6/3/19
Wednesday

PAGE NO.
DATE: / /

Unit - 3

Control Unit Design

Current Instruction's state is PSW

PSW
→ Program Status word (PSW)

Saves the current instruction and provides the services to the intrupts that occur

Instruction Cycle ⇒ The time taken by the inst. to fetch from the memory to execute.

OPcode	Address	Mode

① Fetch the instruction

PC+1 if 1 bit inst.
PC+4 if 4 bits inst.

(PC) ⇒ content / Address of the PC

IR ← ((PR))

② Decode the instruction

ADD R_1, R_2, R_3

$R_1 \leftarrow R_2 + R_3$

In this we don't have to refer the memory it is stored in registers.

Load A, M[100]

A ← M[100]

Store

In this we have to refer memory.

(3) Execute the instruction

(4) ~~Stoo~~ Store

Instruction cycle is made of machine cycle

Machine cycle is used for instruction fetch

Machine cycle is made of clock cycle

~~Instruction~~ fet
Machine cycle is made up of 4 clock cycle for making instruction fetch.

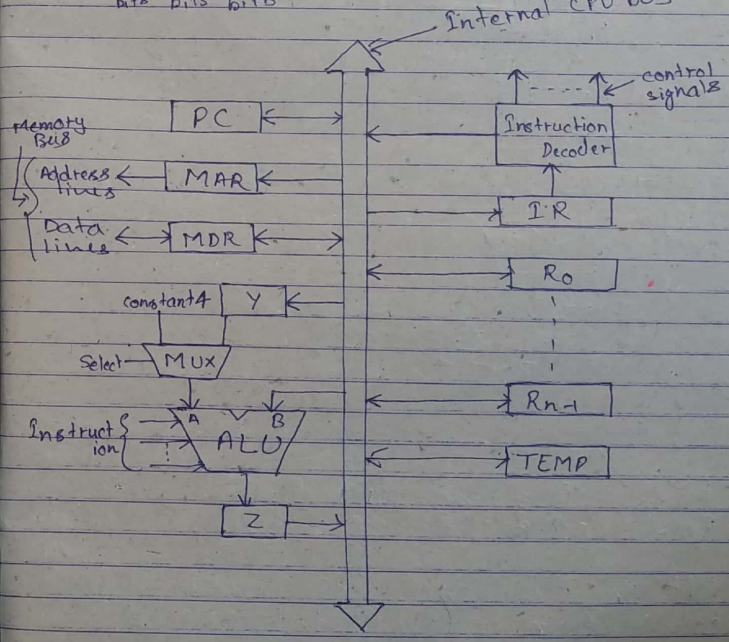
~~Fc~~ = Ma
Inst. cycle = Machine cycle

Inst. cycle = Instr. cycle + Read cycle

Execution of an Instruction

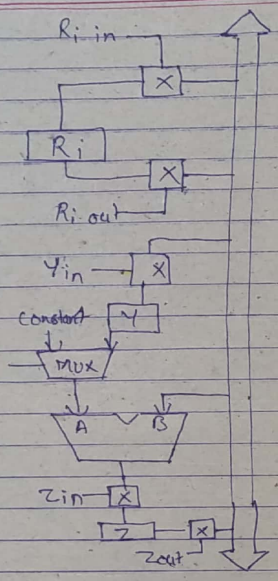
WORD, BYTE

2 4 8
16 32 64
bits bits bits



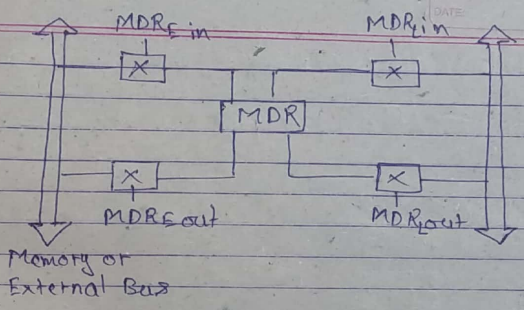
- Y, Z, TEMP are temporary registers these registers are used for temporary use
- Constant

R_i are control signals



R_i in Bus se data R_i me a raha hai
R_i out R_i se data Bus me ja raha hai
MDR is the one from which data comes from internal and external bus.

MDR_E in → Local → MDR in Internal Bus



$R_1 \leftarrow R_2 + R_3$

R_2 out, Y_{in}
R_3 out, Y_{in}, Add, Z_{in} (R_3 out comes directly in B)
Z_{out}, R_{1in}

WMFC → Waiting for Memory Function Complete

If R_1 se instruction read karna hai then

① R_1 out, MAR in, Read
② MDR_E in, WMFC
③ MDR out, R_2 in

} Read

① R_1 out, MAR in, write
② MDR_E in, WMFC
③ MDR_E out, R_2 in

} write

ADD (R3), R1

(R2) ← (R2) + R1

Add (R3) R1

Instruction fetch

Add R1, R2 $R_1 \leftarrow R_1 + R_2$

Add (R3) R1 ⇒ $R_1 \leftarrow R_1 + (R_3)$

1. PCout, MARin, Read, select4, Add, Zin
2. Zout, PCin, WMFC
3. MDRout, IRin
4. R3out, MARin, Read
5. R1out, Yin, WMFC
6. MDRout, selectY, Add, Zin
7. Zout, R1in

PCout is in now internal bus

Mem. k kis loc se read or write kna hai wo MAR pass jayega

1 word 64 bits = 8 byte

1 word 32 bit = 4 byte

Select4 means we are adding PC+4

- We are waiting bez we are fetching instruction from memory to overcome this use cache memory.

Subtract R1, (R2)+

$R_1 \leftarrow R_1 - (R_2)$

$(R_2) \leftarrow (R_2) + 1$

(R2)+ means first use the address of R2 and then add is

1. PCout, MARin, Read, select4, Add, Zin
2. Zout, PCin, WMFC
3. MDRout, IRin
4. R2out, MARin, Read
5. R1out, Yin, WMFC
6. MDRout, selectY, subtract, Zin
7. Zout, R1in
8. MAR R2out, select1, Add, Zin
9. Zout, R2in

Yin is the 1 input of MUX

In this R2out directly comes as the input of ALU

For 1 step this will execute in 1 clock cycle

Hardwired Control

T's are showing that in which clock cycle we are in which phase.

I → Fetch
2 → Decode
3 → Execute
4 → Store

Hardwired inputs of control signal generator

T_1 → is the clock pulse or clock cycle

↳ Whenever it refers to memory whether it is reading or writing it takes 3 clock cyle

1 instruction cycle = 1 M/c cycle

Result is not 0

- for executing any instruction the processor must generate control signals in proper sequency. Computer designer used to generate these signals by

① Hardwired control
② Microprogrammed control.

In hardwired control an encoder (control signal generator) is used to generate various control signals. The encoder gets its input from 4 different modules

(i) Content of the control step counter
(ii) Content of the instruction register.
(iii) Content of the conditions code flag
(iv) External input signal such as interupts request and MFC (Memory function complete)

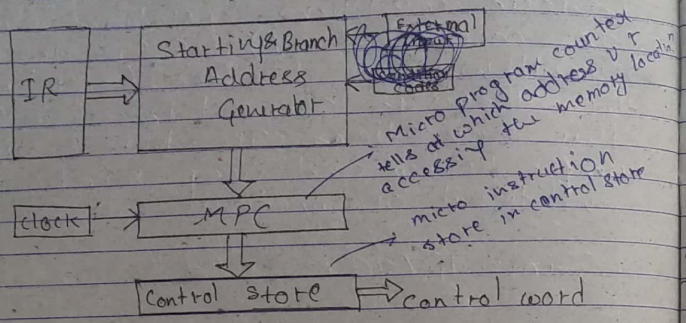
The step decoder provide a separate signal line ~~for~~ for each step or clock period. Similarly the output of the instruction decoder consist of seperate line for each machine instruction. For any instruction loaded in IR one of the output lines through INm is set to 1 and all other lines is set to 0.

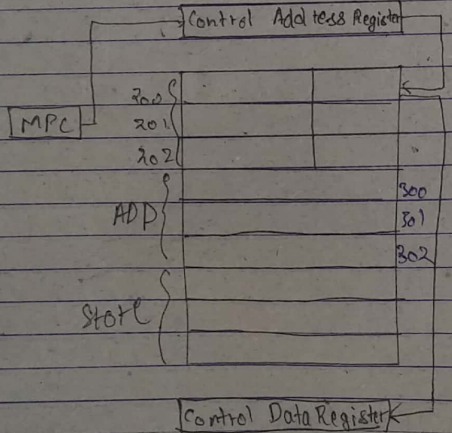
control signal generator also gets its input from condition codes. These conditions may arise due to various conditional operations such as carry (0 or 1), 0 (if the result is zero it is set 1 or if the result is one it is set to 0) other conditions are branch and jump. Branch and jump may be conditional or unconditional

$$Z_{in} = T_1 \cdot Add + T_2 \cdot Add$$

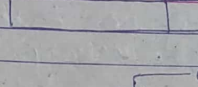
18/3/19

Microprogrammed Control





1 → Control signal Active
0 → Control signal Inactive



updation
MPC ∝ depends upon OPcode and next address

Control signals in Add $(R_3), R_1$

Micro operation	P_{in}	PC_{out}	MAR_{in}	Read	MDR_{out}	IR_{in}	Y_{in}	select	Add	Z_{in}	Z_{out}	
1	0	1	1	1	0	0	0	1	1	1	0	C_1
2	1	0	0	0	0	0	0	0	0	0	1	C_2
3	0	0	0	0	1	1	0	0	0	0	0	C_3
4	0	0	1	1	0	0	0	0	0	0	0	C_4
5	0	0	0	0	0	0	1	0	0	0	0	C_5
6	0	0	0	0	1	0	0	1	1	1	0	C_6
7	0	0	0	0	0	0	0	0	0	0	1	C_7

Each individual bit represent a control signal

micro routine of Microprogrammed Control room

Micro operation	R_{1out}	R_{1in}	R_{8out}	WMFC	END	
1	0	0	0	0	0	Fetch
2	0	0	0	1	0	
3	0	0	0	0	0	
4	0	0	1	0	0	Add
5	1	0	0	1	0	
6	0	0	0	0	0	store
7	0	1	0	0	0	

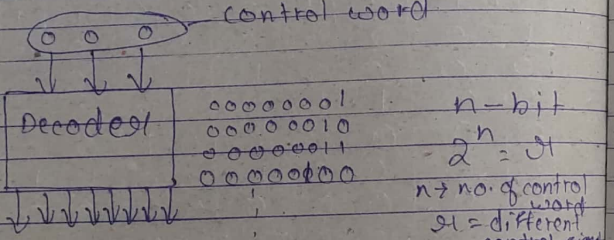
Fetch, Add, store in same memory

- fetch, Add, Store are is same memory but at different memory location
- ROM is used most time for control memory.

low control signal

① Horizontal

② Vertical



n-bit

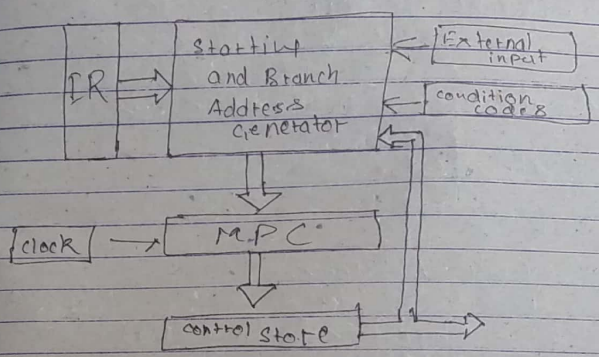
$2^n = 8$

n → no. of control word

8 = different control signal

different

3 input gives 8 control signals



Difference b/w Hardwired & microprogrammed control.

8-9 difference

Cache Memory

① Hardware

② Software

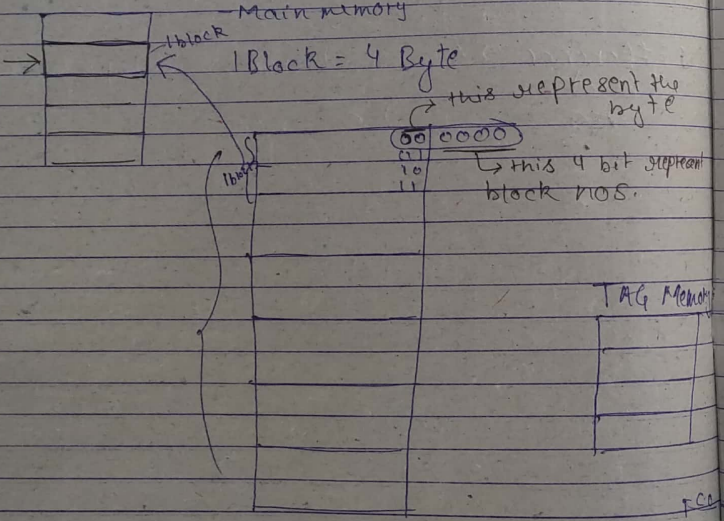
CPU (Generate Address)

↓

Cache (hit) → Data (CPU)

↓miss

Main memory



Size of ~~cache~~ cache block = Size of block of memory

or

~~cache~~ cache Memory

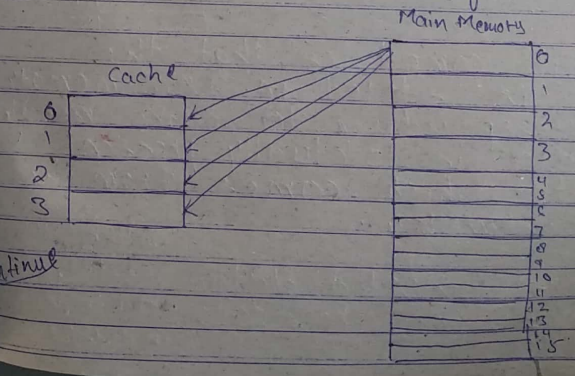
or

cache line

- Cache may be hardware or software
- software by using keyword in creating web page bz it is already stored in cache memory.

Cache Memory is of 3 types

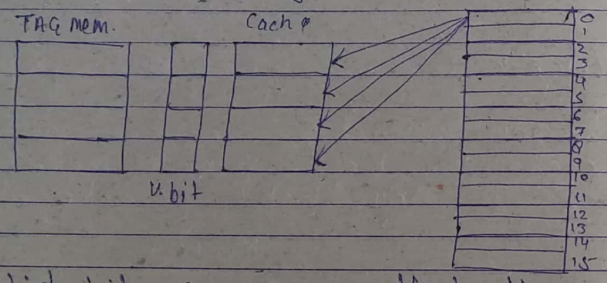
① Associative Cache ⇒ In associative cache the block of main memory can go any block of cache memory.



TAG Memory is used to store the address of the block of cache memory. If n no. of blocks then n no. of addresses is there in tag memory.

4	2
Block No	Byte

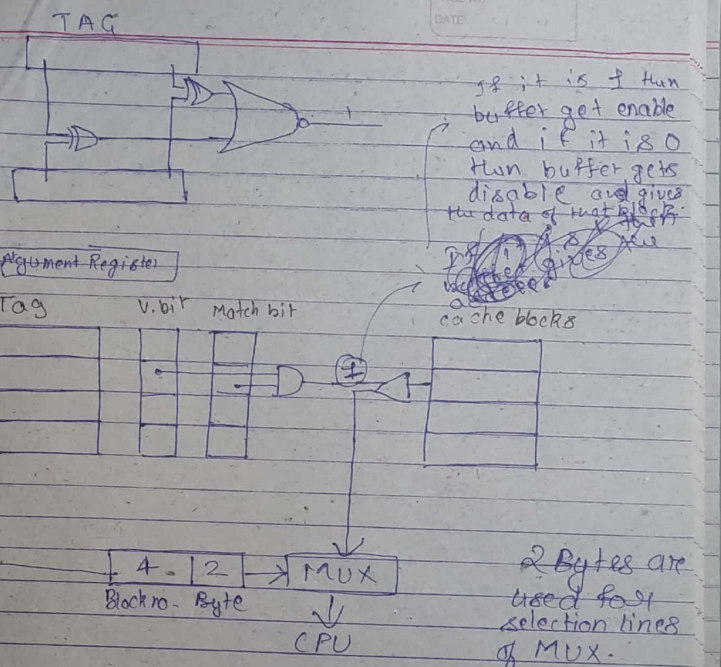
V bit = 1 → valid data
V bit = 0 → Garbage data



Valid bit compares that the address is stored in TAG memory in corresponding to cache memory the data is actually present or not which comes from the main memory.

V-bit = 1 → valid data — data is there in corresponding add. to add
V. bit = 0 → Garbage data — data is not there in corresponding to add.

Buffer



2 Bytes are used for selection lines of MUX.

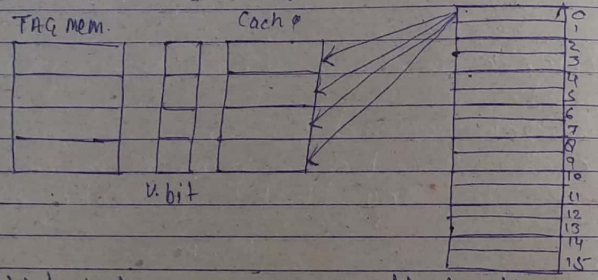
00 - 1 byte
01 - 2 byte
10 - 3 byte
11 - 4 byte

When there is transfer from main memory to cache memory there is only 1 block will transfer only.

TAG Memory is used to store the address of the block of cache memory. If n no. of blocks then n no. of addresses is there in tag memory.

4	2
Block No	Byte

V bit = 1 → valid data
V bit = 0 → Garbage data

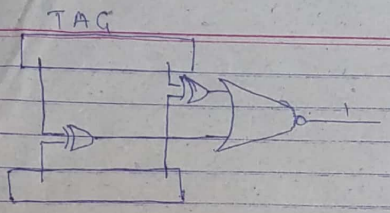


Valid bit compares that the address is stored in TAG memory in corresponding to cache memory the data is actually present or not which comes from the main memory.

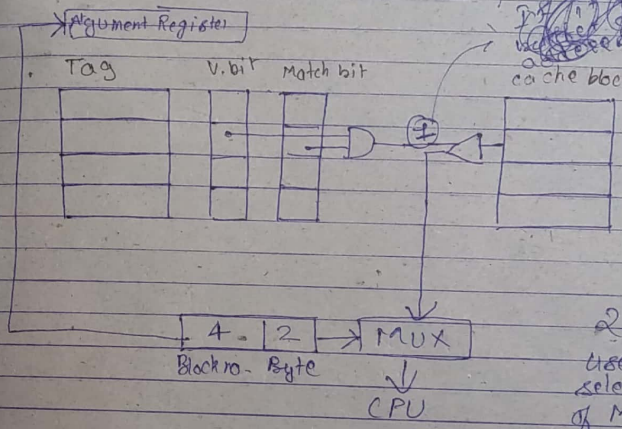
V-bit = 1 → valid data
V. bit = 0 → Garbage data

data is there in corresponding add. in cache
data is not there in corresponding to add.

Buffer



If it is 1 then buffer get enable and if it is 0 then buffer gets disable and gives the data of that block



2 Bytes are used for selection lines of MUX.

00 - 1 byte
01 - 2 byte
10 - 3 byte
11 - 4 byte

When there is transfer from main memory to cache memory there is only 1 block will transfer only.