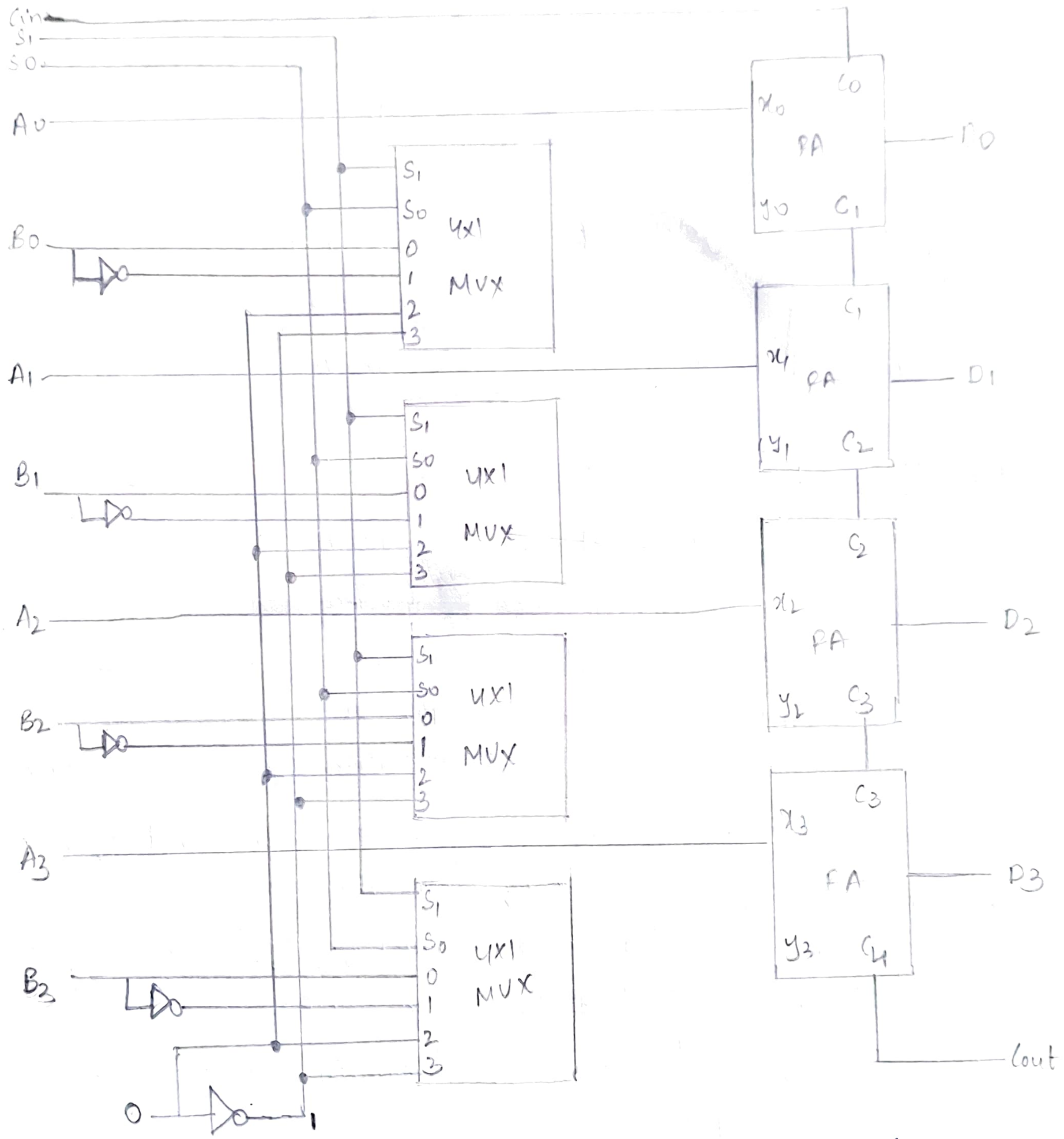
H. Kumudini
2407IA3242

COA Assignment - 1

1. Design a 4-bit Arithmetic Circuit that can perform the following operations based on the control inputs S1, S0 and Cin:
a. Addition, b. Addition with carry, c. Subtraction, d. Subtraction with borrow
e. Increment, f. Decrement, g. Transfer A.

Ans: 4-bit Arithmetic Circuit.



→ Basic Component of an arithmetic circuit is the parallel adder. By controlling the data inputs to the adder, it is possible to obtain different types of arithmetic operations.

→ By controlling the two select lines S1 & S0 making Cin 0 or 1. It is possible to generate the eight arithmetic microoperations.

Control inputs S_1, S_0, C_{in} for

	S_1	S_0	C_{in}	$D = A + Y + C_{in}$.
a) Adding:-	0	0	0	$D = A + B$
b) Add with carry:-	0	0	1	$D = A + B + 1$
c) Subtract :-	0	1	1	$D = A + \overline{B} + 1$
d) Subtract with borrow:-	0	1	0	$D = A + \overline{B}$
e) Increment A :-	1	0	1	$D = A + 1$
f) Decrement A:-	1	1	0	$D = A - 1$
g) Transfer A:-	1	0	0	$D = A$
	1	1	1	$D = A$.

2) Define the instruction cycle. Explain its significance in the content of program execution, focusing on how the CPU fetches, decodes and executes instructions. Draw the flowchart of the instruction cycle of a basic computer.

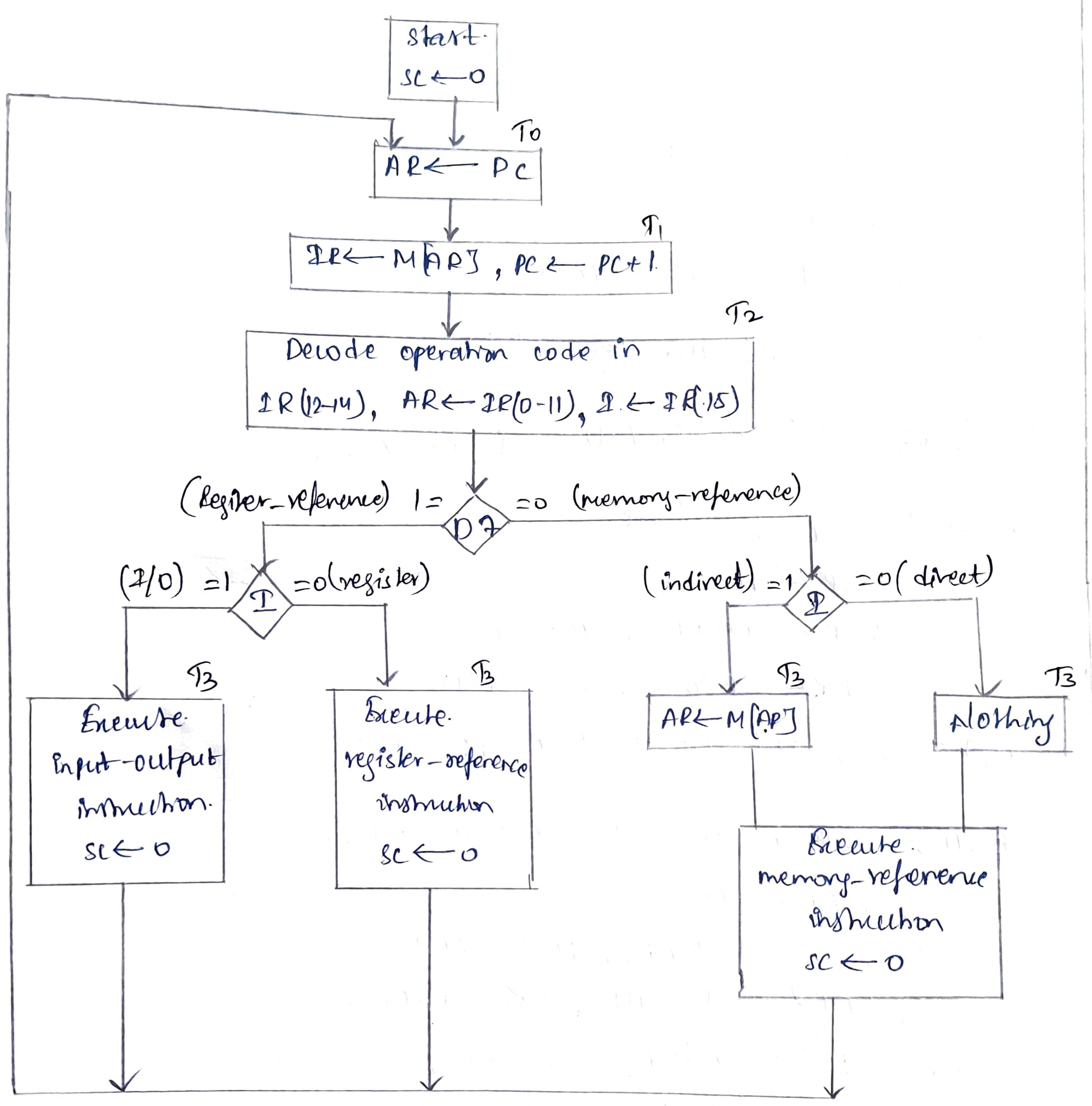
Ans:- Instruction Cycle :- A program residing in the memory unit of the computer consists of a sequence of instructions.

The program is executed in the computer by going through a cycle for each instruction. This is called Instruction cycle.

→ Each instruction cycle in turn is subdivided into a sequence of sub cycles or phases.

1) Fetch
2) Decode
3) Read EA (effective address)
4) Execute.

wchar For Instruction Cycle :-



Start
SC ← 0
T0
AR ← PC
T1
IR ← M[AR], PC ← PC+1
T2
Decode operation code in IR(12-14), AR ← IR(0-11), I ← IR(15)
(Register_reference) 1 =
D7
= 0 (memory-reference)
(I/O) = 1
I
= 0 (register)
(indirect) = 1
I
= 0 (direct)
T3
Execute input-output instruction. SC ← 0
T3
Execute register-reference instruction SC ← 0
T3
AR ← M[AR]
T3
Nothing
Execute memory-reference instruction SC ← 0

→ The timing signals T0 control unit fetches, T_1 → Fetches, T_2 → Decode.

→ The timing signal that is active after decoding is T_3.

→ The execution starts from T_3, and T_0, T_1, T_2 i.e fetching and decoding is common to all programs.

→ After T_3 the timing signals differ program to program.

3) For the following instructions, explain the sequence of microoperati
performed during its execution in a basic computer system.

a. CIR b. SPA c. BSA d. ISZ e. INP f. SKI.

Ans:

a. CIR :- circulate right

T_0 : AR ← PC

T_1 : IR ← M[AR] , PC ← PC+1

T_2 : AR ← IR(0-11), I ← IR(15)

$D_7I'T_3B_7$: AC ← shr AC, AC(15) ← E, E ← AC(0)

b. SPA :- skip if positive

T_0 : AR ← PC

T_1 : IR ← M[AR] , PC ← PC+1

T_2 : AR ← IR(0-11) , I ← IR(15)

$D_7I'T_3B_4$: If (AC(15)=0) then PC ← PC+1.

c. BSA :- Branch and save return Address.

T_0 : AR ← PC

T_1 : IR ← M[AR], PC ← PC+1

T_2 : AR ← IR(0-11) , I ← IR(15)

IT_3 : AR ← M[AR] or $I'T_3$: Nothing.

D_5T_4 : M[AR] ← PC , AR ← AR+1

D_5T_5 : PC ← AR , SC ← 0

d. ISZ :- Increment skip if zero.

T_0 : AR ← PC

T_1 : IR ← M[AR] , PC ← PC+1

T_2 : AR ← IR(0-11), I ← IR(15)

IT_3 : AR ← M[AR] or $I'T_3$: Nothing.

D_5T_4 : ~~M[AR] ← PC, AR ← AR+1~~ DR ← M[AR]

D_5T_5 : ~~PC ← AR, SC ← 0~~ DR ← DR+1.

D_6T_6 : M[AR] ← DR. , if (DR=0) then (PC ← PC+1)

INP :- Input Character.

T_0 : AR ← PC

T_1 : IR ← M[AR], PC ← PC+1.

T_2 : AR ← IR(0-11), I ← IR(15)

$D_7 I T_3 B_{11}$: AC(0-7) ← INPR, FGI ← 0.

f. SKI :- Skip on Input Flag.

T_0 : AR ← PC

T_1 : IR ← M[AR], PC ← PC+1

T_2 : AR ← IR(0-11), I ← IR(15)

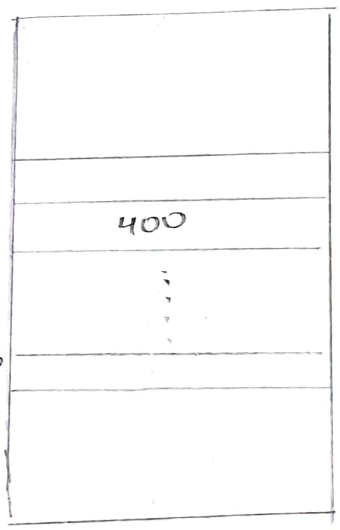
$D_7 I T_3 B_9$: If (FGI=1) then (PC ← PC+1)

4) An instruction is stored at location 300 with its address field at location 301. The addresses field has value 400. A processor register R_1 contains the number 200. Evaluate the effective address if the addressing mode of the instruction is (a) direct (b) immediate (c) Relative (d) Register Indirect (e) Index with R_1 as Index register.

sol:-

(a) direct, i.e the affective address of operand is the value present in the address field. i.e. equal to '400'.

Hence, ans = 400



(b) immediate, i.e the data is the value stored in the address field. Hence, EA is 301.

ans = 301

(c) Relative, i.e for relative we add the value in address field + next instruction address ⇒ 400+302 = 702

Hence, EA = 702

(d) Register index; For register index the EA will be the value of register itself. Hence, EA = 200.

(e) Index with R_1 as index register ⇒ for this we add the value of Register 'R_1' and the value stored in address field i.e 400

⇒ 400 + 200 = 60

Hence, EA = 600.

5) Discuss the impact of using different address instruction formats (zero, one, two, three-address) in the execution of arithmetic expression such as x = (~~A~~A+B) * (C+D).

Ans:- Zero Address:- A stack-organized computer doesn't use an address field for the instructions ADD and MUL.
→ PUSH and POP instructions, need an address field to specify the operand.

X = (A+B) * (C+D)

→ The name zero-address given to this type of comp. because of the absence of an address field in the computational instructions.

PUSH	A	TOS ← A
PUSH	B	TOS ← B
ADD		TOS ← (A+B)
PUSH	C	TOS ← C
PUSH	D	TOS ← D
ADD		TOS ← (C+D)
MUL		TOS ← (C+D) * (A+B)
POP	X	M[X] ← TOS.

One-Address:- One-Address instructions use an implied Accumulator (AC) Register for all data manipulation.
→ For mul & div there is a need for second Register.
→ We neglect 2nd register & assume, the AC contains result of all operations.

LOAD	A	AC ← M[A]
ADD	B	AC ← AC + M[B]
STORE	T	M[T] ← AC
LOAD	C	AC ← M[C]
ADD	D	AC ← M[D] + AC
MUL	T	AC ← AC * M[T]
STORE	X	M[X] ← AC.

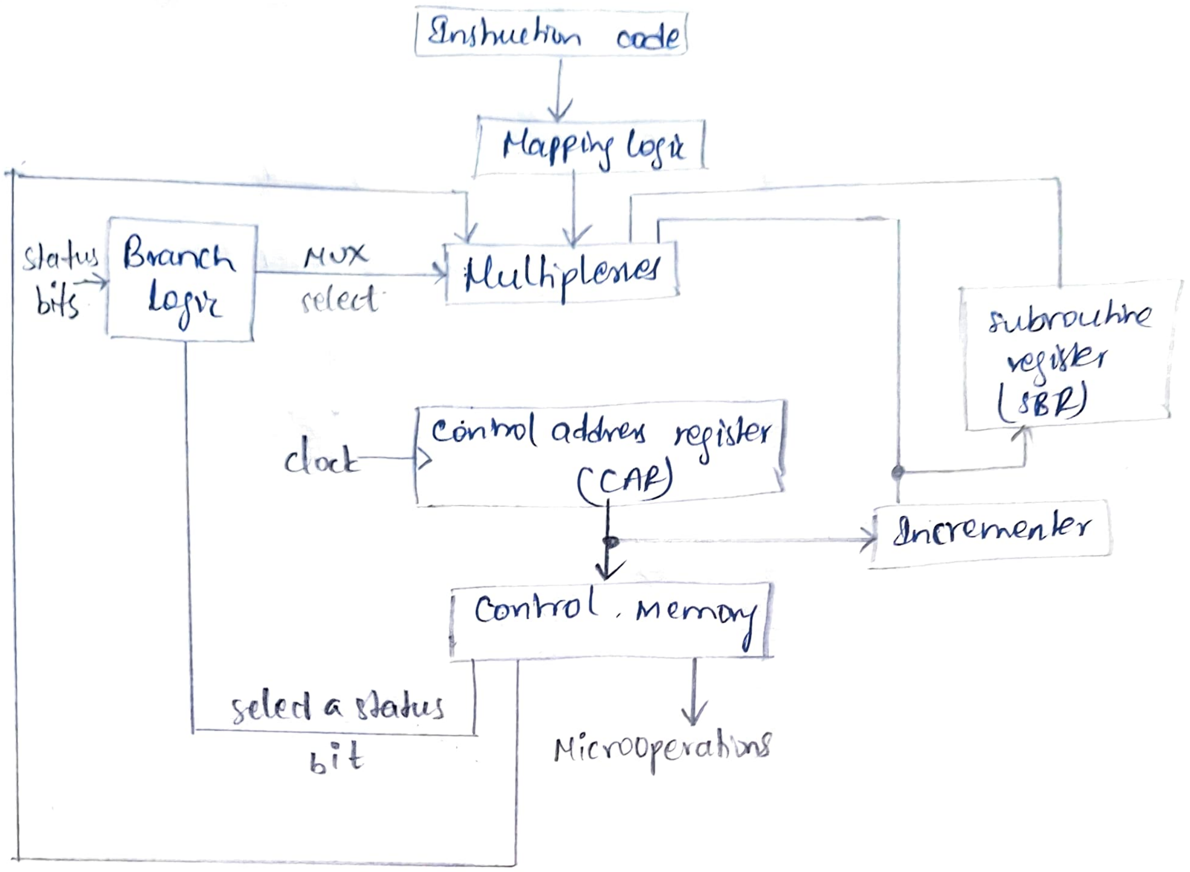
Two-Address:- These are most common in commercial computers.

MOV	R1, A	R1 ← M[A]
ADD	R1, B	R1 ← R1 + M[B]
MOV	R2, C	R2 ← M[C]
ADD	R2, D	R2 ← R2 + M[D]
MUL	R1, R2	R1 ← R1 * R2.
MOV	X, R1	M[X] ← R1

Three-Address:- Computers with three-address instruction formats can use each address field to specify either a processor register or a memory operand.

ADD	R1, A, B	R1 ← M[A] + M[B]
ADD	R2, C, D	R2 ← M[C] + M[D]
MUL	X, R1, R2	M[X] ← R1 * R2.

the help of block diagram, explain the process of address sequencing
detail.



~~Steped~~ Process of Address Sequencing

~~1.~~ Increment CAR (sequential Execution)
- Next memory location → next micro instruction.
- CAR ← CAR+1

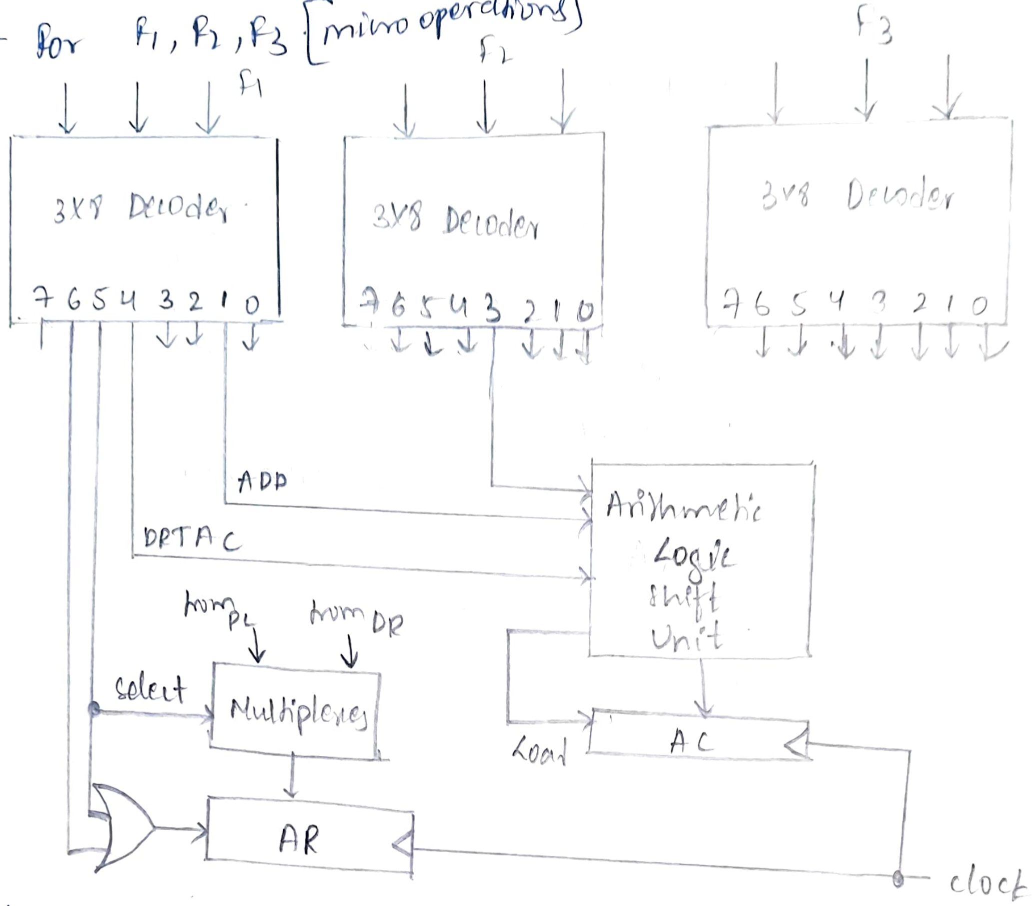
2. Branching (unconditional or conditional jump)
- If branch specified, CAR updated with that branch and
- condinal branches depend on status flag.

3. Mapping from Instruction Opcode.
- At the start of executing a machine instruction, opcode field is used to determine starting address.
- CAR ← Mapping(opcode)

4. Subroutine calls and Returns
- Microprograms may call subroutines.
- Current CAR pushed onto stack when returning popped back.

7) Design a microprogrammed control unit for a CPU that suppor
decode, and execute of ADD, BRANCH, STORE, EXCHANGE instruction

Ans:- for F_1, F_2, F_3 [micro operations]



Fetch:-

AR ← PC

DR ← M[AR], PC ← PC+1

AR ← DR (0-10), CAR(2-5) ← DR(11-14), CAR(0,1,6) ← 0

→ The fetch routine needs there three microinstructions, which are placed in control memory at addresses 64, 65, 66.

Assembly Level language:-

```
FETCH :   ORG 64
          PCTAR          U    JMP    NEXT
          READ, INCPC    U    JMP    NEXT
          DRTAR.         U    MAP
```

Micro Instruction Format:-

3	3	3	2	2	7
F1	F2	F3	CD	BR	AD

ion field :- CD (consists two bits)
we will use the symbols U, I, S, Z for the four status bits.

CD	Condition	Symbol	Comments
00	Always=1.	U	unconditional branch
01	DR(15)	I	Indirect address bit
10	AC(15)	S	sign bit of AC
11	AC=0	Z	zero value in AC

Branch field :- BR.

BR	Symbol	function
00	JMP	CAR ← AD if condition=1. CAR ← CAR+1 " " =0
01	CALL	CAR ← AD, SBR ← CAR+1 if CD=1. CAR ← CAR+1 if CD=0
10	~~RBT~~ RET	CAR ← SBR.
11	MAP	$CAR_{(2-5)}$ ← DR(11-14), C(0,1,6) ← 0

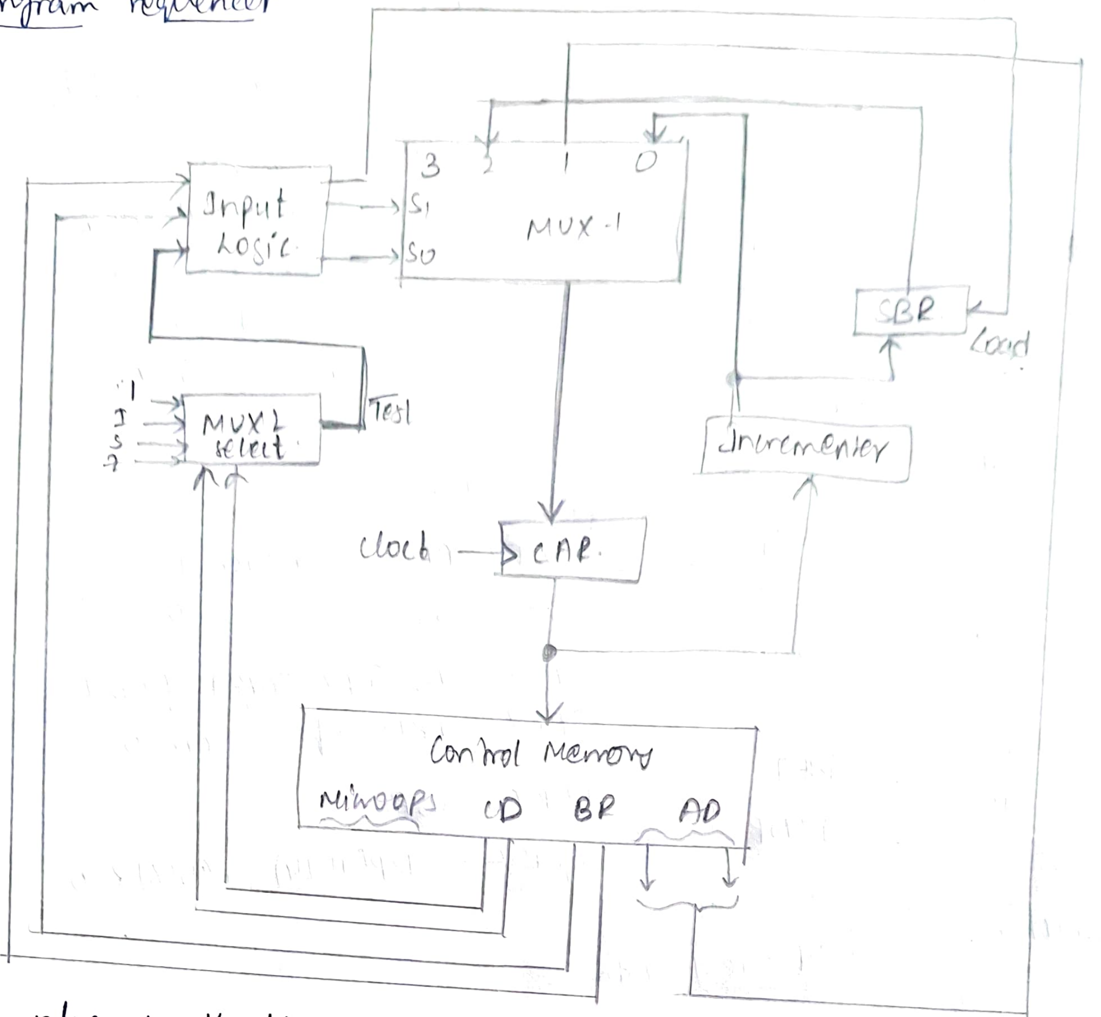
Instruction :- 15 14 11 10 0

$\boxed{I \mid \text{Opcode} \mid \text{Address}}$

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] ← AC.

$\longrightarrow$

Micro program Sequencer



8) The value of a float type variable is represented using single-precession 32-bit floating point format IEEE-754 standard. $X = -27.65$. the representation of X in hexadecimal notation is.

Sol:- $X = -27.65$ $\boxed{S=1}$

27 ⇒ 11011 · 101

$0.65 \times 2 = 1.25 \to 1$
$0.25 \times 2 = 0.5 \to 0$
$0.5 \times 2 = 1.$

Normalization $= (-1)^1 \times 1.1011101 \times 2^4$

$e = 4$

$E = 4 + \boxed{127}$ ⟶ IEEE-754

$E = 131$

131 ⇒ 10000011 = E.

$-27.625 = $ | 1 | 10000011 | 1011101 00...0 |

$\underbrace{1}_{S}$ 1 bit, $\underbrace{10000011}_{E}$ 8 bit, $\underbrace{101110100\cdots0}_{M}$ 23 bit

Hexadecimal = $\underbrace{1100}_{C}$ $\underbrace{0001}_{1}$ $\underbrace{1101}_{D}$ $\underbrace{1101}_{D}$ $\underbrace{00\cdots0}_{0000}$

-27.625 ⇒ 0X C1DD 0000.