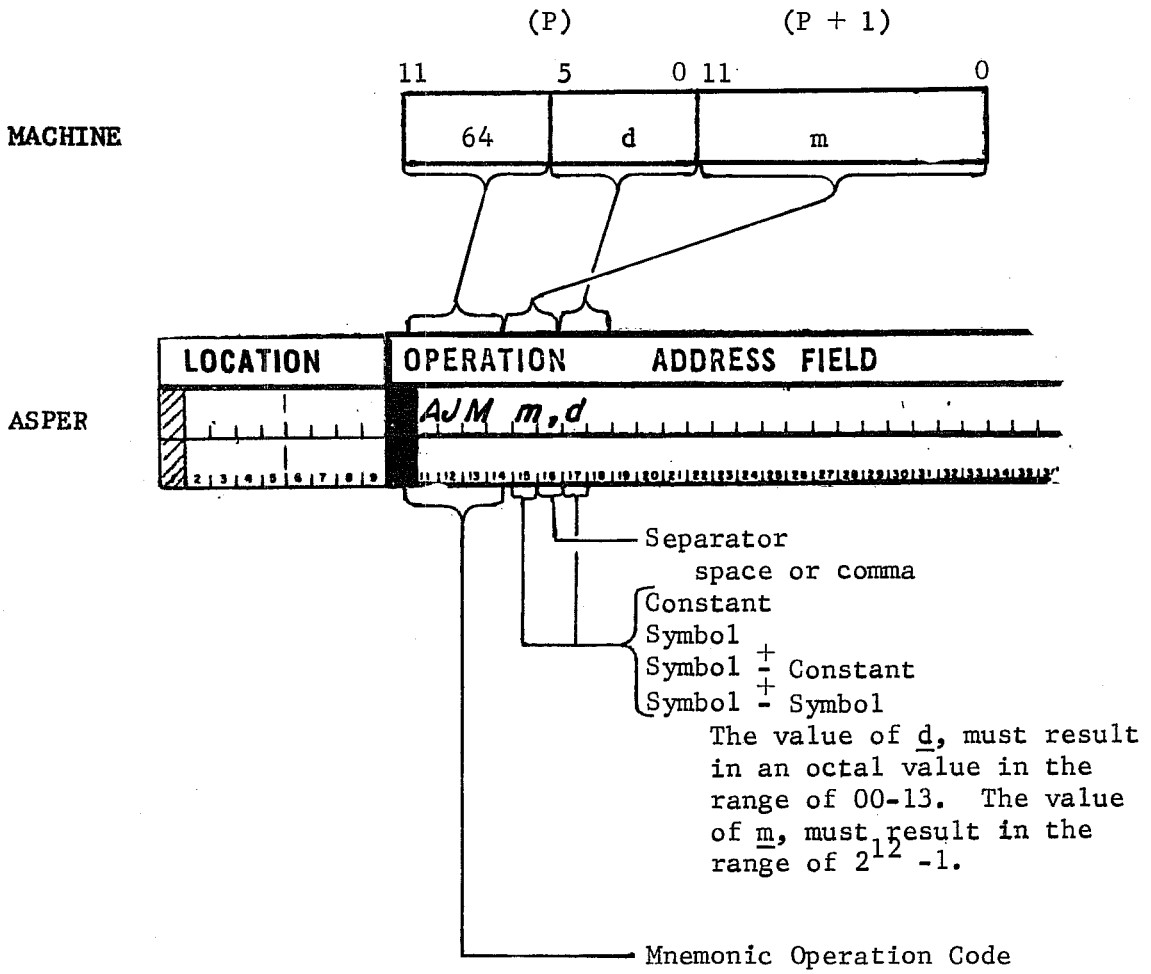
Note: A contains the address of Cent. Mem. that is to receive the first word.

ANSWER



FORMATS

ACTIVE JUMP



DESCRIPTION

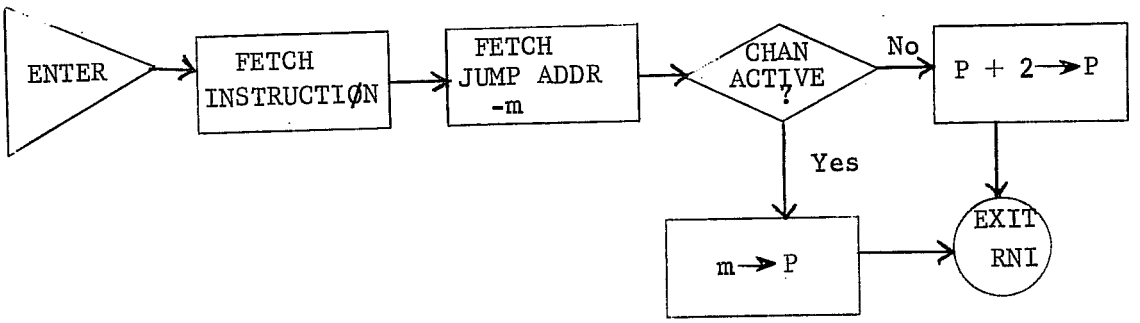
Jump to m if channel d active. This instruction provides a conditional jump to a new program sequence beginning at address m if the channel specified by d is active. If the channel is inactive, the current program sequence continues.

RNI @ P+2

REFERENCES :

AJM

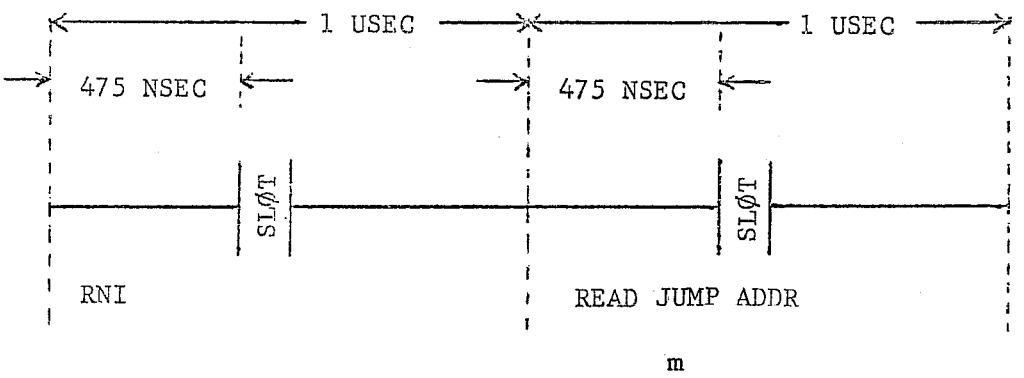
FLOW DIAGRAM



TIMING

6400/6600

2 USEC



SLØT TIME = 100 NSEC

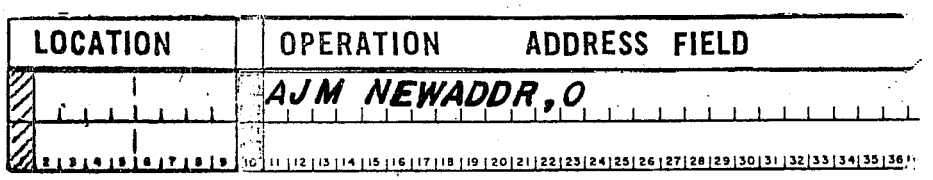
EXAMPLES

Exercise #1 - Code an instruction to test if a channel is active, and if it is jump to a new sequence of instructions.

Let d = 00 (channel #) and m = NEWADDR

Note: RNI @ P+2 if the condition is not met.

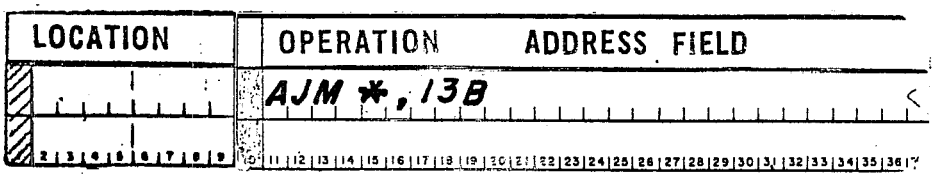
ANSWER



Exercise #2 - Using the exercise above test channel 13 and loop (Wait) until the channel becomes free.

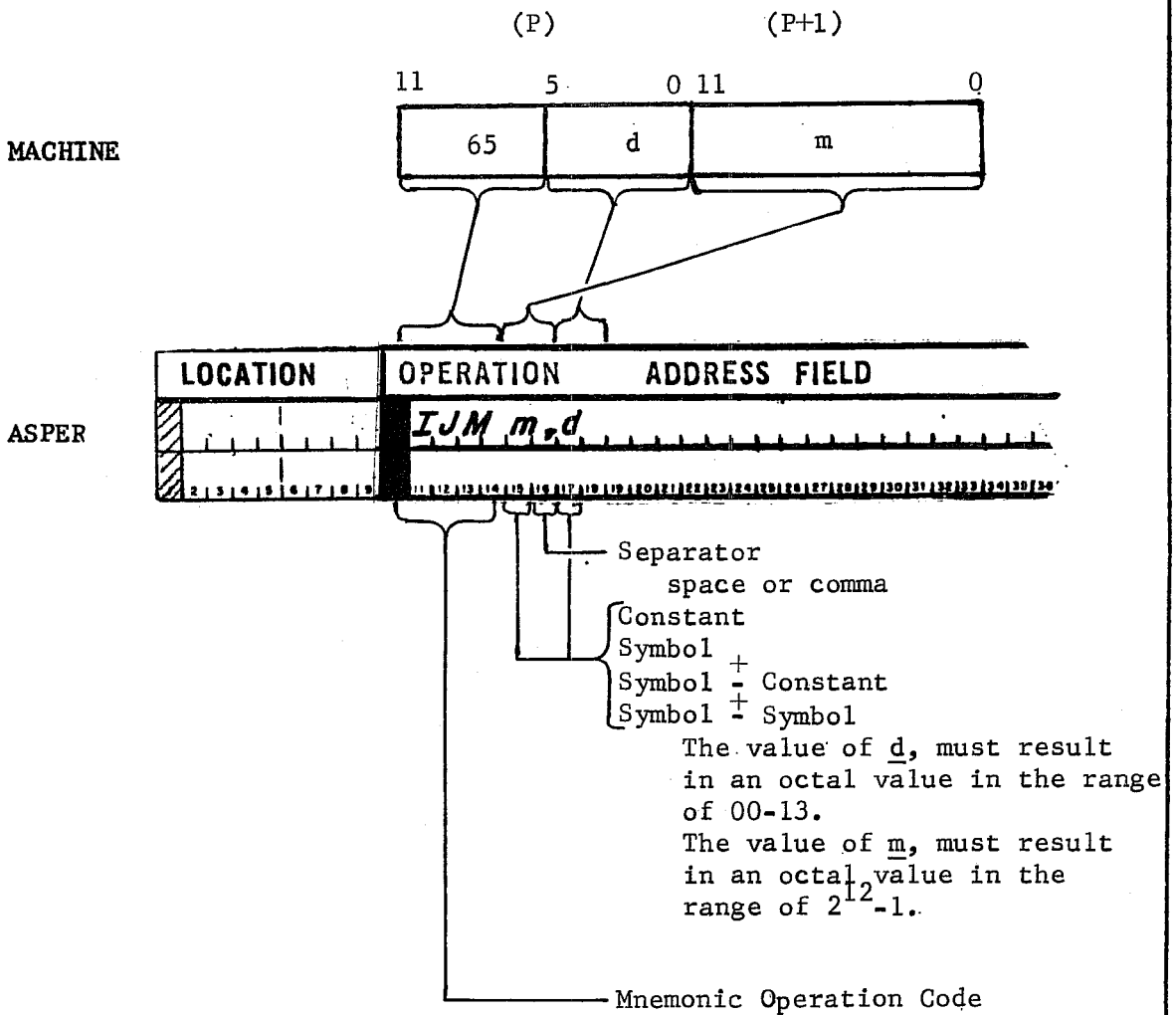
Note: See above note.

ANSWER



FORMATS

INACTIVE JUMP



DESCRIPTION

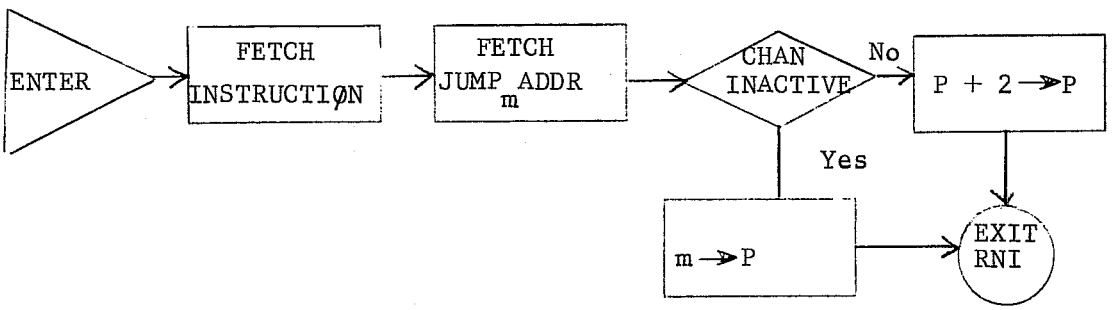
Jump to m if channel d inactive. This instruction provides a conditional jump to a new program sequence beginning at address m if the channel specified by d is inactive. If the channel is active, the current program sequence continues.

RNI @ P+2

REFERENCES :

IJM

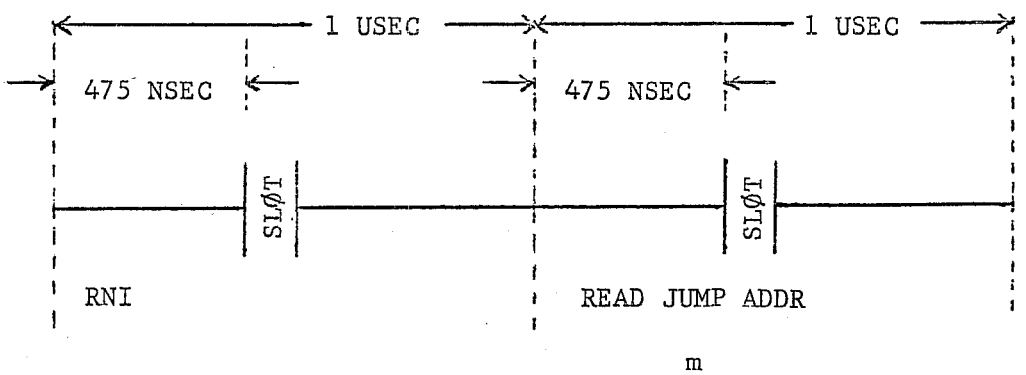
FLOW DIAGRAM



TIMING

6400/6600

2 USEC

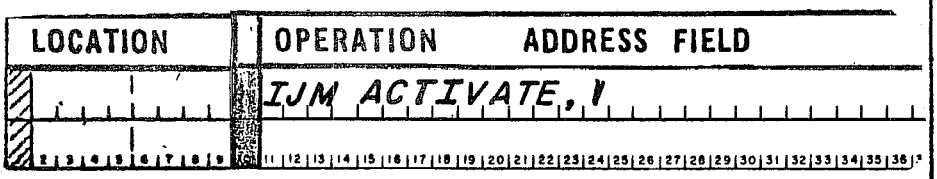


SLØT TIME = 100 NSEC

EXAMPLES

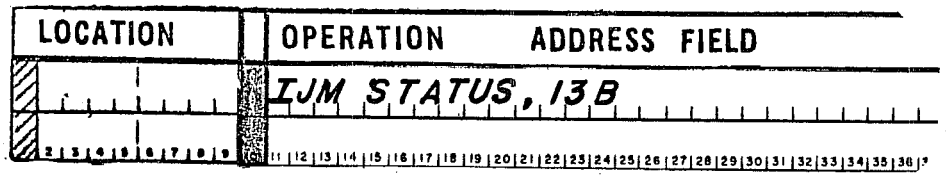
Exercise #1 - Test channel 1, and if it is inactive jump to a location called ACTIVATE, otherwise RNI @ P+2.

ANSWER



Exercise #2 - Test channel 13_8, and if it is inactive jump to location STATUS, otherwise RNI @ P+2.

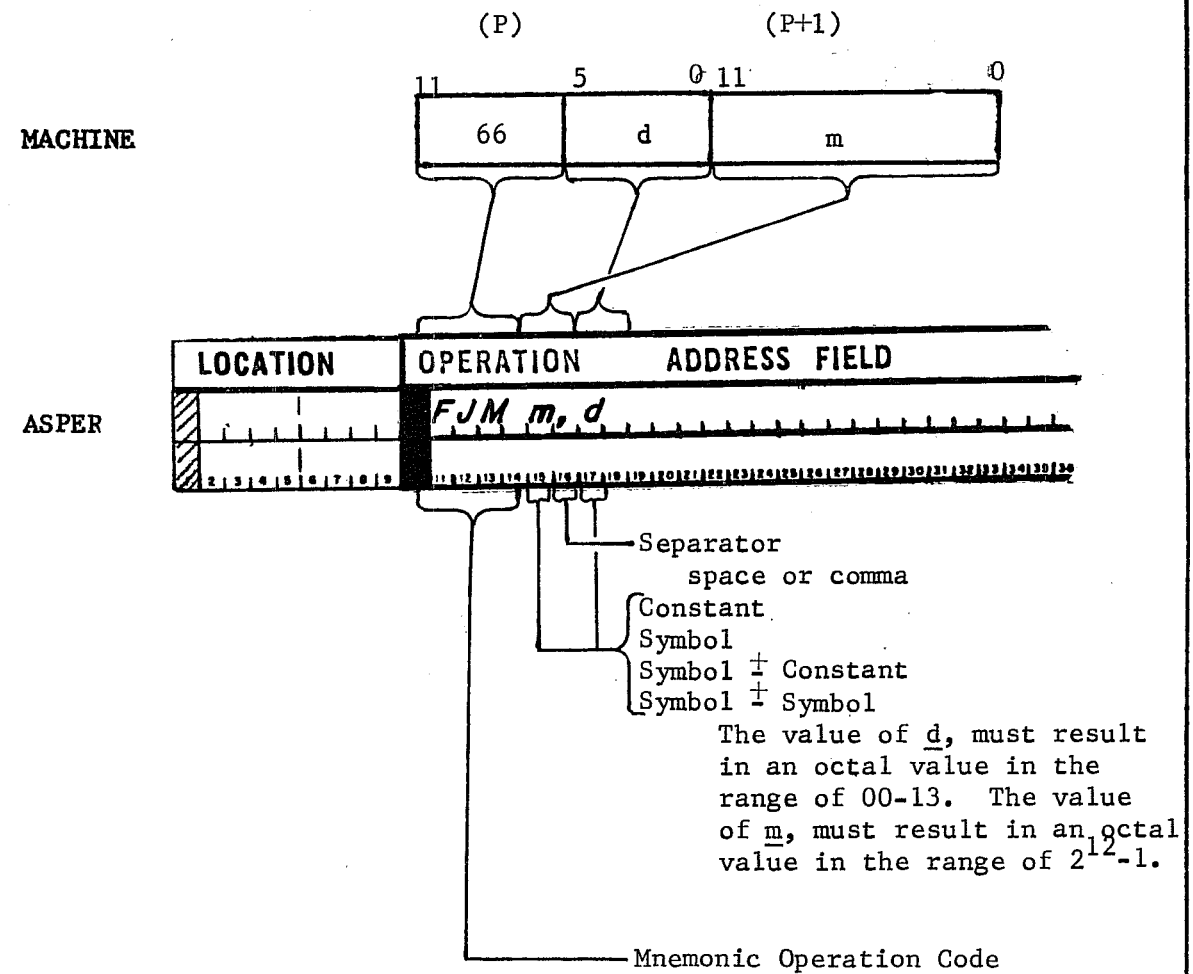
ANSWER



FJM

FULL JUMP

FORMATS



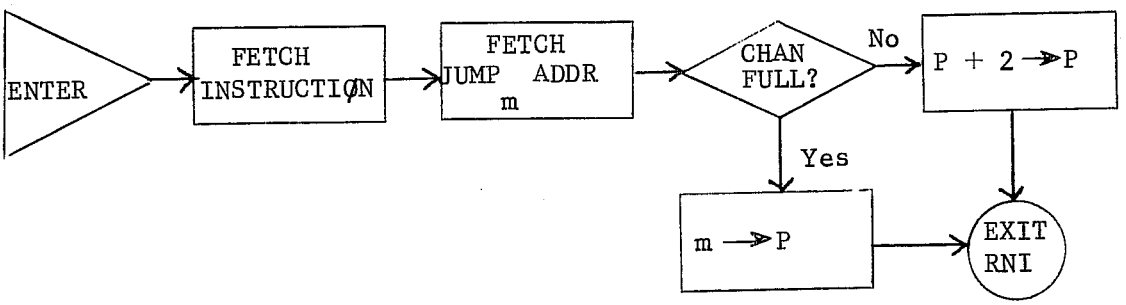
DESCRIPTION

Jump to m if channel d is full. This instruction provides a conditional jump to a new program sequence beginning at address m if the channel specified by d is full. If the channel is empty, the current program sequence continues. A channel is full: 1) Input operation and the input equipment sends a word to the channel register and sets the full flag. The channel goes empty when the PP accepts the word and clears the full flag. 2) Output operation and the PP places a word in the channel register and sets the full flag. The channel is empty when the external equipment accepts the word and clears the full flag.
RNI @ P+2

REFERENCES :

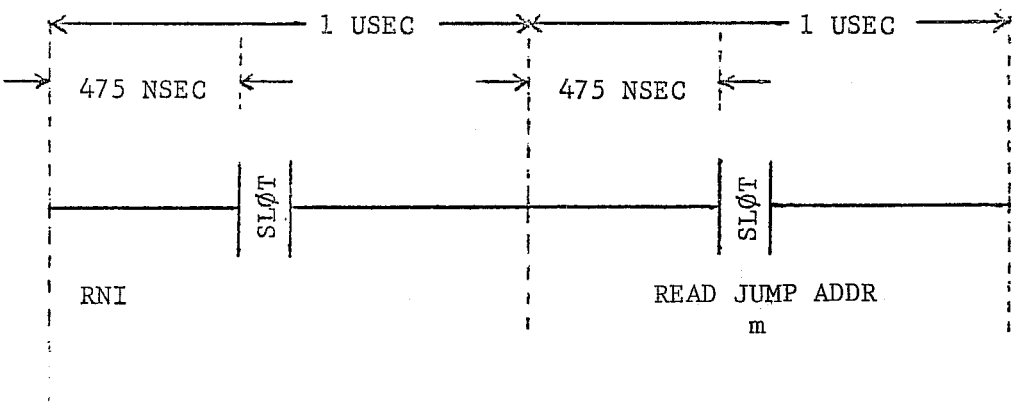
FJM

FLOW DIAGRAM



TIMING

6400/6600 2 USEC

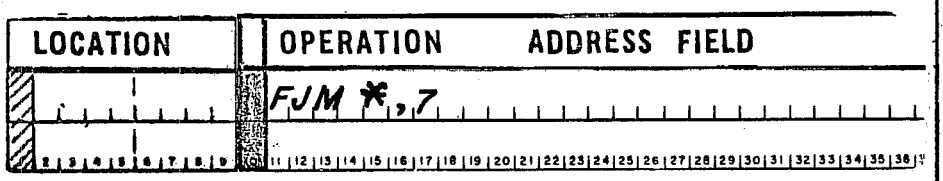


SLØT TIME = 100 NSEC

EXAMPLES

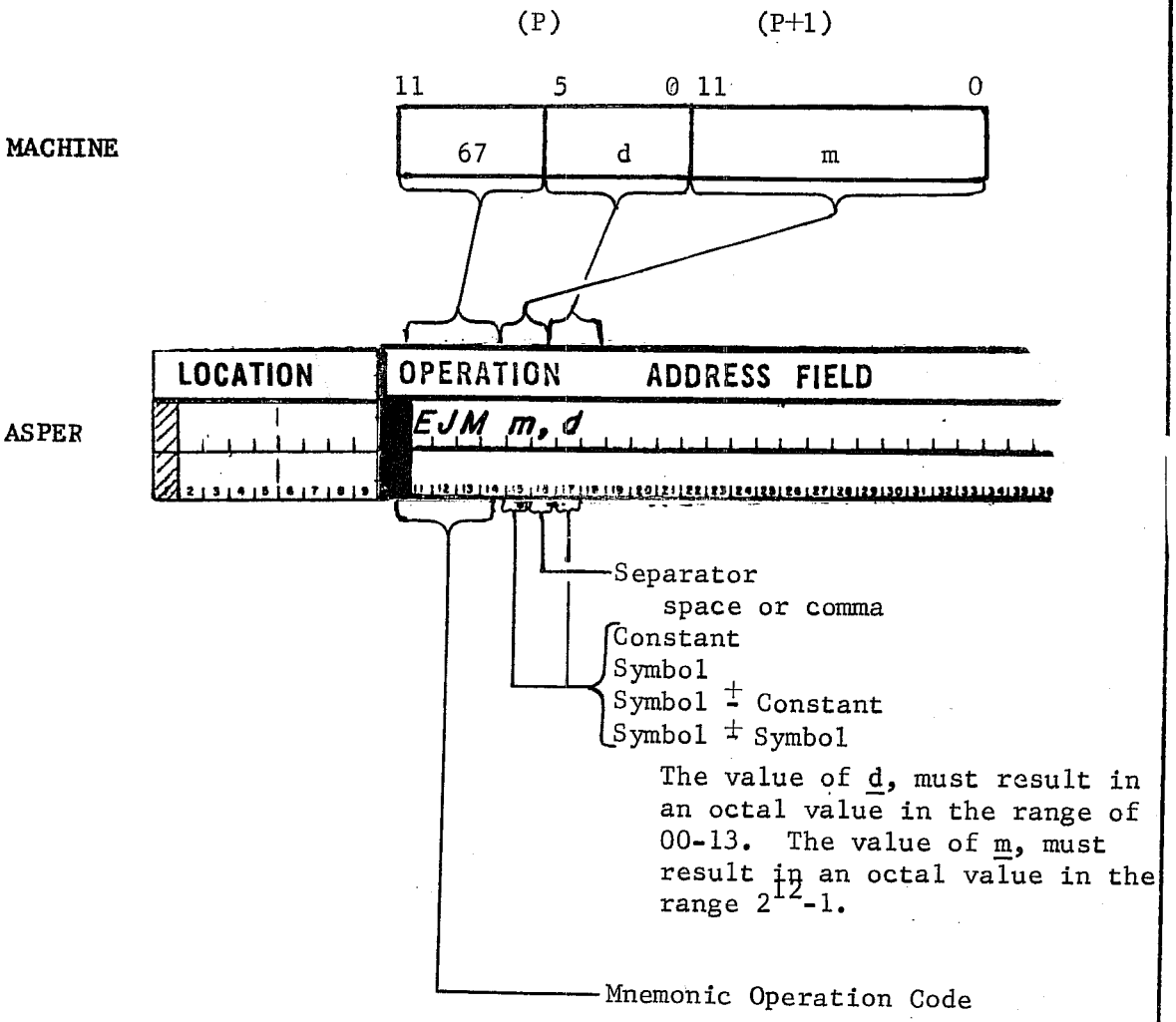
Example #1 - Code an instruction to test a channel to see if an output operation is completed. If not wait and otherwise RNI @ P+2 and disconnect the channel (07).

ANSWER



FORMATS

EMPTY JUMP



DESCRIPTION

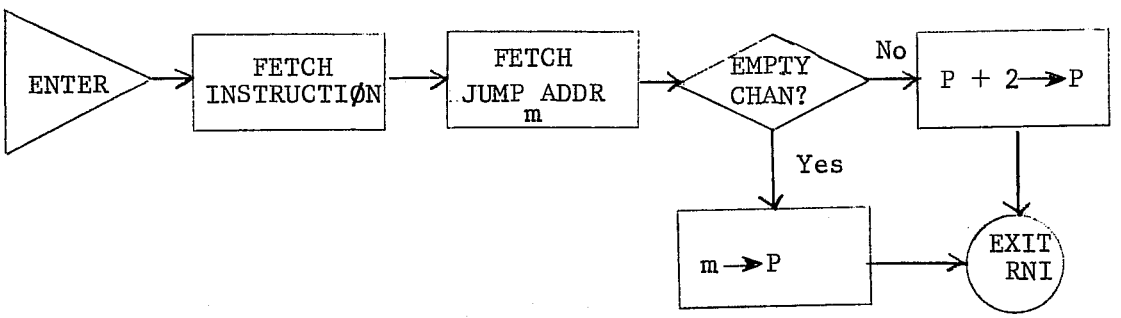
Jump to m if channel d empty. This instruction provides a conditional jump to a new program sequence beginning at address m if the channel specified by d is empty. If the channel is full, the current program sequence continues. (See page 55-0 for full and empty explanation.)

RNI @ P+2

REFERENCES :

EJM

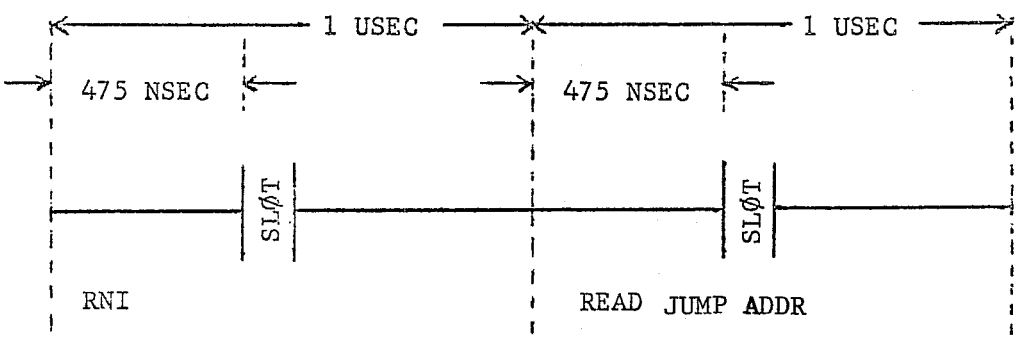
FLOW DIAGRAM



TIMING

6400/6600

2 USEC

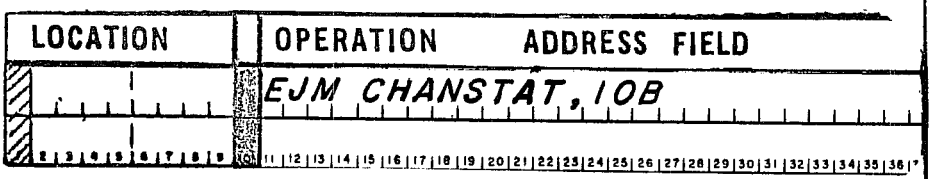


SLØT TIME = 100 NSEC

EXAMPLES

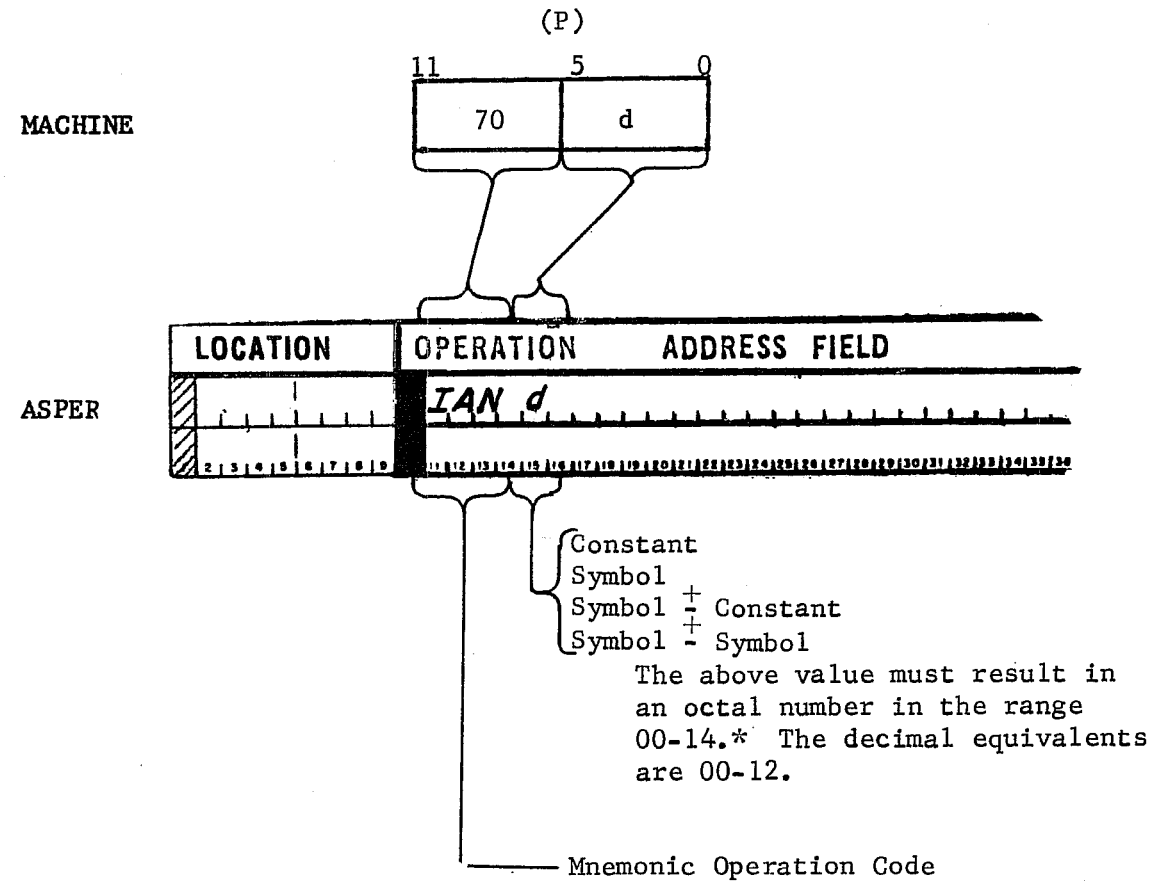
Exercise #1 - Code an instruction to test channel 10_8 for the empty condition. If it is empty jump to CHANSTAT, otherwise RNI @ P+2.

ANSWER



FORMATS

INPUT TØ A



DESCRIPTION

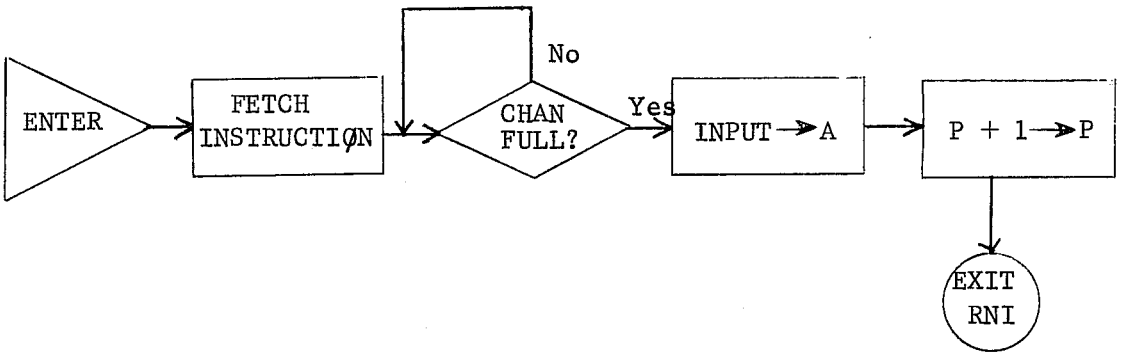
This instruction transfers a word from input channel d to the lower 12 bits of the A-Register. The upper 6-bits are zero. Before this instruction is executed the channel must have been set activate, and the external equipment previously selected for this operation. RNI @ P+1

*The real time clock is read by simply giving this instruction with channel 14_8.

REFERENCES :

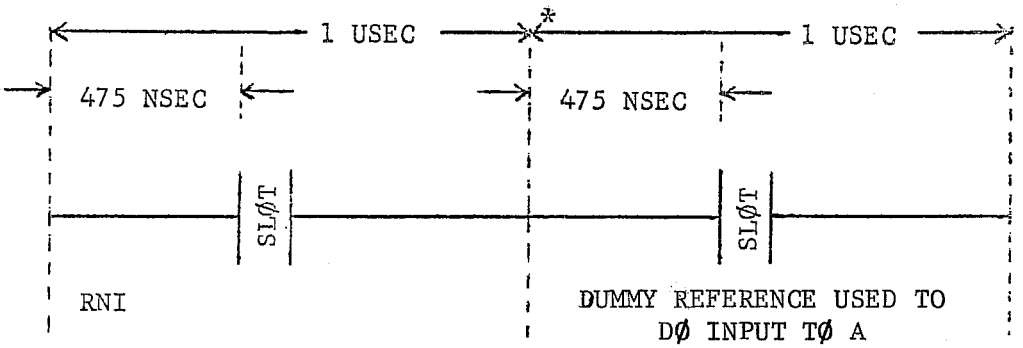
IAN

FLOW DIAGRAM



TIMING

6400/6600 2 USEC



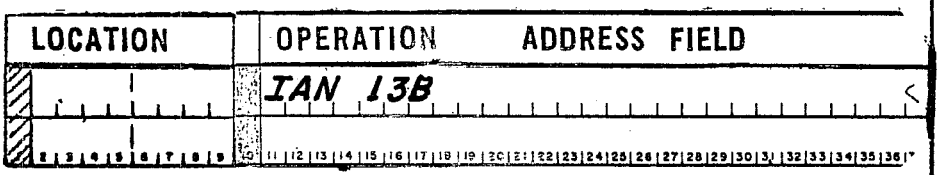
*The number of times it loops depends upon the speed of the external equipment. However, for status responses, the select and following activate bring the data word in the input register.

SLØT TIME = 100 NSEC

EXAMPLES

Exercise #1 - Code an instruction to bring the status in from channel 13 on a piece of I/O equipment that was properly selected for that operation, prior to this instruction. The upper 6-bits of A will contain zeros.

ANSWER

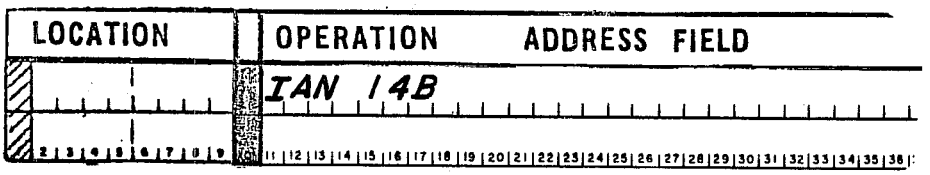


Exercise #2 - Code an instruction to make reading from the RTC (Real Time Clock).

NØ previous instructions are required.

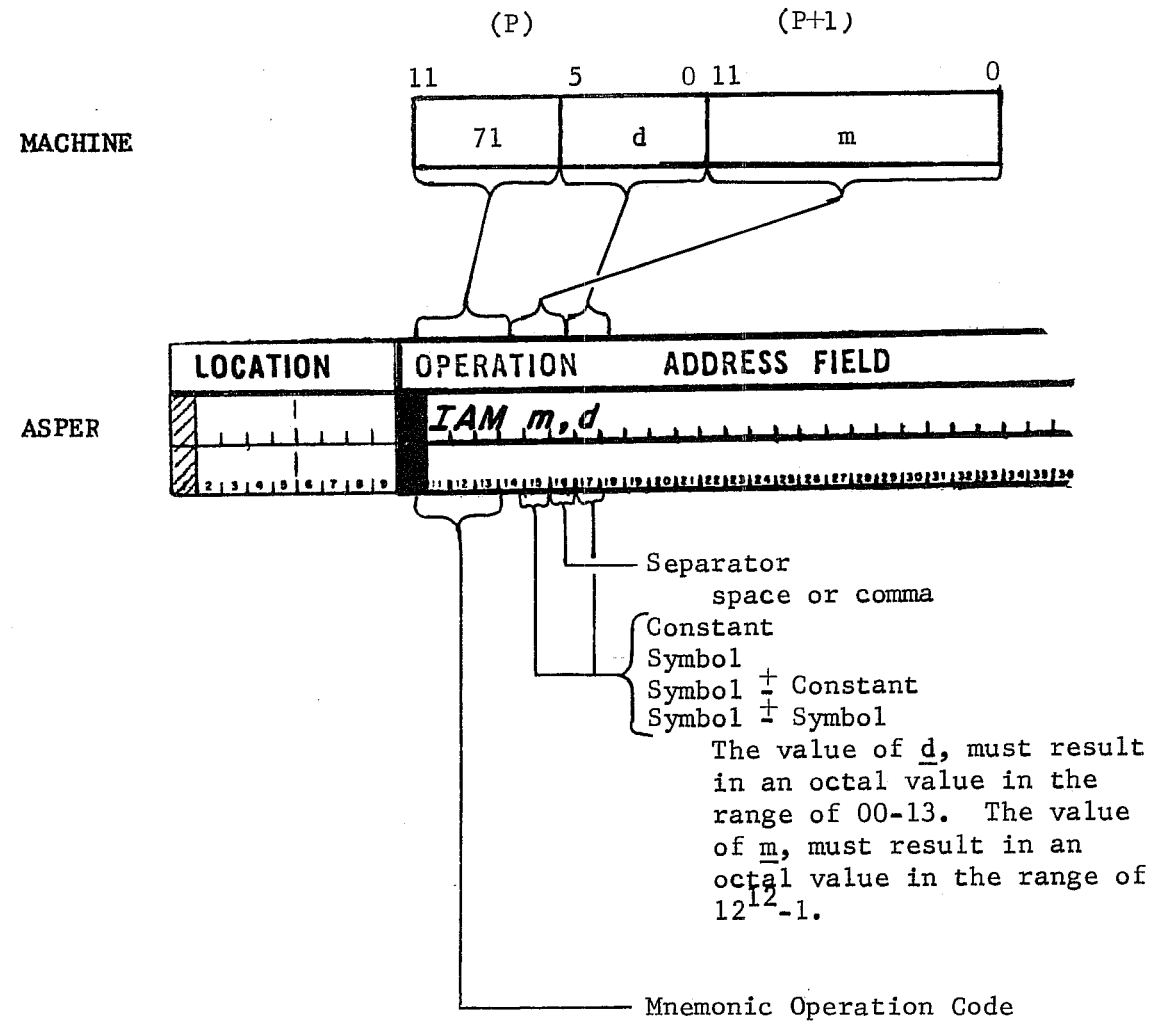
Note: RTC is on channel 14_8 and used for input only.

ANSWER



FORMATS

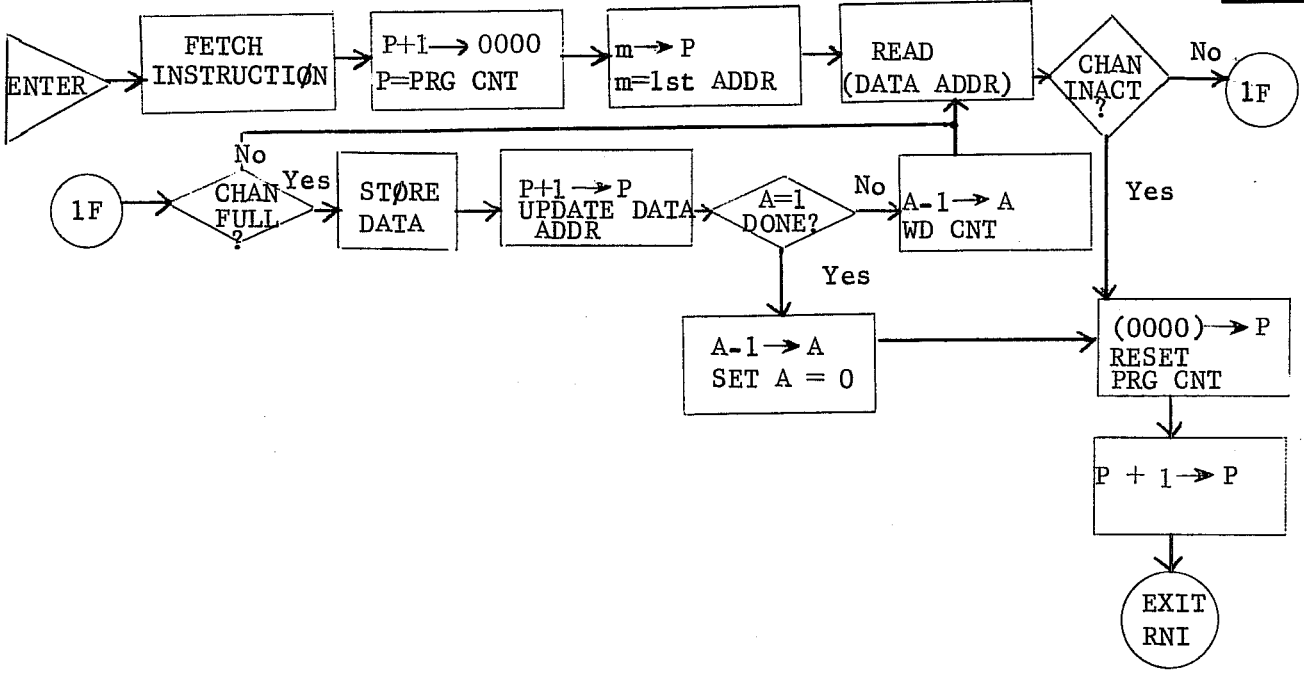
INPUT A BLØCK



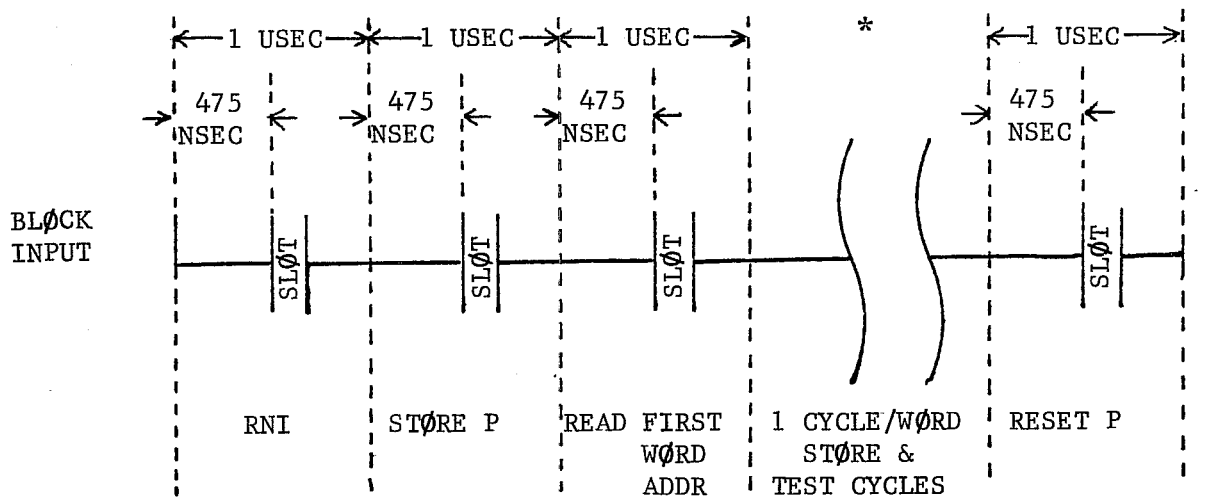
DESCRIPTION

Input (A) words from channel d to m. This instruction transfers a block of words from input channel d to PP memory, beginning at a location specified by m. The A-Register contains the block length and is reduced by one as each word is read. The input operation is completed when A=0. The equipment selection and channel activate must have been done prior to this instruction. During this instruction the current contents of the P-Register are stored in PP location 0000. The P-Register is used as an address to store the data, and is restored to its original contents upon completion of the operation. RNI @ P+2.

REFERENCES :



6400/6600 MINIMUM 4 PLUS 1/WORD usec



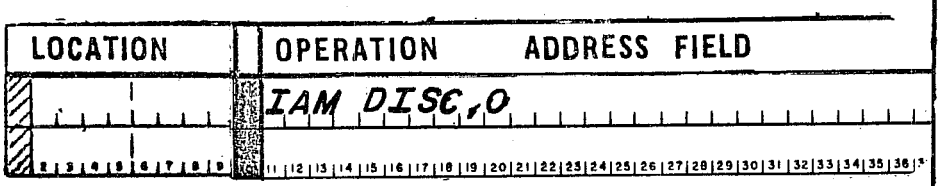
* The number of cycles depend upon the number of words, plus the speed of the input/output equipment.

SLØT = 100 nsec

EXAMPLES

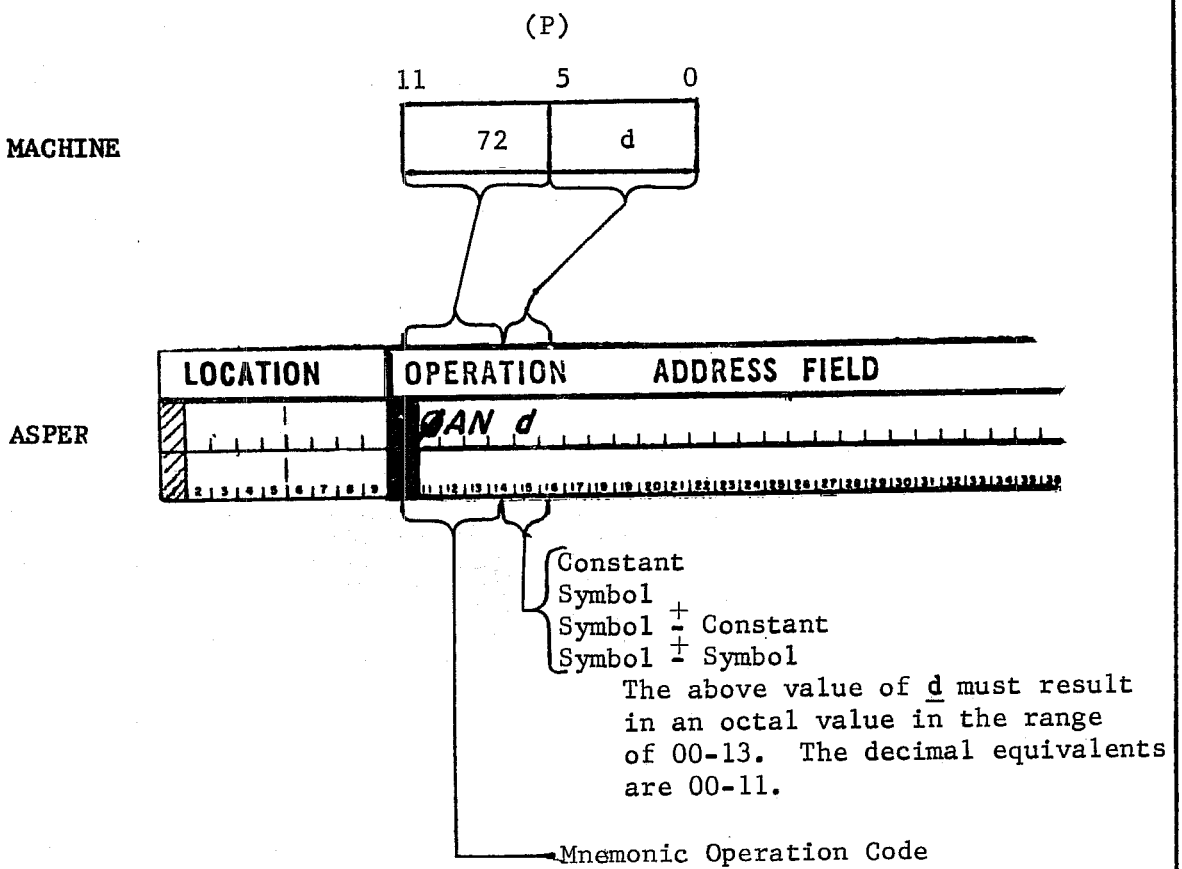
Exercise #1 - Assume the external equipment has been properly selected and the channel is ready. Code an instruction to input a block of words from Channel 0, to starting address DISC (this would be m) the A-Register would contain the number of words to be transferred. All transfers will be 12-Bits.

ANSWER



FORMATS

ØUTPUT FRØM A



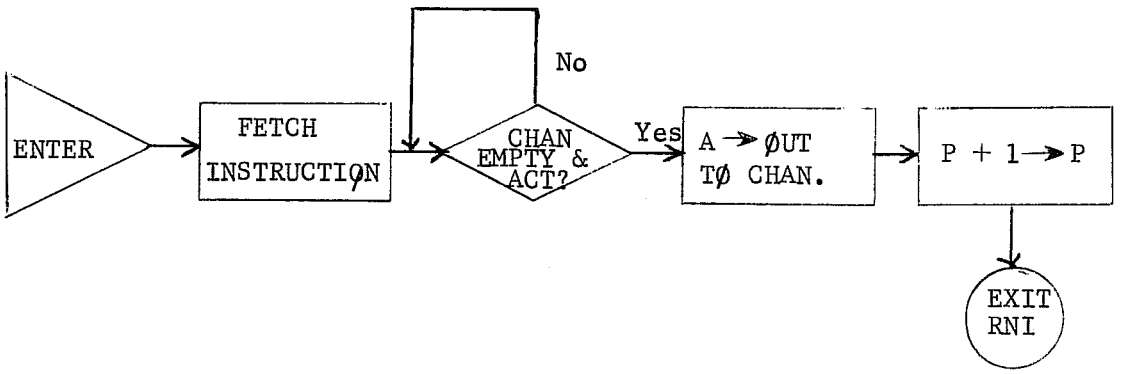
DESCRIPTION

Output (A) on channel d. This instruction transfers a word from the lower 12 bits of the A-Register to output channel d. The external equipment and channel activate must have been previously set up.

REFERENCES :

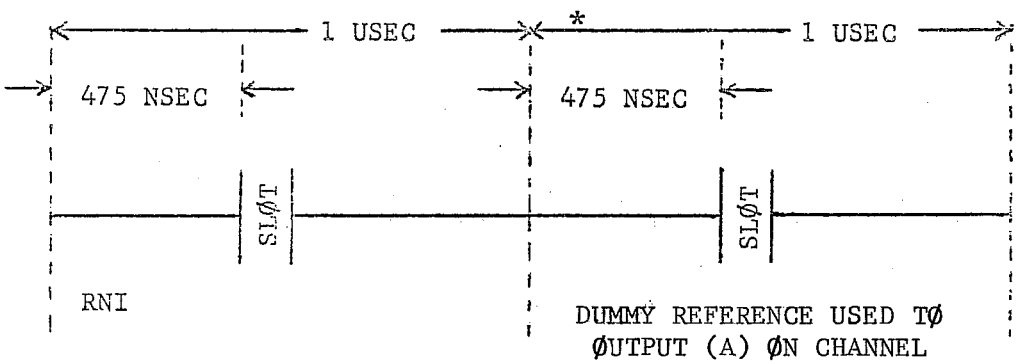
ØAN

FLOW DIAGRAM



TIMING

6400/6600 2 USEC



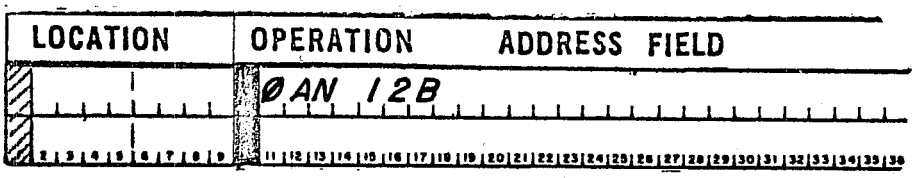
*The number of times it loops depends upon the speed of the equipment it is communicating with.

SLØT TIME = 100 NSEC

EXAMPLES

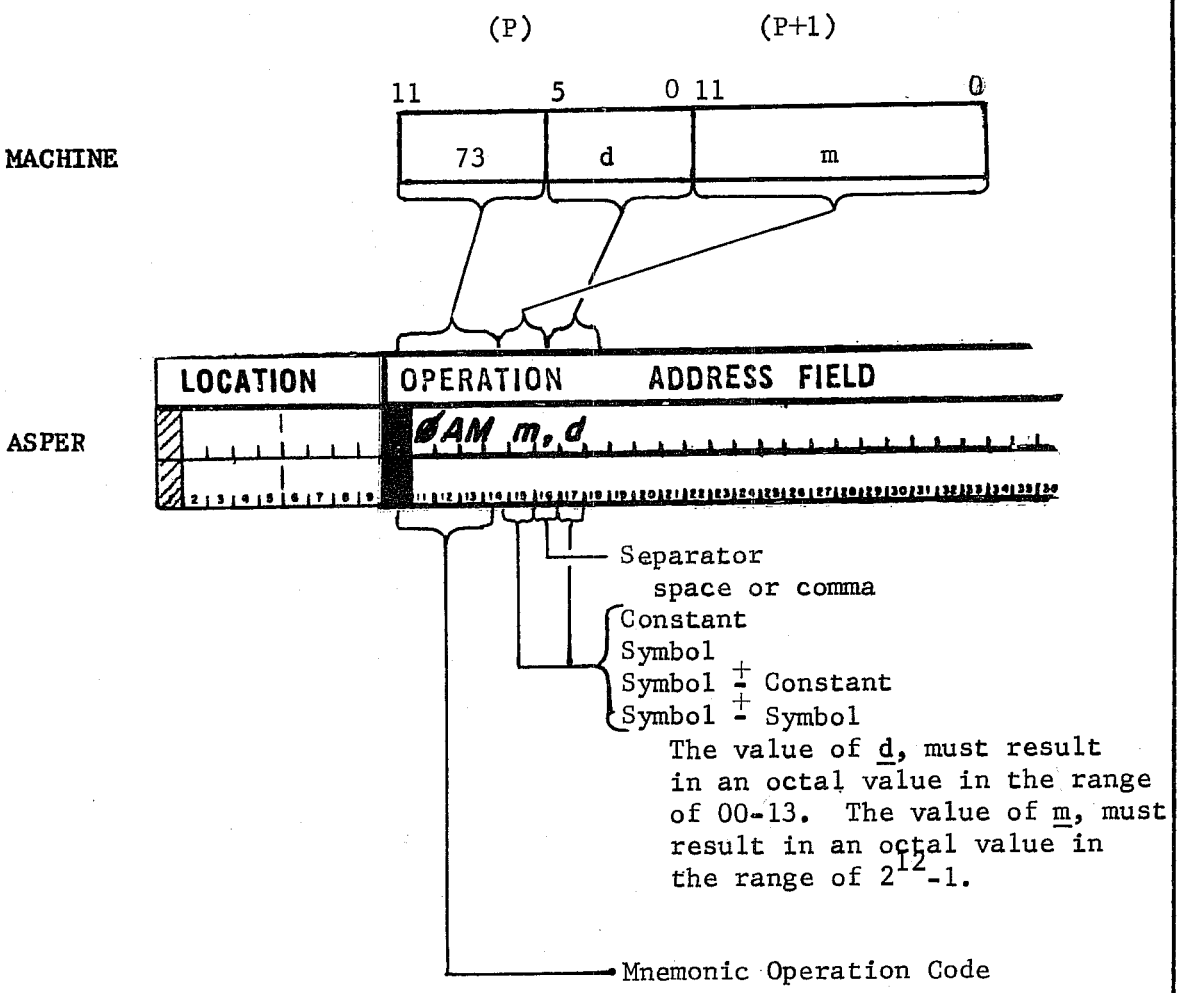
Exercise #1 - Code an instruction to output the lower 12-bits of the A-Register to a previously selected piece of equipment output to channel 12_8.

ANSWER



FORMATS

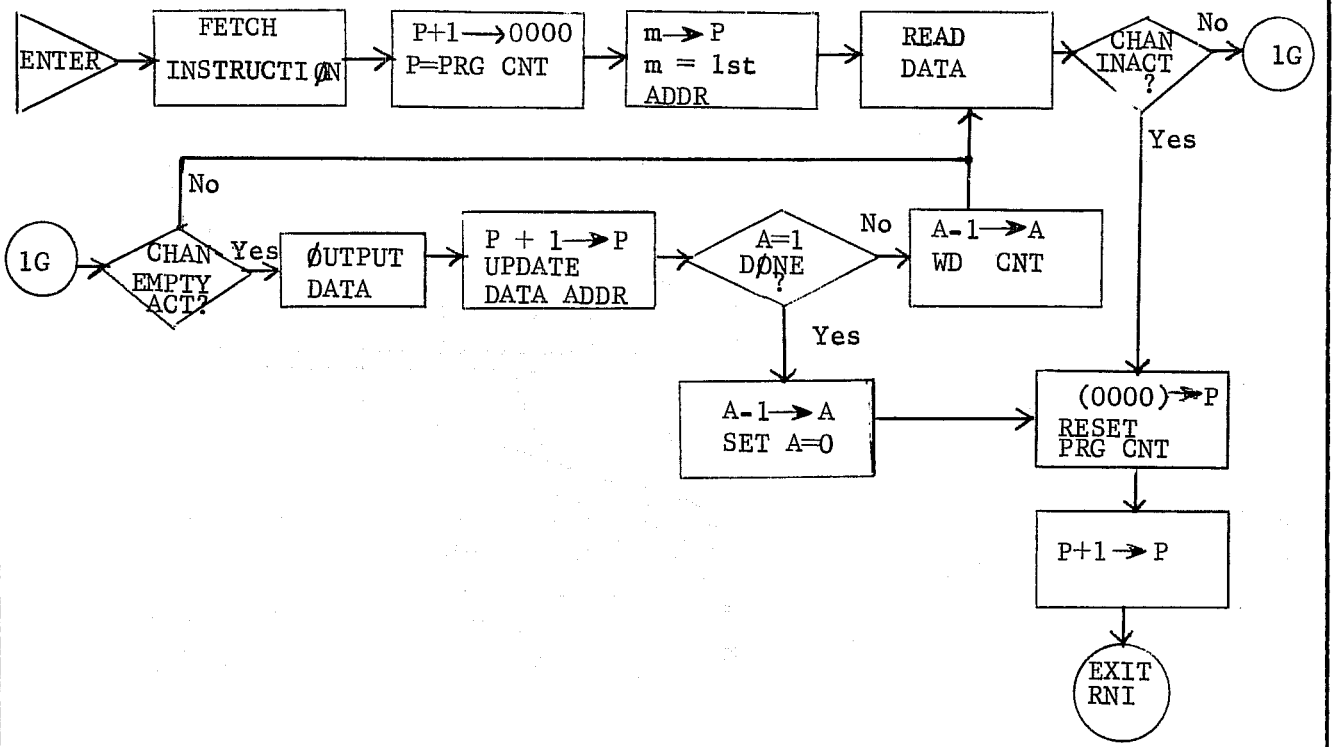
ØUTPUT A BLØCK



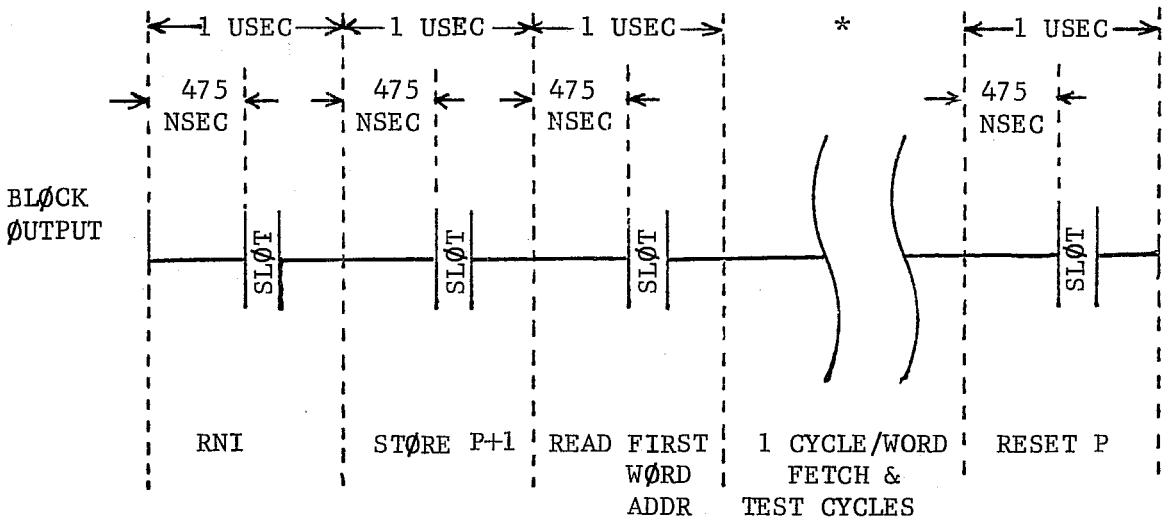
DESCRIPTION

Output (A) words from m on channel d. This instruction transfers a block of words on output channel d from PP memory, beginning at the location specified by m. The contents of the A-Register specify the number of words to be sent and are reduced by one as each word is sent. The operation is completed when A = 0. The equipment selection and channel activate must have been done prior to this instruction. During this instruction the current contents of the P-Register are stored in location 0000. The P-Register is used as an address to store the data, and is restored to its original contents upon completion of the operation. RNI @ P+2.

REFERENCES :



6400/6600 MINIMUM 4 PLUS 1/WORD usec



* The number of cycles depend upon the number of words, plus the speed of the input/output equipment.

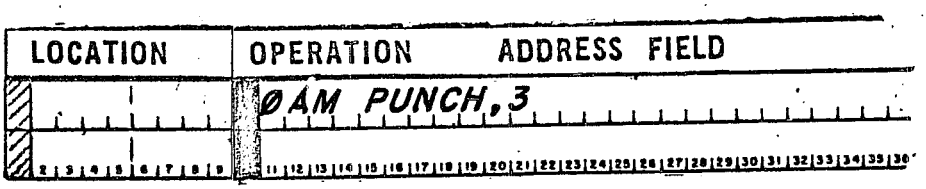
SLØT = 100 nsec

EXAMPLES

Exercise #1 - Code an instruction to output a block of information to a previously selected piece of equipment. The equipment is on Channel 3, the first word address in memory is called PUNCH, and the A-Register contains the length of the block.

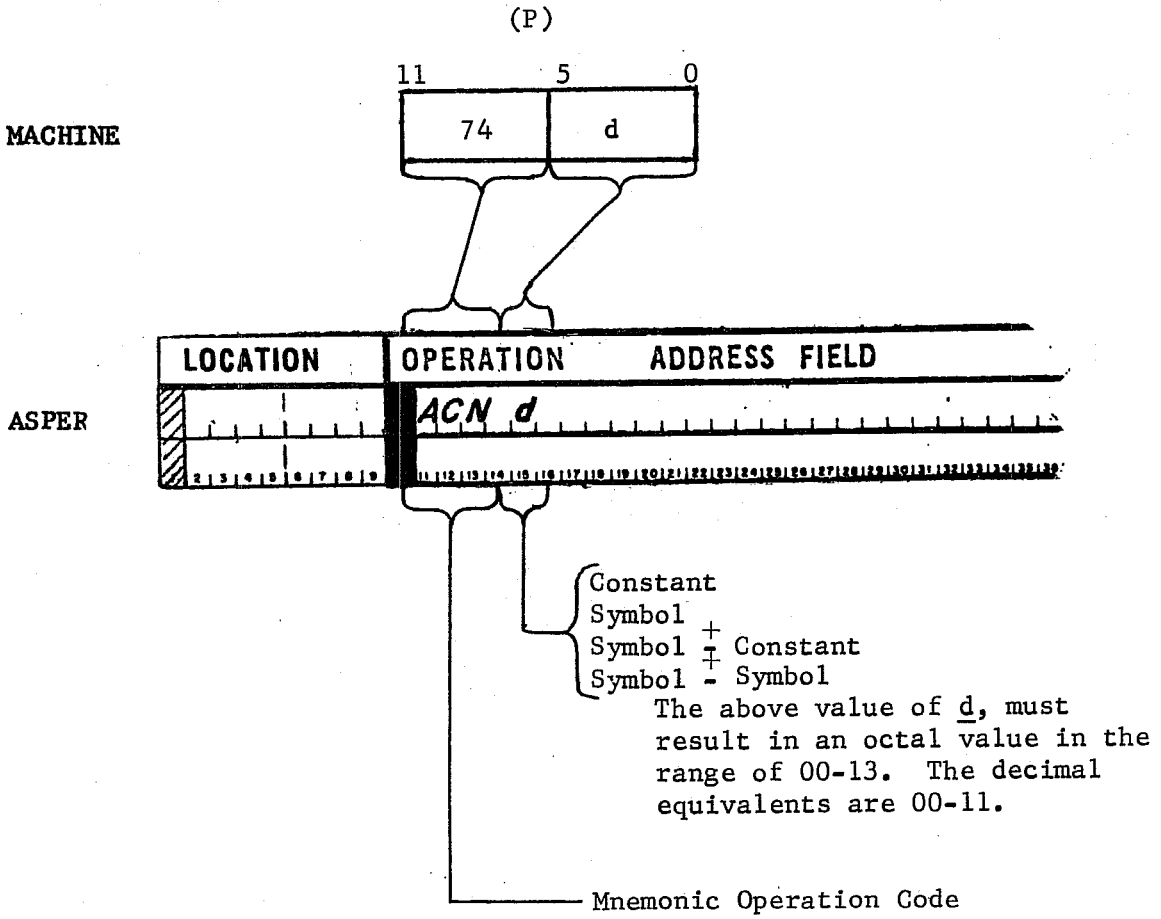
All word transfers will be 12-bits.

ANSWER



FORMATS

ACTIVATE CHANNEL



DESCRIPTION

This instruction activates the channel specified by d. This instruction must precede a 70-73 instruction. Activating a channel alerts and prepares the I/O equipment for the exchange of data.

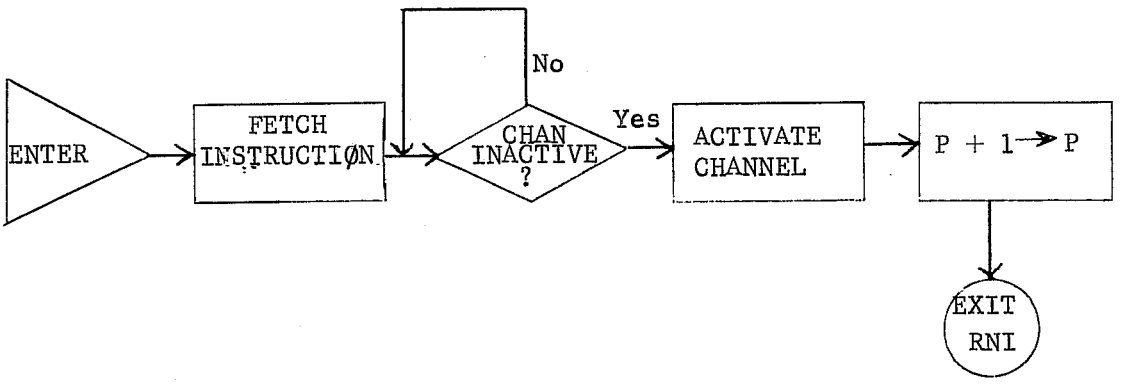
Note: Trying to activate a channel that is already active would hang up that one PP, until a monitor, (or some other PP) would deactivate it.

RNI @ P+1

REFERENCES :

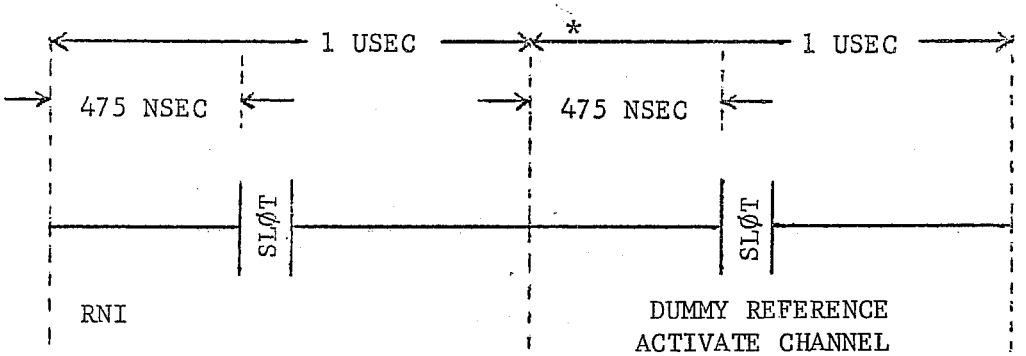
ACN

FLOW DIAGRAM



TIMING

6400/6600 2 USEC



*Instruction will loop indefinitely if the channel is already active.

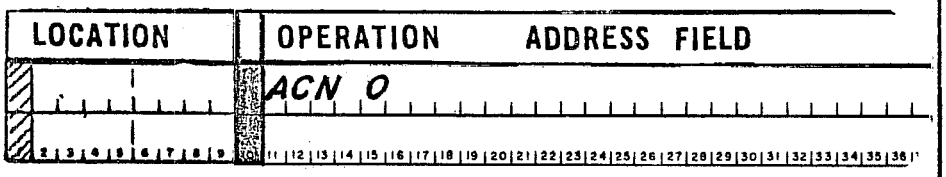
SLØT TIME = 100 NSEC

EXAMPLES

Exercise #1 - Code an instruction to activate channel 0.

Note: If the channel is already active the processor executing this instruction will loop indefinitely.

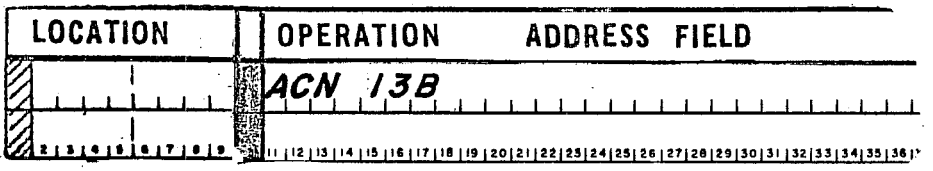
ANSWER



Exercise 2 - Code an instruction to activate Channel 13_8.

Note: See above note

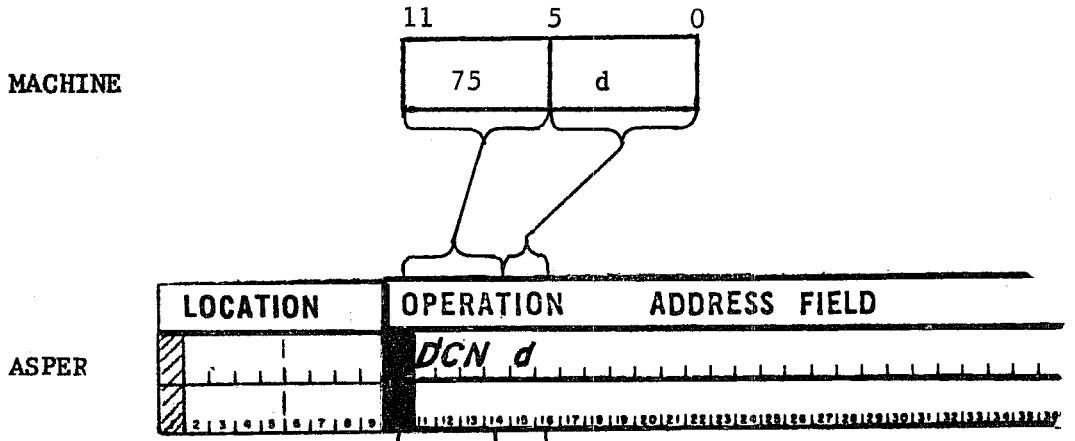
ANSWER



FORMATS

DISCØNNECT CHANNEL

(P)



Constant
Symbol + Constant
Symbol - Symbol

The above value of d, must result in an octal value in the range of 00-13. The decimal equivalents are 00-11.

Mnemonic Operation Code

DESCRIPTION

This instruction deactivates the channel specified by d. This stops the I/O equipment and the buffer terminates.

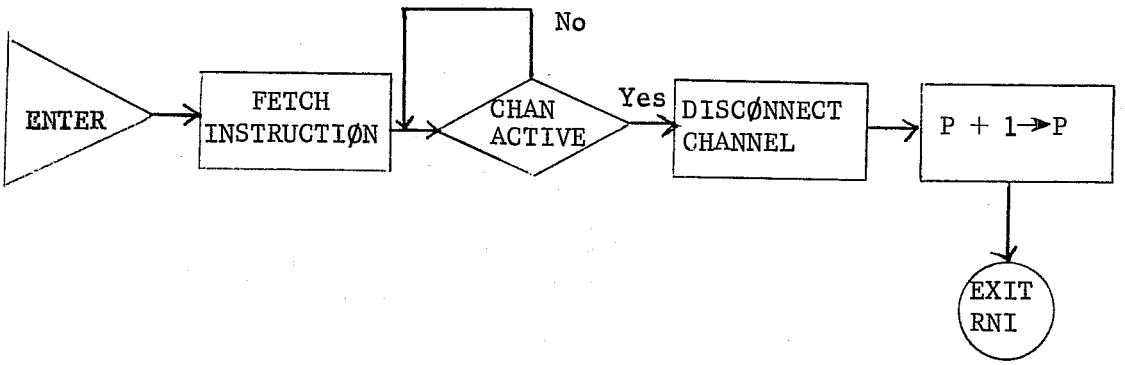
Note: Trying to deactivate a channel that is already deactive would hang up that one PP, until a monitor, (or some other PP) would activate it.

RNI @ P+1

REFERENCES :

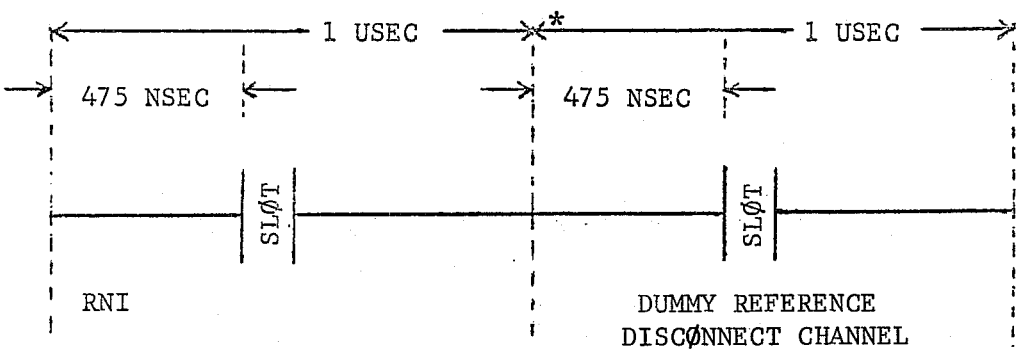
DCN

FLOW DIAGRAM



TIMING

6400/6600 2 USEC



*Instruction will loop indefinitely if the channel is already deactive.

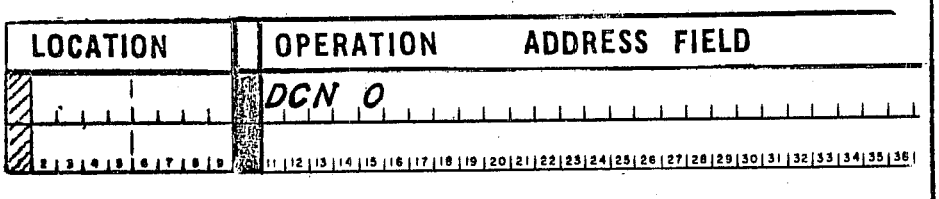
SLØT TIME = 100 NSEC

EXAMPLES

Exercise #1 - Code an instruction to deactivate (or disconnect) Channel 0.

Note: If the channel is already inactive the processor will loop indefinitely.

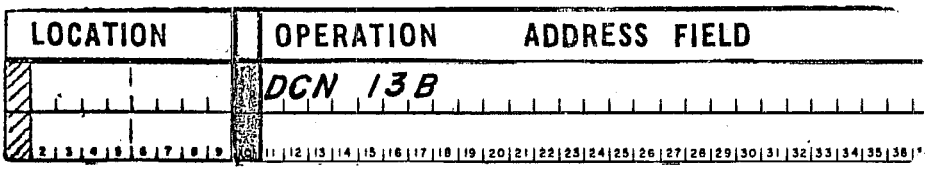
ANSWER



Exercise #2 - Code an instruction to deactivate channel 13_8.

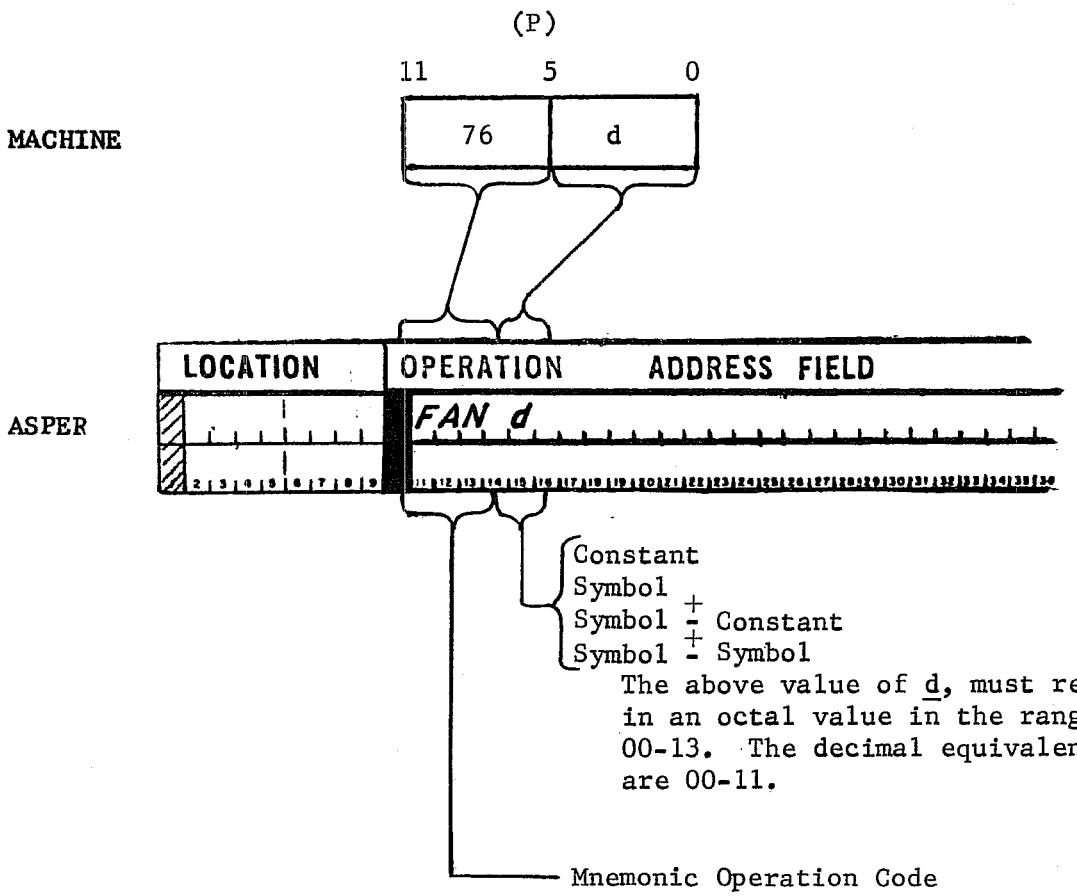
Note: See above note.

ANSWER



FORMATS

FUNCTIØN CØDE (A)



DESCRIPTION

Function (A) on channel d. This instruction sends out on channel d the external function code, previously placed in the lower 12-bits of the A-Register.

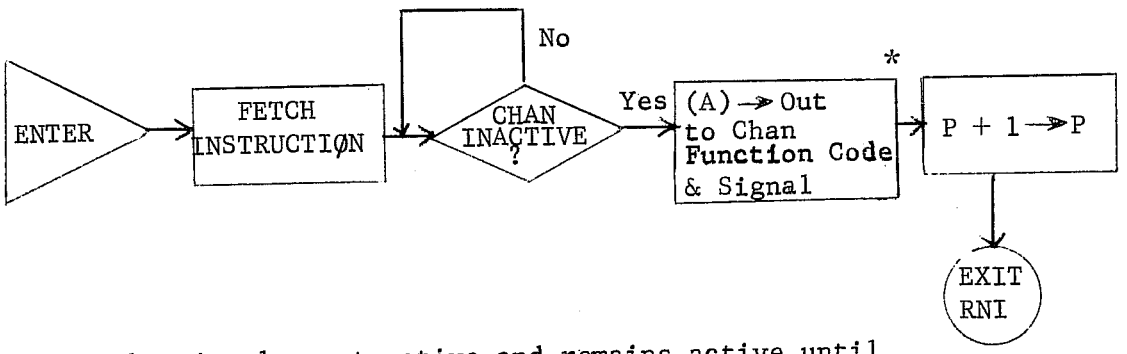
Note: Trying to function a channel that is in the active state would hang that one PP indefinitely, until a monitor, or some other PP deactivated that channel.

RNI @ P+1

REFERENCES :

FAN

FLOW DIAGRAM

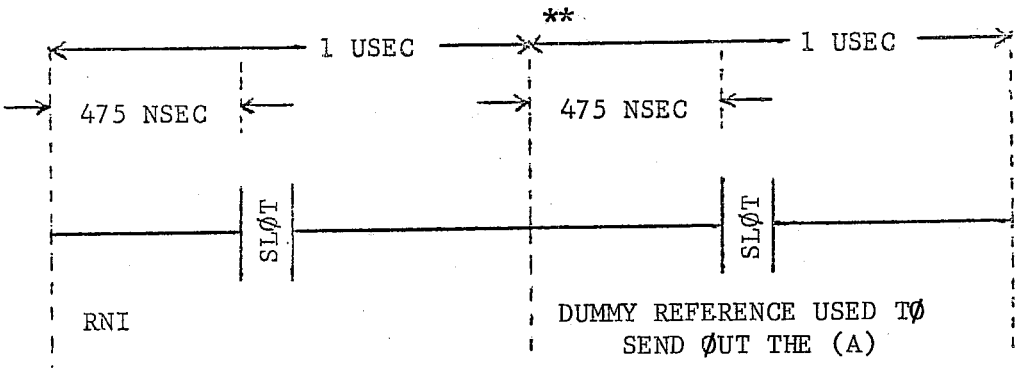


* The chan is also set active and remains active until the external equipment responds with an inactive; therefore, the PP could continue reading instructions, but the chan remains active until the equipment responds.

TIMING

6400/6600

2 USEC



**The instruction will loop indefinitely if the channel is already active.

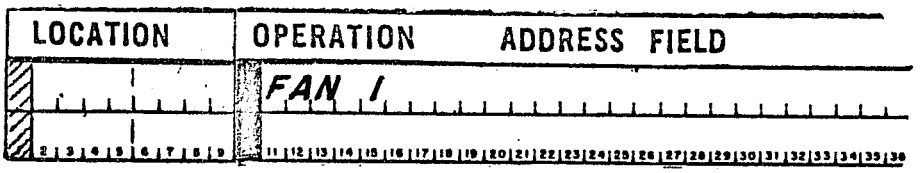
SLØT TIME = 100 NSEC

EXAMPLES

Exercise #1 - Code an instruction that will send as a function code the lower 12-bits of the A-Register. d specifies the channel number (01).

Note: The channel must be inactive prior to a function code instruction or the processor will loop indefinitely.

ANSWER

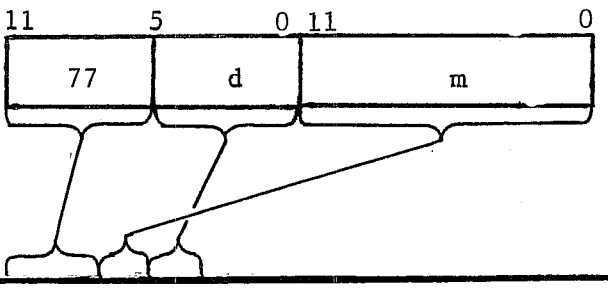


FORMATS

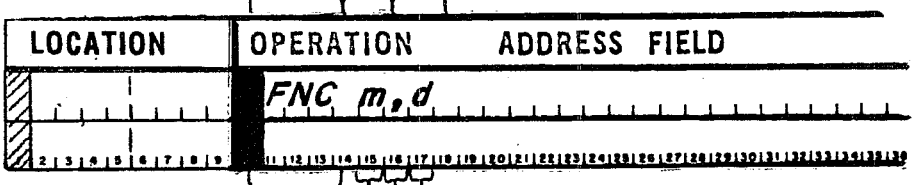
FUNCTIØN CØDE m

(P) (P = 1)

MACHINE



ASPER



Separator
space or comma
Constant
Symbol
Symbol ± Constant
Symbol ± Symbol

The value of d, must result in an octal value in the range of 00-13.
The value of m, must result in an octal value in the range of 2^{12}-1.

Mnemonic Operation Code

DESCRIPTION

Function m on Channel d. This instruction sends out on channel d the external function code specified by m.

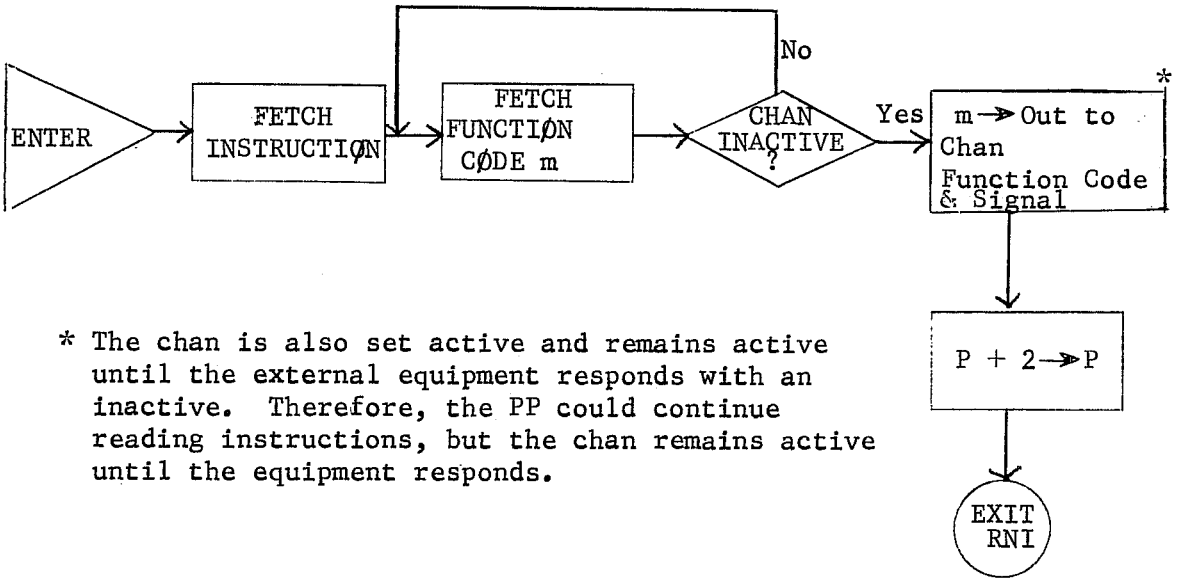
Note: Trying to function on a channel that is already active would hang up that one PP indefinitely until a monitor, or some other PP deactivated the channel.

RNI @ P+2

REFERENCES :

FNC

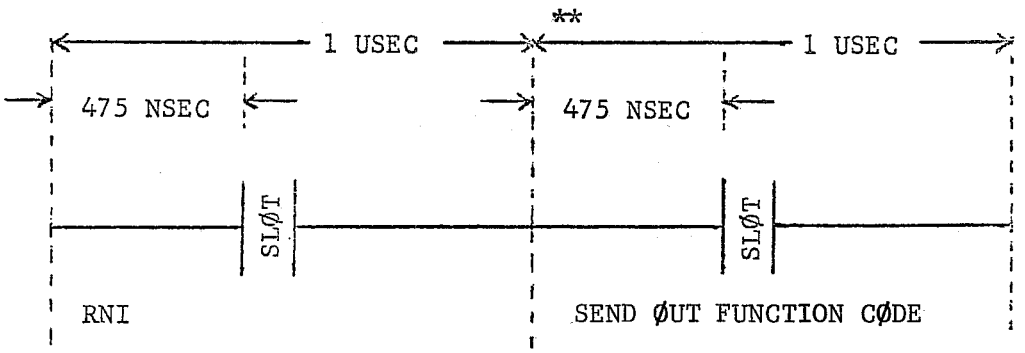
FLOW DIAGRAM



* The chan is also set active and remains active until the external equipment responds with an inactive. Therefore, the PP could continue reading instructions, but the chan remains active until the equipment responds.

TIMING

6400/6600 2 USEC



**The instruction will loop indefinitely if the channel is already active.

SLØT TIME = 100 NSEC

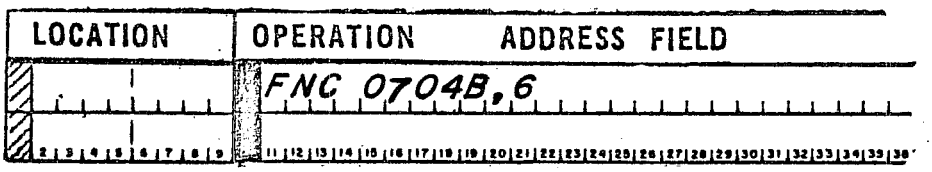
EXAMPLES

Exercise #1 - Code an instruction that sends out to a piece of external equipment the select code (m) on channel 6.

Note: See note on page 63-2.

0704 = m = 405 - card reader status request.

ANSWER



CONTROL DATA INSTITUTES

Arlington, Virginia 22204
3717 Columbia Pike

Atlanta, Georgia 30327
500 Interstate North

Dallas, Texas 75235
Suite 224 Garden Mall
Frito-Lay Building

Frankfurt, Germany
Bockenheimer Landstrasse 10
6 Frankfurt/Main

Los Angeles, California 90045
5630 Arbor Vitae Street

Miami, Florida 33131
174 East Flaglar

Minneapolis, Minnesota 55408
3255 Hennepin Avenue South

Rockville, Maryland 20952
11428 Rockville Pike

St. Louis, Missouri 63108
Des Peres Hall
3694 W Pine Street

Southfield, Michigan 48075
23775 Northwestern Highway

Waltham, Massachusetts 02154
60 Hickory Drive

New York, New York 10011
Control Data Computer Training School
66 West 12 Street

San Francisco, California 94102
760 Market Street, Suite 600

Oakland, California 94612
508 16th Street

Chicago, Illinois 60603
37 South Wabash

CONTROL DATA
CORPORATION