Layout

Design of ALU

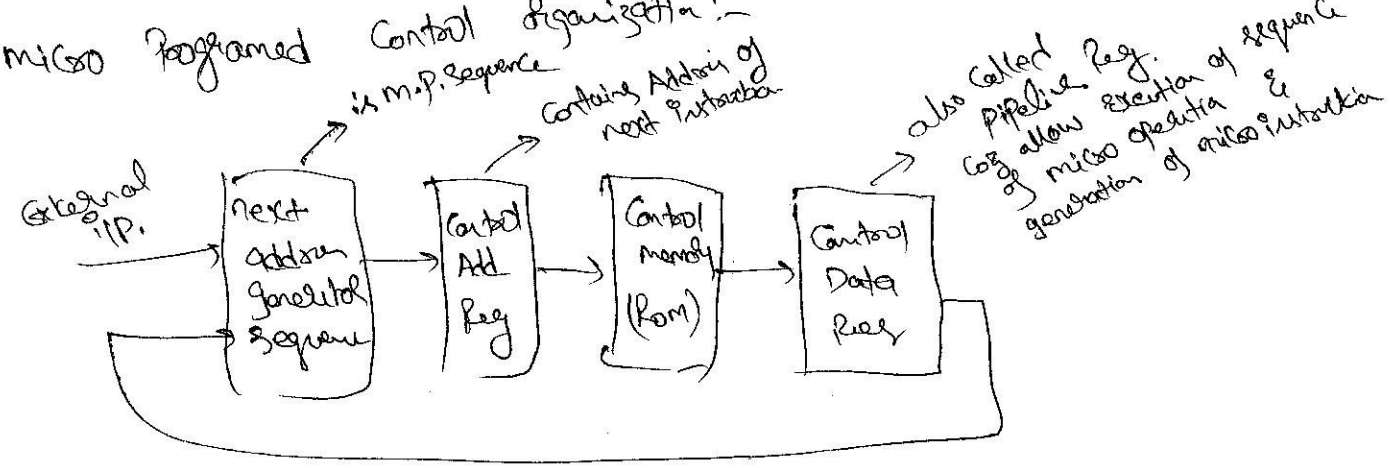
Function is to initiate sequence of micro operations ← Control Unit ← Control memory.

Control Unit
- Hardwired Control
  - Produce fast mode
  - requires changes in wiring among various components if design is modified.
- Microprogrammed Control
  - Control info stored in Control memory
  - initiated the required sequence of micro operations no.
  - Any changes can be done by updating the micro program in C.M.
  - Main memory → (User storing Program constitutes of machine instruct & data)

Control memory:- (specify M.O)
- when - 1: M.O executed
- when - 0: State of register is not altered.

Each word in C.M contains micro instruction (M.I)
→ M.I specifies 1 (or) more M.O.
→ sequence of M.I constitute a micro program
→ Rom is used in C.M.

(2) Micro Programmed Control Organization:-



Activity of CPU Placing on Bus

Address	→	Address Bus
Read Controlled	→	Control Bus
Data in from memory	→	Data Bus
Data Bus	→	I. Reg

Time Cycle.

- T_1 Cycle } fetch.
- T_2 Cycle → decode.
- T_3 Cycle } Execute
- T_4 Cycle

Unit - III

Micro Programmed Control & Computer Architecture :-

Learn about { Design of Arithmetic Logic unit. & Control of Memory. } Depends on Control Unit → Hardwired, Micro Programmed.

(i) Micro Programmed Control :-

→ function of control unit is to initiate sequence of micro operations.

→ Control unit designed in 2-ways
- (a) Hardwired Control
- (b) Micro Programmed Control

(a) **Hardwired Control :-**

Control unit is said to be hardwired when the control signal is generated by hardware using Conventional logic design technique.

Adv :- optimized to produce a fast mode operation.

Dis Adv :- Requires changes in the wiring among the various components of the design has to be modified.

→ Risc Architecture concept is used hardwired control.

(b) **Micro Programmed Control :-**

↳ Control information is stored in control memory

↳ Control memory is programmed to initiate the required sequence of micro operations.

→ Adv :- Any changes can be done by updating the micro program in control memory.

1) Control * Memory :-

Control function specify micro operation in binary variables. { when it is '1' micro operation is executed. / where it is '0' state of register is not altered.

→ In hor organized system { C.S specify micro operations as groups of bits
select paths in multiplexers, decoders, & ALU.

→ Each word in Control memory contains micro instructions.

→ Micro instruction specifies 1 (or) more micro operations for system.

→ A sequence of micro-instructions constitute a micro program.

→ Rom is used as Control memory because one Control unit is in operation no need of any alterations in micro programs.

→ A Computer that employs micro programmed Control unit will have 2 memories

- main memory (Available to user for storing programs. It constitutes of machine instruction & data)
- Control memory. (Holds fixed micro program)

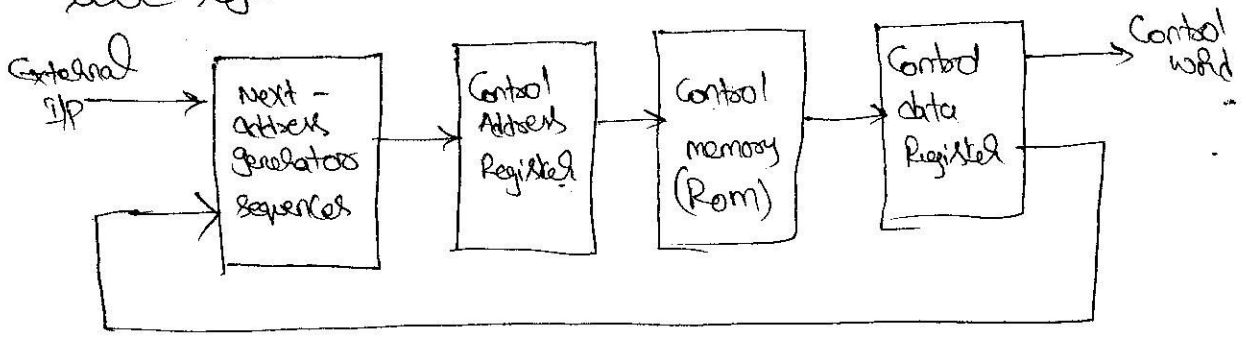
→ In Control memory

↳ Each Machine instruction initiates a series of micro instructions

→ micro instructions generates micro operations

(a) To fetch instruction from main memory
(b) To Evaluate the effective Address
(c) To Execute the operation specified by instruction
(d) To return the Control to fetch phase in order to repeat the cycle for next instruction.

Micro Programmed Control Organization:-



→ Control memory is assumed to be ROM which inserts a set of micro instructions

→ Control Memory Address Register (CAR) Contains the address of next micro instruction to be read.

→ When M.I is read from C.M it is transferred to Control Data Register (CDR)

→ The CDR sometimes called pipeline Register bcoz it allow executions simultaneously with the generation of microoperations specified by Control word of next microinstruction.

→ A M.I Contains bits for initiating micro operations in data Processor Part and bits that determine address sequence for Control memory.

→ The next address generator is called a Micro Program Sequencer as it determines the address sequence that is read from Control memory

Functions of micro program sequencer:-

- → Increment CAR by '1'.
- → Loading into CAR an address from memory
- → ~~loading~~ ~~or~~ transfering External address into CAR
- → Loading as initial address to start Control operations.

→ Generation of Control signals are generated by Software instead of hardwired circuits.

→ In hardware control signals.

t_0 : MAR ← PC // fetch instruction Address

t_1 : IR ← M[AR] ; PC ← PC+1 // find Effective Address

t_2 : DCD (IR) Decode ; MAR ← $IR_{(0:11)}$ Lower (12-bit) // decode..

Content

~~14~~MAR ~~→1%~~

Control bits

$LD_{MAR} \rightarrow C_0$ // Load memory Address Reg associated with bit C_0 (i.e 0th bit in Control

$OE_{PC} \rightarrow C_1$ // output Enable of Program Counter associat with bit C_1

$LD_{IR} \rightarrow C_2$ // Load instruction Reg - associate bit C_2.

$RD_m \rightarrow C_3$ // Read Control signal for memory C_3

$PC_{INR} \rightarrow C_4$ // PC increment is associated with bit C_4 in Control memory

$OE_{IR} \rightarrow C_5$ // output Enable of Instruction Reg with C_5.

Controls involved are: for instruction fetch:-

@ machine Cycle

1) LD_{MAR} ; OE_{PC} — be Active → T_0

2) LD_{IR} ; RD_m ; PC_{IR} — be Active → T_1

3) LD_{MAR} ; OE_{IR} — be Active → T_2

Control signals.

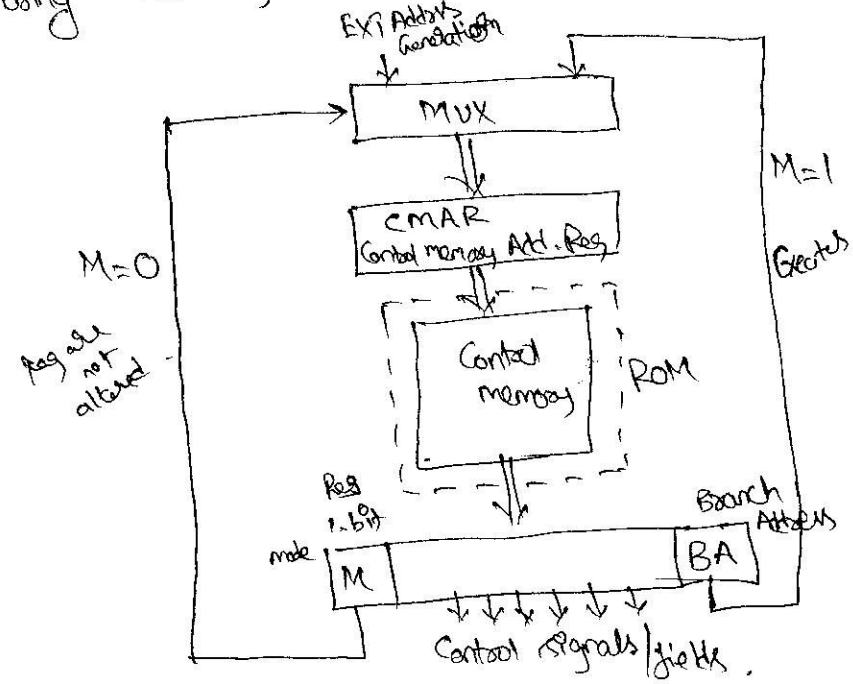
Sequence Counter.

Generation of Control signals in a sequence is done by having a memory whose every location contain n-no of bits.

no of Control signals = 'n'

A Control signal C_q = q^{th} bit in a memory location. (Every bit used to represent Control signal.)

For storing Control signal requires a memory called Control memory //.



(All components except Control memory take part in decoding next location to be read in memory so all this operation are called micro Program - sequencer)

	C_5	C_4	C_3	C_2	C_1	C_0		Required Control signal.
(MAR ← PC)	0	0	0	0	1	1	← T_0	$LD_{MAR} \leftarrow OE_{PC}$
(IR ← m[AR] ; PC ← PC+1)	0	1	1	1	0	0	← T_1	$LD_{IR} \leftarrow RD_m$; $PC_{INR} \leftarrow OE_{PC}$
(Decode ; MAR ← $IR_{(0-1)}$ bits)	1	0	0	0	0	1	← T_2.	$LD_{MAR} \leftarrow OE_{IR}$

Address	M	C_5	C_4	C_3	C_2	C_1	C_0	BA	machine Cycle
00	1	0	0	0	0	1	1	01	→ T_0
01	1	0	1	1	1	0	0	02	→ T_1
02	0	1	0	0	0	0	1	XX	→ T_2.

Consider the instruction :-

MOV R_1, R_2. ; T_3 : ← ($R_1 \leftarrow R_2$)

Control signals required.

OE_{R_2} → C_6 (bit) } 4th location Execution

LD_{R_1} → C_7 (bit)

(should be given from Binary Representation)

Addr	M	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0	BA	Machine Cycle
(E) 00	1	0	0	0	0	0	0	1	1	01	T_0
(F) 01	1	0	0	0	1	1	1	0	0	02	T_1
02	0	0	0	0	1	0	0	0	1	XX	T_2. (Decode)
MOV R_1, R_2	M									00	T_3. (After Execution is complete next goes to T_1 cycle)
	1	1	1	0	0	0	0	0	0	M.I	

(E) Should read from next memory location

(F) Address should from EXT-generator

Eg:- ADD R_1.

T_3 : DR ← R_1

T_4 : ACC ← ALU_{ADD} (ACC, DR). } Control signals needed

T_3: LD_{DR} ; OE_{R_1}

T_4: LD_{ACC} , ALU_{ADD} ; OE_{ALU}

LD_{DR} : C_8 ; LD_{ACC} : C_{10} ; OE_{ALU} : C_{12}

OE_{R_1} : C_9 ; ALU_{ADD} : C_{11}

ADD R_1, R_2

	M	C_{12}	C_{11}	C_{10}	C_9	C_8		C_0	BA
L →	1	0	0	0	1	1	0 0 . . .	0	L+1
	1	1	1	1	0	0	0 0	0	00

The simplest way to reduce the n:- of control signal in control memory instead of directly keeping control signal in control memory we just encode control signals and place the encoded control signal in control memory.

i.e $n = \lceil \lg 2^n \rceil$ → but this is disadvantage.

Horizontal M.P :-

→ Every bit is assigned to control signal

⇒ If more no: of control signals are present, then size of control word is also large. ⇒ instead of directly placing the control signal in control word,

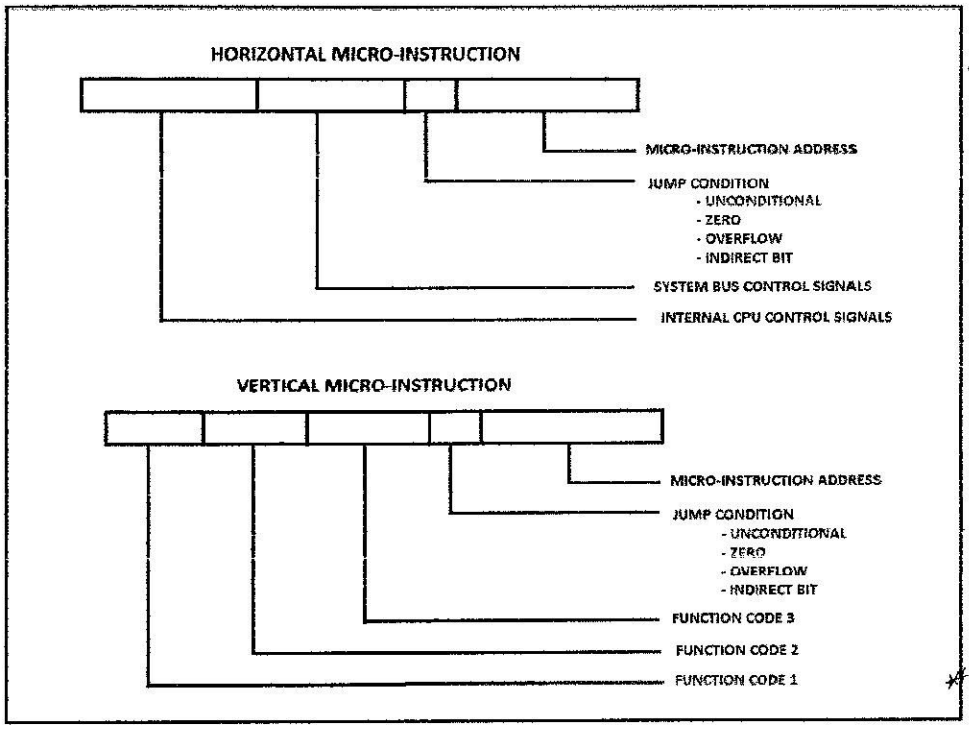
→ Encoded control signals placed in control memory.

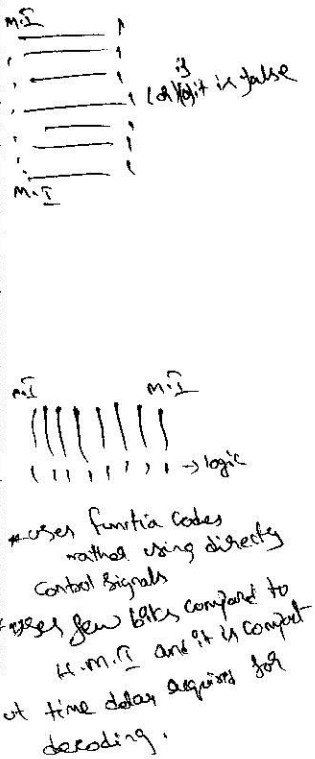
[redacted]

- ✓ *If we want to change the sequence of control signals*
  - ✓ *In hardwired control, entire hardware should be changed.*
  - ✓ *In micro-programmed, just change the control memory → more flexible and compact.*
- ✓ *Since generation of any control signal requires a control memory read operation, the micro-programmed control unit is slower than hardwired control unit.*
- ✓ *No. of bits in each control word = No. of control signals*
  → ***Horizontal micro-instruction***
- ✓ *If control signals are large, then size of control memory is also large → instead of directly placing the control signals in the control word, encoded control signals are placed in control memory → if no. of control signals = n, log(n) bits are required in each control word*
- ✓ ***Disadv: At any point of time we can activate only one control signal.***
- ✓ ***Sol: Vertical micro-instruction***

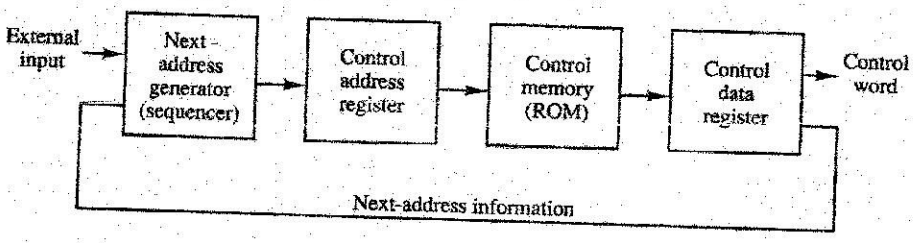
- ➢ *HORIZONTAL MICRO-INSTRUCTION:*
- ✓ *Construct a control word in which each bit represents a control signal.*
- ✓ *Now, add an address field to the control word, indicating the location of next control word to be executed if a certain condition is true.*
- ✓ *Add few bits to specify the condition*
- ✓ → *Horizontal Micro-instruction*
- ✓ *Horizontal micro-instruction is interpreted as follows:*
  - ✓ *To execute this micro-instruction, turn on all the control lines indicated by a 1 bit. Leave off all control lines indicated by a 0 bit. The resulting control bits will cause one or more micro-instruction in sequence.*
  - ✓ *If the condition indicated by the condition bits is false, execute the next micro-instruction in sequence.*
  - ✓ *If the condition indicated by the condition bits is true, execute the micro-instruction indicated by address field.*

- ➢ *VERTICAL MICRO-INSTRUCTION:*
- ✓ *A more compact micro-instruction which uses function codes rather than directly the control signals is Vertical Micro-instruction.*
- ✓ *This uses fewer bits when compared to horizontal micro-instruction. Hence, it is more compact at the expense of a small additional amount of logic and time delay required for decoding functional codes.*





➢ *MICRO-PROGRAMMED CONTROL ORGANIZATION*



- ✓ *Control memory is assumed to be ROM, which consists of a set of micro-instructions stored in it.*
- ✓ *CAR (Control Address Register) contains the address of the next micro-instruction to be read.*
- ✓ *When a micro-instruction is read from the control memory, it is transferred into CDR (Control Data Register). The Data register is sometimes called a Pipeline register.*
- ✓ *A portion of CDR connects to control signals emanating from CPU.*
  - ➔ *reading micro-instruction from control memory is same as executing that micro-instruction.*
- ✓ *Sequencer unit loads the control address register and issues a read command.*

signal.

Solution : Vertical Micro-instruction

* Horizontal & Vertical Micro-instructions

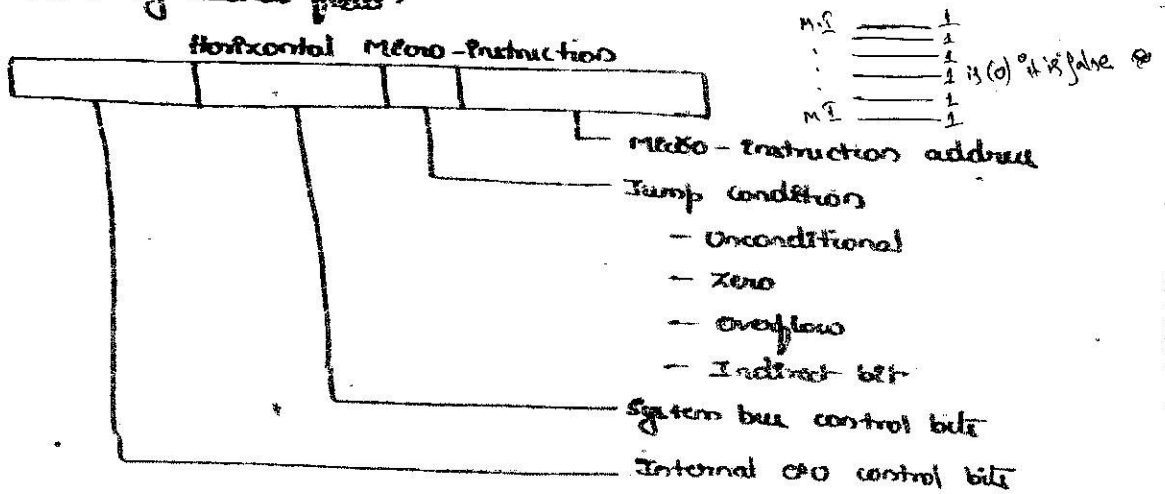
a) Horizontal Micro-instructions

→ 1) Construct a control word in which each bit represents a control signal.
2) Now, add an address field to the control word. Indicating the location of next control word to be executed if a certain condition is true
3) Add few bits to specify the condition

This is called Horizontal μ-instruction

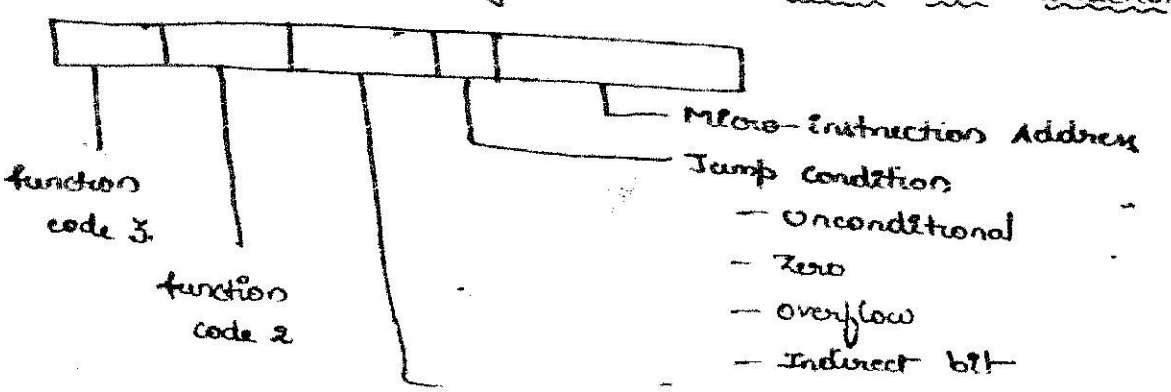
— Horizontal μ-instruction is interpreted as follows:

1) To execute this micro-instruction, turn on all the control lines indicated by 1 bit. Leave off all ~~all~~ control lines indicated by 0 bit. The resulting control signals will cause 1 or more micro-instruction in sequence.
2) If the condition indicated by the condition bits is false, execute the next micro-instruction in sequence.
3) If condition indicated by the condition bits is true, execute the micro-instruction indicated by address field.



b) Vertical Micro-instruction

— A more compact micro-instruction which uses function codes rather than directly the control signals is called Vertical Micro-instruction.

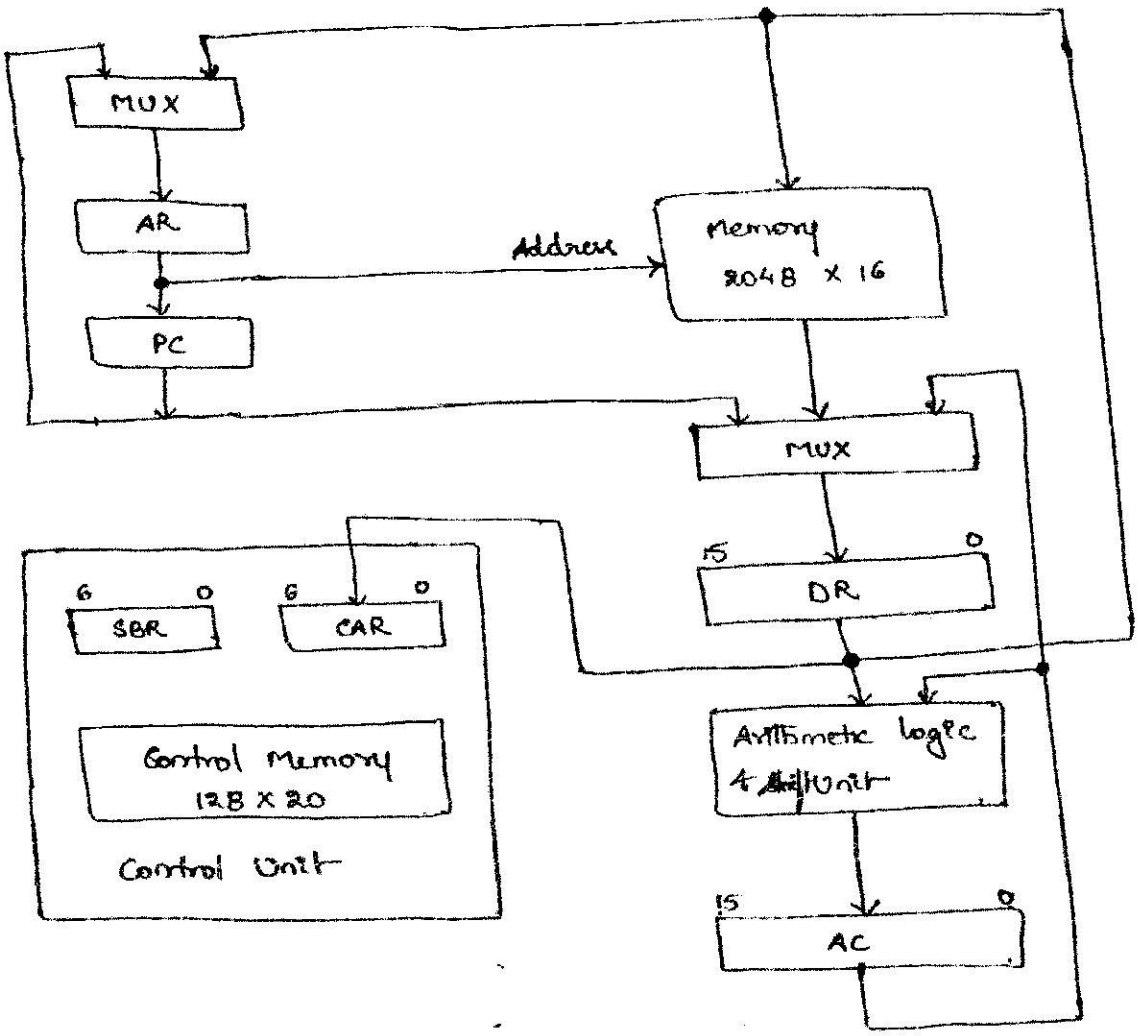


— This uses fewer bits when compared to horizontal micro-instruction. Hence, it is more compact at the expense of a small additional amount of logic & time delay required for decoding functional codes.

* Microprogram Example

— Computer Configuration

The block diagram of the computer is shown below



— It consists of 2 memory units : a main memory & control memory. Main Memory for storing instructions & data and control memory for storing the microprograms.

— Four registers are associated with the processor : PC, AR, DR and AC.

— Two registers are associated with control Unit : CAR, SBR

— Transfer of information among the registers in the processor is done through multiplexers rather than a common bus.

— DR can receive data from AC, PC or memory.

— AR can receive information from PC or DR.

— PC can receive information only from AR.

- ✓ ***The location of next instruction may be***
  - ✓ *one next in sequence*
  - ✓ *located somewhere else in control memory*
  - ✓ *function of external i/p conditions*
- ✓ *The next-address generator is sometimes called* ***Micro-program Sequencer.***
- ✓ ***Typical functions of micro-program sequencer are:***
  - ✓ *increment CAR by 1*
  - ✓ *loading into CAR an address from memory*
  - ✓ *transferring an external address into CAR*
  - ✓ *loading an initial address to start the control operations*

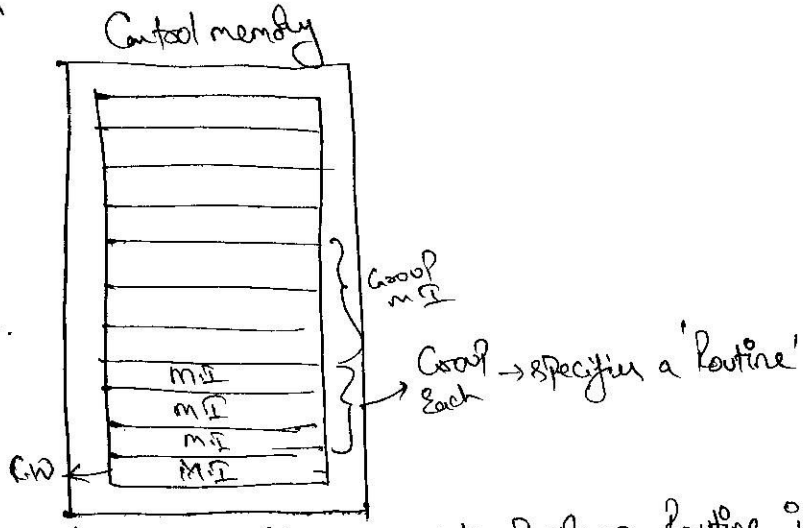
ADDRESS SEQUENCING

1. ***An initial address is loaded into the CAR when power is turned ON.***
   1. *Usually the first address of the 1st instruction that activates fetch instruction.*
   2. *Fetch routine is sequenced by incrementing CAR through rest of its micro-instructions.*
   3. *At the end of fetch phase, instruction is in IR.*
2. ***Now, routine for determine EA of operand should be executed.***
   1. *EA computation routine in control memory can be reached through a branch micro-instruction, which is conditioned on the status bits of the instruction.*
   2. *At the end, address of operand is available in MAR.*
3. ***Generate micro-instructions to execute the instructions fetched from memory.***
   1. *Each instruction has its own micro-program routine stored in a given location in control memory.*
   2. *The transformation from instruction code bits to an address in control memory where the routine is located is called MAPPING PROCESS.*
   3. *Once the routine is reached, micro-instructions in the routine are sequenced by incrementing CAR or depends on status bits and branch address.*
   4. *If micro-program calls a subroutine, we need to store the return address which is not possible in control memory which is ROM.*
4. ***Return to fetch routine is accomplished by an unconditional branch to the first address of fetch routine.***

→ It is an operand to an instruction which reference memory.

EA :- The address that is obtained by applying an specified indexing(d) indirect addressing rules to specified address.

i.e EA is used as Effect Address.



* Each Computer Instruction has its own micro program Routine in Control memory to ~~specify~~ its micro operations. that Executes instructions Execute.

* Steps during the execution of single Computer instruction are :-

(i) → ① Initial address loaded CAR when Power on.
② 1st instruction address Activate fetch operation of instruction
③ fetch Phase is sequence by incrementing (CAR)↑ through rest of M.I
④ at the End fetch phase, Instruction is in Instruction Reg (IR).

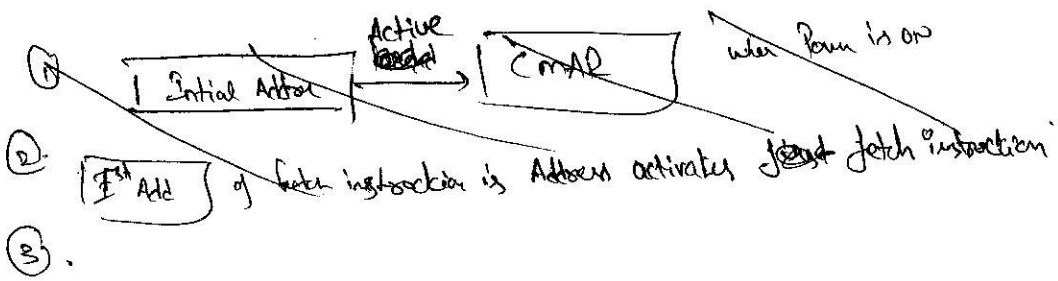
(ii) → Now Routine that determine effective Address of operand that should be executed.
① E.A Computation routine is in Control memory can be reached through a [branch (M.I)] which is conditional on [status bit] of instruction.
②. At the End address of operand is present in [MAR].
③ In a Control memory at given location Each instruction has its own micro Programmed routine stated.
④. The transformation from instruction codes bits to an address where the routine is located in Control memory is called mapping Process.
⑤. once the instruction is reached. the M.I in the in the routine are repeated by incrementing CAR depending upon status bits & branch Address.
⑥ the mi.Prog Calls the subroutine when needed to store the return address which is not possible in Control memory (i.e Rom).

Generate a M.I to Execute instruction

(iii) fetch from memory.

(iv) → Return to fetch routine is accomplished by an unconditional branch to the first address of fetch routine :-

- ✓ *In summary, address sequencing capabilities required in a control memory are:*
  - ✓ *Incrementing of CAR*
  - ✓ *Unconditional branch or conditional branch, depending on status bit conditions.*
  - ✓ *A mapping process from the bits of the instruction to an address for control memory.*
  - ✓ *Facility for subroutine call and return.*



The diagram shows 4 different ways from which CAR receives address

(1) Incrementer → Increments content of CAR by '1' to select m.i sequence
(2) Branching → Acived by specifying branch Address in one field of m.i
↳ Conditional branching is achived by using part of m.i to specify status bit in order to determine condition.
(3) External Address → is transfered to CAR via mapping logic circuit
(4) Return → address of subroutine is stored in special Reg whose value is then used when the µprogr wishes to return from subroutine.

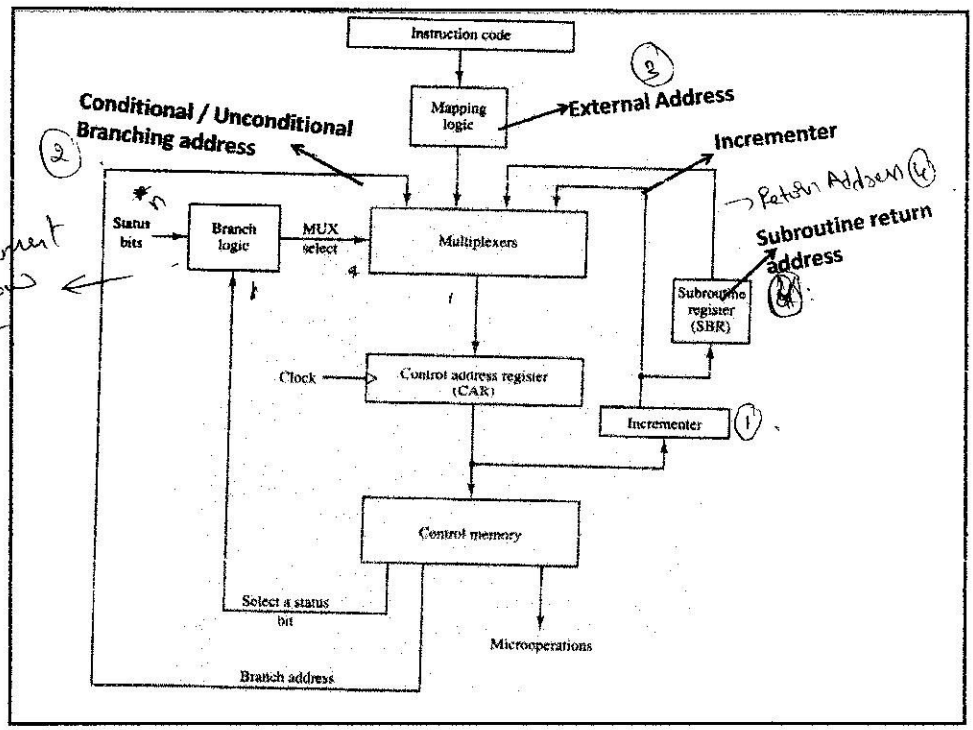
→ Conditional branching :-

⇒ Branch logic → making capabilities for decision making
→ Status conditions are special bits in system besides parameters
↳ Carryout of an adder
↳ sign bit of a number.
↳ mode bit of an instruction
↳ I/P & O/P status condition
→ status bits together with in micro operation specifies branch Address. and control the conditional branch decision generated in the branch logic.

→ implementing branch logic :-

① test specified condition branch to indicated address if condition is ok ; otherwise address register is incremented.
② Can be implemented using multiplexer.
③ branch logic o/p are select lines for multiplexer to select specified status bit condition. i.e 3-bits in m.i are used to specifies '1' out of 8 status bit condition

These 3-bits provide selection variable for multiplexer.
→ if selected status bit is 1 ; o/p of multiplexer is '1' ; otherwise '0'.
→ if 1 → o/p of multiplexer → generates control signal to transfer the branch address from CAR. (in form of micro instruction)
→ if 0 → o/p of MUX cause the address Reg to be incremented.



Status Condition parameters :- Carryout of Adder, sign bit of a number, mode bit of instruction. & o/p o/p - Status Condition

➢ *Conditional Branching*

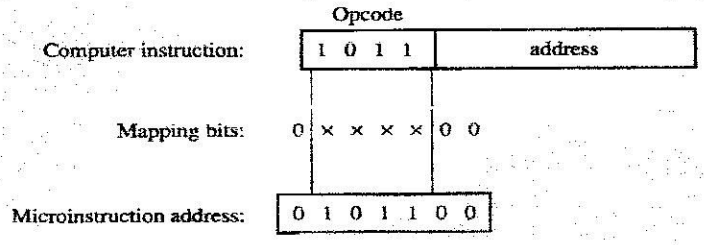
- ✓ *Branch logic provides decision-making capabilities in the control unit.*
- ✓ *Status conditions are special bits in the system that provide parameter information such as carry-out of an adder, sign bit of a number, mode bits of an instruction and i/p or o/p status conditions. Information in these bits are tested and actions are initiated based on their condition, whether 0 or 1.*
- ✓ *Status bits together with the field in the micro-instruction that specifies a branch address, control the conditional branch decisions generated in the branch logic.*
- ✓ ***Implementation of Branch Logic:***
  - ✓ *Test the specified condition and branch to the indicated address if the condition is met; otherwise, the address register is incremented.*
  - ✓ *Can be implemented using Multiplexer. Suppose that there are 8 status bit conditions in the system. Three bits in the micro-instruction are used to specify any one of 8 status bit conditions. These 3 bits provide the selection variables for the multiplexer.*
    - ✓ *If selected status bit is 1, o/p of multiplexer is 1; otherwise it is 0.*
    - ✓ *A 1 o/p in the multiplexer generates a control signal to transfer the branch address from the micro-instruction into the CAR.*
    - ✓ *A 0 o/p in the multiplexer causes the address register to be incremented.*

Status bit = 1

An unconditional branch micro-instruction can be implemented by loading the branch address from control memory into CAR. This can be accomplished by fixing the value of one status bit at the input of multiplexer, so it is always equal to 1.

➢ *Mapping of Instruction*

- ✓ *A special type of branch used when a micro-instruction specifies a branch to the 1st word in control memory where a micro-program routine for an instruction is located.*
- ✓ *Status bits = bits in the operation code of the instruction* Status bits = Opcode
- ✓ *If opcode = 4bits and control memory has 128 words (7bits to address each word)*



- ✓ ***More General mapping rule: Use a ROM to specify the mapping function.***
  - ✓ *The bits in the instruction specify the address of a mapping ROM. The contents of the mapping ROM give the bits for the CAR.*
    - ➔ *Micro-program routine can be placed in any desired location in control memory.*

➢ *Subroutines*

- ✓ *Subroutines are programs that are used by other routines to accomplish a particular task.*
- ✓ *A subroutine can be called from any point within the main body of a micro-program.*
- ✓ *Micro-programs that use subroutines must have provision for storing the return address during a subroutine call and restoring the address during subroutine return.*
- ✓ ***Implementation:*** *Place the incrementer output from CAR into a subroutine register and branch to the beginning of the subroutine. The subroutine register can then become the source fro transferring the address for the return to the main routine.*
- ✓ *Best way to structure a register file that stores addresses for subroutines is to organize the registers in LIFO stack.*

opcode bits 4 the $2^4 = 16$ operations
(Memory Reference Instruction)

128 x 20 → 20 bits.

Accessed by 7 address bits ← 128 - words

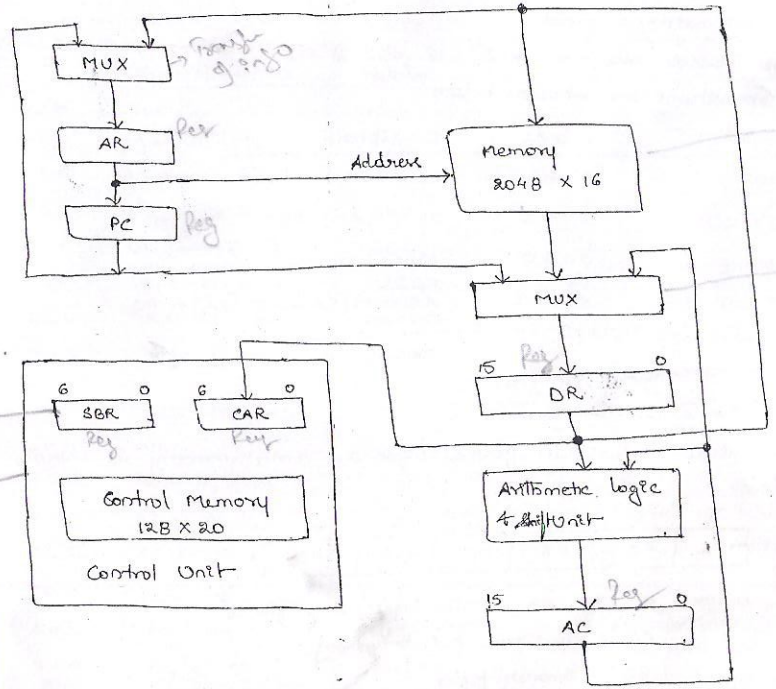
F_1, F_2, F_3 - 3 x 3 bits each.
CD - 2 bits
BR → 2 bits
Addfield → 7 bits.
20 bits.

— this uses fewer bits when compared to horizontal micro-instruction. Hence, it is more compact at the expense of a small additional amount of logic & time delay required for decoding functional codes.

* Microprogram Example

— Computer Configuration

The block diagram of the computer is shown below



— It consists of 2 memory units: a main memory & control memory. Main Memory for storing instructions & data and control memory for storing the microprograms.

— Four registers are associated with the processor: PC, AR, DR and AC

— Two registers are associated with control Unit: CAR, SBR

— Transfer of information among the registers in the processor is done through multiplexers rather than a common bus.

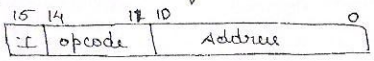
— DR can receive data from AC, PC or memory.

— AR can receive information from PC or DR.

— PC can receive information only from AR.

data from AC and DR and places the result in AC)

- Memory receives its address from AR. Input data written to memory come from DR, and data read from memory go only to DR.
- The computer instruction format is shown below



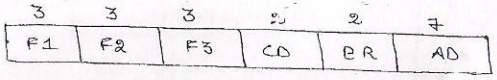
- It consists of 3 fields: an indirect bit, 4-bit opcode and 11-bit address field.
- Since opcode bits = 4 $\Rightarrow 2^4 = 16$ operations (memory-reference Instructions)
- 4 operations are shown below:

Symbol	opcode	Description
ADD	0000	AC ← AC + M[EA]
BRANCH	0001	If (AC < 0) Then (PC ← EA)
STORE	0010	M[EA] ← AC
EXCHANGE	0011	AC ← M[EA], M[EA] ← PC

EA = Effective address.

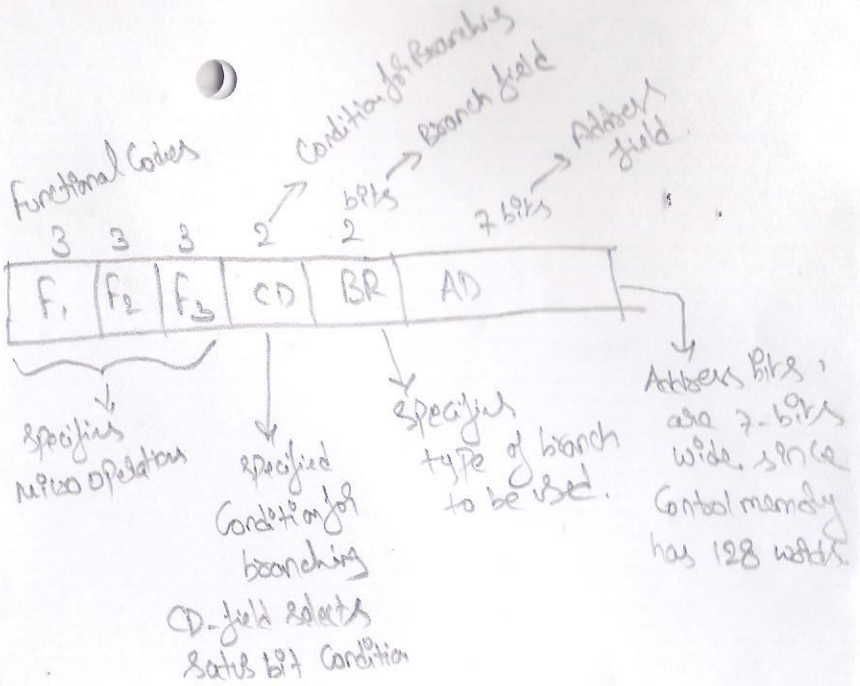
Micro-Instruction Format

- The micro-instruction format for the control memory is shown below.

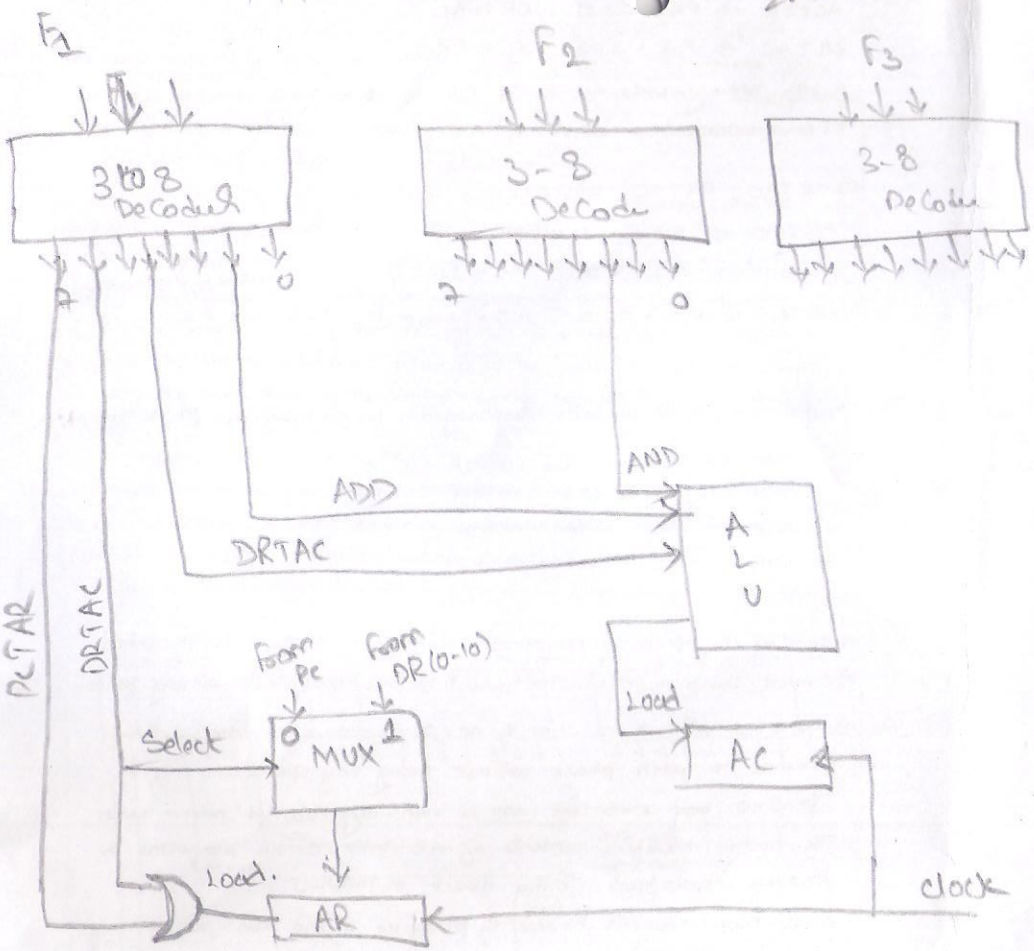


where F1, F2, F3 = micro-operation fields
CD = Condition for branching
BR = Branch field
AD = Address field

- F1, F2 and F3 specify micro-operations for the computer.
- CD field selects status bit conditions.
- BR field specifies the type of branch to be used.
- AD field contains a branch address. Address field is 7 bits wide, since the control memory has $128 = 2^7$ words.
- The micro-operations are subdivided into three fields of 3 bits each. The 3 bits in each field are encoded to specify seven distinct micro-operations ⇒ gives a total of 21 micro-operations as listed below:



MRoperation - 3-fields → gives total 21 Moperations



Decoding of micro operation field.

F1	Micro-operation	Symbol
000	None	NOP
001	AC ← AC + DR	ADD
010	AC ← 0	CLRAC
011	AC ← AC + 1	INCAC
100	AC ← DR	DRTAC
101	AR ← DR (0-10)	DRTAR
110	AR ← PC	PCTAR
111	M[AR] ← DR	WRITE

F2	Micro-operation	Symbol
000	None	NOP
001	AC ← AC − DR	SUB
010	AC ← AC ∨ DR	OR
011	AC ← AC ∧ DR	AND
100	DR ← M[AR]	READ
101	DR ← AC	ACTDR
110	DR ← DR + 1	INCDR
111	DR(0-10) ← PC	PCTDR

F3	Micro-operation	Symbol
000	None	NOP
001	AC ← AC ⊕ DR	XOR
010	AC ← $\overline{AC}$	COM
011	AC ← shl AC	SHL
100	AC ← shr AC	SHR
101	PC ← PC + 1	INCPC
110	PC ← AR	ARTPC
111	Reserved	

* All transfer-type micro-operations symbols use 5 letters.
  - 1st 2 letters – source (DR)
  - 3rd letter – TO
  - last 2 letters – Destination (AC)

Eg:- AC ← DR
= DRTAC
⇒ DR to AC.

Table: Symbols & binary code for micro operations field

- No more than 3 micro-operations can be chosen for micro-instruction 1 from each field. If fewer than 3 micro-operations are used, 1 or more fields will use the binary code 000 for no operation
- Eg:- DR ← M[AR], PC ← PC + 1.
  ⇒ F2 = 100 F3 = 101
  ⇒ Nine bits of the micro-operation fields will be F1 F2 F3: 000 100 101
- two or more conflicting micro-operations cannot be specified simultaneously
  Eg:- 010 001 000 – has no meaning since it specifies clear AC to 0 and subtract DR from AC at the same time.