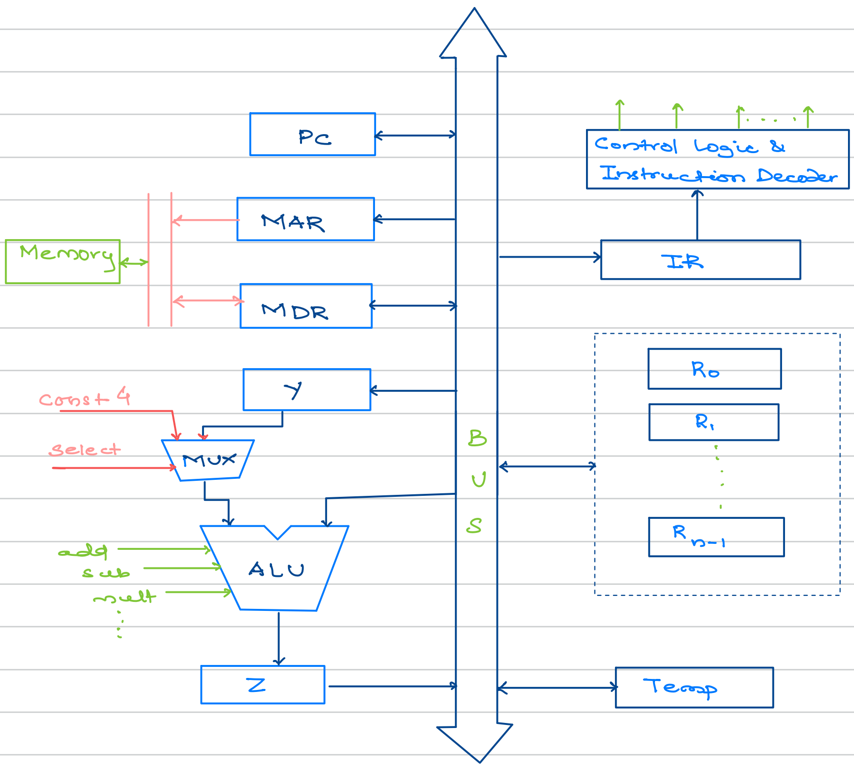
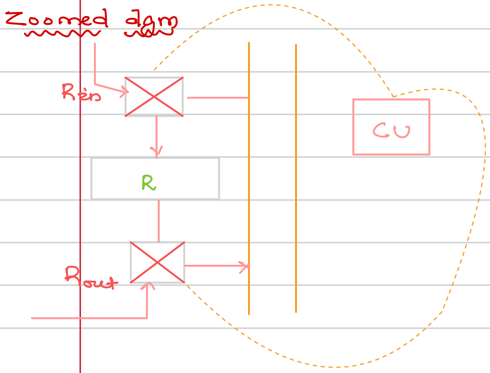
Basic processing unit



Note :- BUS is used to transfer data
In one clk pulse, the bus can hold only one data / information in a particular clk pulse.



ADD R1, R2, R3 → Memory Instr → 1
ADD (R1), (R2) → Memory Instr → 4

→ With few exceptions, any Assembly Language Instruction can be carried out by performing any of the /or combination of the folln operations :-

1. Transferring the content of 1 CPU register to another register (Register Transfer Operation)

   eg:- $R_1 \leftarrow R_2$

2. Performing arithematical & logical operations betn the content of 2 processor register & storing the content of another register (ALU operation)

   eg:- $\boxed{R_3} \leftarrow [R_2] + [R_1]$

3. Transferring the content of a memory location to the CPU register (Memory Read Operation)

   eg:- LOAD A, R_1

4. Transferring the content of a CPU register to a memory location (Memory Write Operation)

   eg:- STORE R_1, A

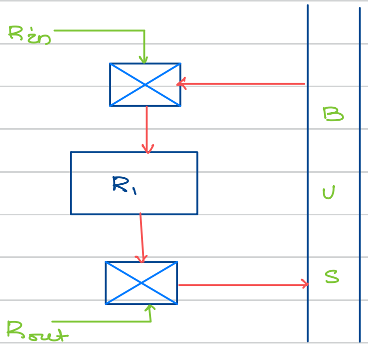
5. Condition control or Branch Instruction

27-01-2025

1. Register Transfer Instr :-

MOVE R_1, R_2

R_1 out , R_2 in



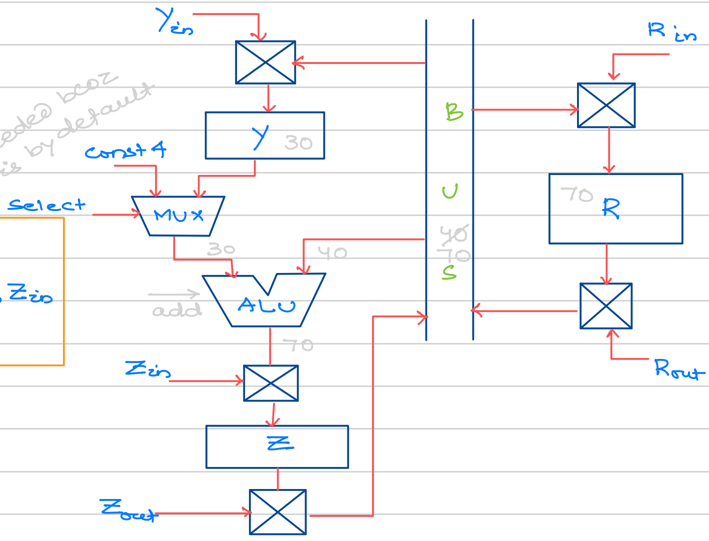
2. ALU Operations

(30) (40)
ADD R_1, R_2, R_3

+

1. R_1 out , Yin , READ
2. R_2 out , Select Y , ADD , Zin
3. Zout , R_3 in

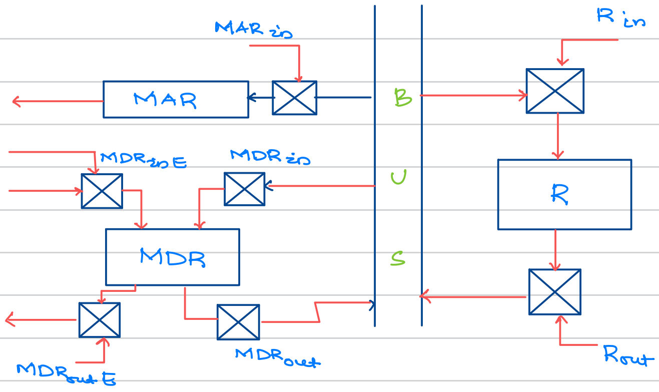
not needed bcoz it is by default



3. Memory Read Operations :-

LOAD (R1) , R2

1. R1 out , MAR in
2. MDRin E , WMFC
3. MDRout , R2 in



MARout is by default on.

WMFC → Wait for Memory Function Completed.
→ it waits for memory func to get completed.

4. Memory Write Operations :-

STORE R1, (R2)
STORE R1 , A

1. R2 out , MAR in
2. R1 out , MDR in , MDR outE , WRITE
3. WMFC

* After completing the process , write `END'.

ADD (R3) , R1

1. Fetch (read the instruc from memory to IR)

(i) PCout , MARin , Const 4 , ADD , Zin — whenever there is arithematic operation

(ii) Zout , PCin , Yin , MDRinE , WMFC
↳ not compulsory
↳ whenever there is read/write operation

(iii) MDRout , IRin

(iv) R3out , MARin , READ → if we write Read then next line write WMFC compulsory

(v) R1out , Yin , MDRinE , WMFC

(vi) MDRout , Select Y , ADD , Zin
↳ always a ALU operation followed by Zin

(vii) Zout , R1in

* (viii) END

(Q) ADD R3 , (R1)

(i) PCout , MARin , READ , Const 4 , ADD , Zin

(ii) Zout , PCin , Yin , MDRinE , WMFC

(iii) MDRout , IRin

(iv) R1 out , MARin , READ

(v) R3out , Yin , MDRinE , WMFC

(vi) MDRout , Select Y , ADD , Zin

(vii) R1 out , MARin , WRITE

(viii) Zout , MDRin , MDR

*imp * (ix) END

Q → ADD (R3), (R1)

1. PC_{out}, MAR_{in}, READ, Const 4, ADD, Z_{in}
2. Z_{out}, PC_{in}, Y_{in}, MDR_{inE}, WMFC
3. MDR_{out}, IR_{in}
4. $R3_{out}$, MAR_{in}, READ, MDR_{inE}, WMFC
5. MDR_{out}, Y_{in}
6. $R1_{out}$, MAR_{in}, READ, MDR_{inE}, WMFc
7. MDR_{out}, Select Y, ADD, Z_{in}
8. $R1_{out}$, MAR_{in}, WRITE
9. Z_{out}, MDR_{in}, MDR_{outE}, WMFC
10. END

Q- MOVE (R1)+, R2

i) PC_{out}, MAR_{in}, READ, Const 4, ADD, Z_{in}
ii) Z_{out}, PC_{in}, Y_{in}, MDR_{inE}, WMFC
iii) MDR_{out}, IR_{in}
iv) $R1_{out}$, MAR_{in}, Read
v) WMFC, MDR_{outE}, R_2in
vi) $R1_{out}$, Y_{in}, SELECT 4, ADD, Z_{in}
vii) Z_{out}, $R1_{in}$

Q- MOVE -(R1), R2

1
2
3
4. $R1_{out}$, Y_{in}, Select 4, SUB, Z_{in}
5. Z_{out}, MAR_{in}, READ
6. MDR_{inE}, WMFc
7. MDR_{in}, $R_{2}in$
8. End

Branch Instructions :-

1. Unconditional Branch
2. Conditional Branch

5000	________
5004	________
5008	________
5012	Increment R1
5016	Branch > 0 −16
5020	________

Unconditional Branch

1. PC_{out} , MAR_{in} , READ , Select 4 , ADD , Z_{in}
2. Z_{out} , PC_{in} , Y_{in} , MDR_{inE} , WMFC
3. MDR_{out} , IR_{in}
4. offset field of IR_{out} , Select Y , ADD , Z_{in}
5. Z_{out} , PC_{in} , END

Conditional Branch

1. PC_{out} , MAR_{in} , READ , Select 4 , ADD , Z_{in}
2. Z_{out} , PC_{in} , Y_{in} , MDR_{inE} , WMFC
3. PC_{out} , IR_{in}
4. offset field of IR_{out} , Select Y , ADD , Z_{in} if
   $R1 > 0$ else END
5. Z_{out} , PC_{in}, END

MUL (R1) ,(R2)

i) PC_{out} , MAR_{in} , READ , Const 4 , ADD , Z_{in}
ii) Z_{out} , PC_{in} , Y_{in} , MDR_{inE} , WMFC
iii) MDR_{out} , IR_{in}
iv) $R1_{out}$, R = B , MAR_{in} , READ
v) MDR_{inE} , WMFC
vi) MDR_{outB} , R = B , $R3_{in}$
vii) $R2_{out}$, R = B , MAR_{in} , READ
viii) MDR_{inE} , WMFC
ix) $R3_{outA}$, Select A , MDR_{outB} , MUL , MDR_{in}
x) R_{2outB} , R = B , MAR_{in} , WRITE
xi) MDR_{outE} , WMFC
xii) END

Q) MOVE (R1) + , R2
Q) MOVE -(R1) , R2
Q) BRANCH > 0 #200

$\Big\}$ H.W

MOVE (R1) + , R2

i) PC_{out} , MAR_{in} , READ , Const 4 , ADD , Z_{in}
ii) Z_{out} , PC_{in} , Y_{in} , MDR_{inE} , WMFC
iii) MDR_{out} , IR_{in}
iv) $R1_{out}$, MAR_{in} , Read
v) WMFC , MDR_{outE} , R_2in
vi) $R1_{out}$, Yin , SELECT 4 , ADD , Zin
vii) Z_{out} , $R1_{in}$
viii) END

MOVE -(R1) , R2

i> PCout , MARin , READ , Const 4 , ADD , Zin
ii> Z_{out} , PCin , Yin , MDR_{inE} , WMFC
iii> MDRout , IRin
iv> R1out , Yin , Select 4 , SUB , Zin
v> Zout , MARin , READ
vi> MDRinE , WMFC
vii> MDRin , R_2in
viii> End

Branch >0 #200

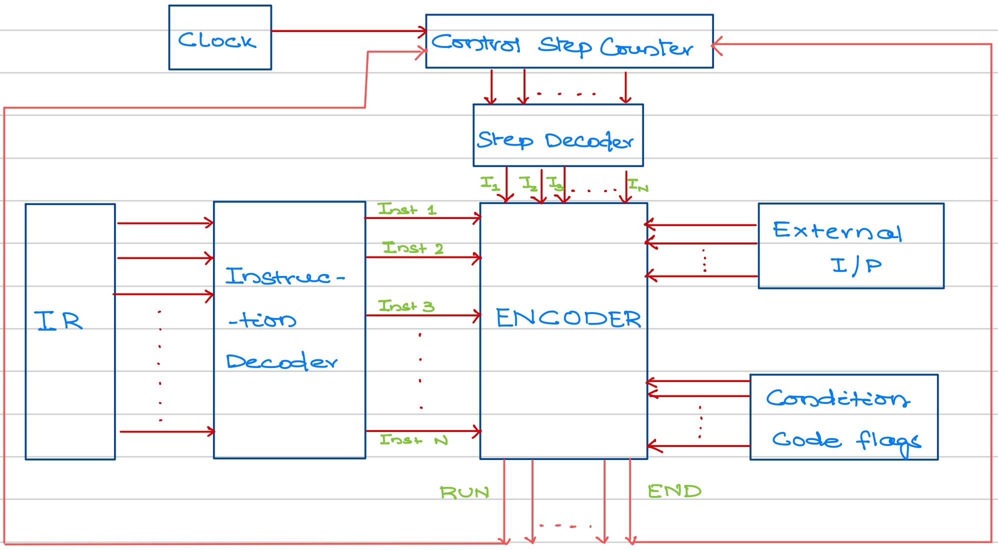
i> PCout , MARin , READ , Const 4 , ADD , Zin
ii> Z_{out} , PCin , Yin , MDR_{inE} , WMFC
iii> MDRout , IRin
iv> if acc > 0 ; Address out , PCin (#200)
v> if acc ⩽ 0 ; no instruction
vi> end

Control Unit Design

Control unit design

1. Hardware Control CU
2. Microprogrammed Control CU

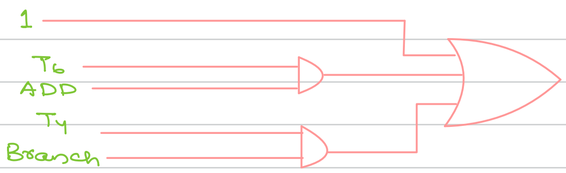
Hardware Control CU



∴ ⟶ = bus

Draw for Zin

$Z_{in} = 1 + T_6 . ADD + T_4 . Z_{in} + ...$



Draw for END.

$END = T_7 . ADD + T_5 . BRANCH + (T_5 . N + T_5 . \bar{N})\,CBR$

↓
Conditional
Branch
Instruction

A hardwired CPU uses 10 control signals S1 to S10; in various time steps T1 to T4 to implement 4 instruction I1 to I4 as shown below.

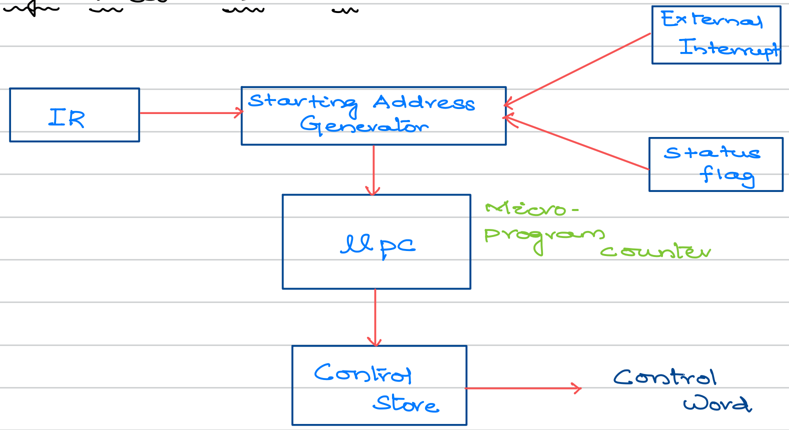
	T1	T2	T3	T4
I1	S1, S3, S5	S2, S4, S6	S1, S7	S10
I2	S1, S3, S5	S8, S9, S10	S5, S6, S7	S6
I3	S1, S3, S5	S7, S8, S10	S2, S6, S9	S10
I4	S1, S3, S5	S2, S6, S7	S5, S10	S6, S9

$S1 = T_1 + (I_1 . T3)$

$S9 = (I_2 . T_2) + (I_3 . T_3) + (I_4 . T_4)$

(option D)

Microprogrammed Control CU :-



Control Word

# It is a word whose individual bits represents various control signals. (represented in 1's & 0's)

# A control sequence of instructions (control word) constitute the microroutine for the instructions.

# Individual control word in microroutine represents a microinstruction It is represented by a string of 1's & 0's.

# The microroutine for all instructions in the instruction set of a processor are stored in the control store.

# Microprogram control is used to read the control word sequentially from the control store. The μpc is automatically incremented by the clock so that successive microinstruction can be read from control store.

ADD (R3) , R1

(i) PC_{out} , MAR_{in} , Const 4 , ADD , Z_{in}

(ii) Z_{out} , PC_{in} , Y_{in} , MDR_{inE} , WMFC

(iii) MDR_{out} , IR_{in}

(iv) R_3 out , MAR_{in} , READ

(v) R_{1out} , Y_{in} , MDR_{inE} , WMFC

(vi) MDR_{out} , select Y , ADD , Z_{in}

(vii) Z_{out} , R_{1in}

(viii) END

Control Word / Clock	PC_{in}	PC_{out}	MAR_{in}	MDR_{in}	MDR_{out}	MDR_{inE}	MDR_{outE}	Y_{in}	Select 4	Select Y	ADD	Z_{in}	Z_{out}	R_{1in}	R_{1out}	R_{3in}	R_{3out}	IR_{in}	IR_{out}	WMFC	END	READ	WRITE
1	0	1	1	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	1	0
2	1	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0
3	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
4	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
5	0	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0
6	0	0	0	0	1	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0