(1)

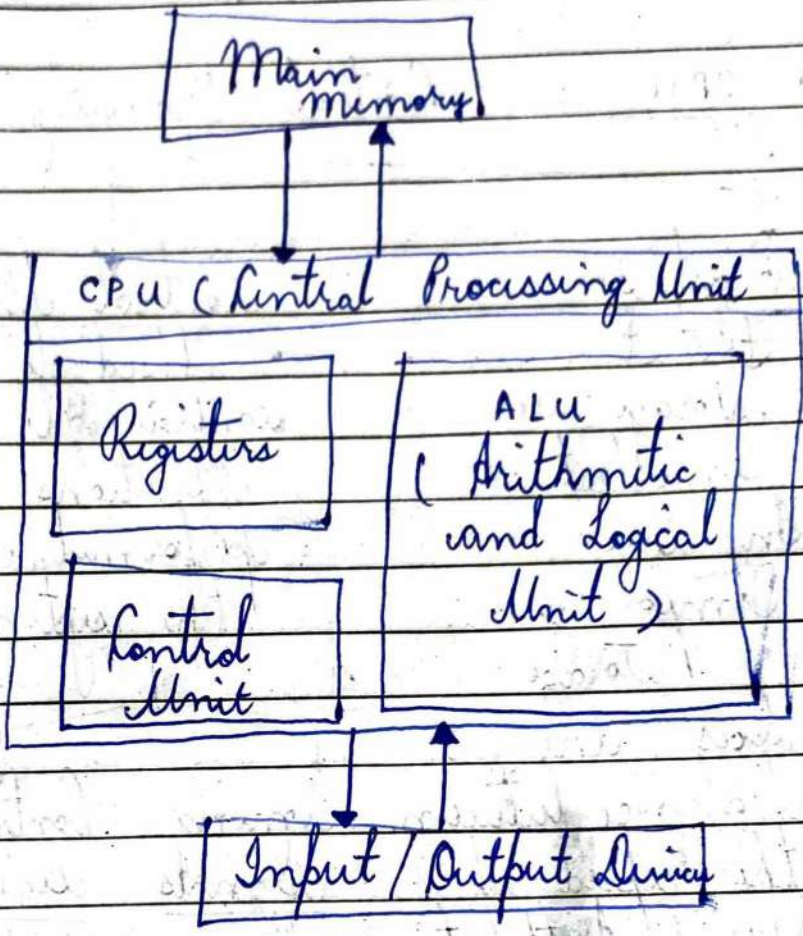
Instruction is a binary sequence bind with some operation.

Lecture 2

VOON - NEWMANN ARCHITECTURE

→ Von Numann's Architecture states that the data and it's program as stored in main memory, but can be stored at different address locations.

Diagram =



Explanation =

i) Main Memory = data [Main Memory
program [

→ Stores data and it's program.

→ Data is the unprocessed / raw and

als such as variables, values, datatypes

Program = The Implementation and values allocated to variables

eg = int a = 10; } data
int b = 20; } data } program
int c = a+b;

ii) CPU = (Central Processing Unit)

Registers
→ Fastest storage
→ Smallest in size
→ Temporary storage
→ Does the balance between the speed of ALU (Arithmetical and Logical Unit) and main memory.
→ Combination of flip flops.

Control Unit
→ Used to control all operations performed in the system

Timing Signals
→ All particular time fixed Instructions are to be performed
these are controlled by timing signals.

Control Signals
→ All Operations performed by Registers, timing signals, main memory
Arithmetical and Logical Unit

ALU (Arithmetical And Logical Unit)
→ All mathematical and logical simple or complex calculations are performed by Arithmetical and Logical U
eg = +, -, *
→ Built h
hub of multipl

(3)

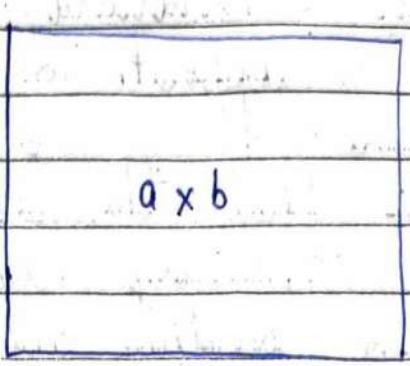
iii) Input / Output Unit =
→ The entry and display of data through peripheral devices is termed as Input / Output Unit.

Note = Hardward Architecture is totally opposite of Von Neumann's Architecture.
It states that the data is stored in another memory and it's program stored in another memory.

→ Introduced in 1945.

DIFFENT TYPES OF REGISTERS

Explanation of Memory =



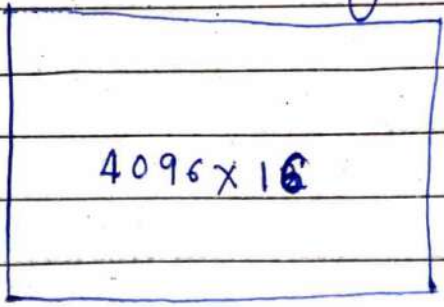
a = Number of Words
b = Number of bits per word

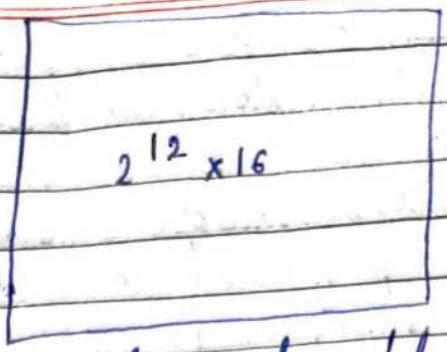
Note = Here, In today's era memory is word Addressable because memory, a word can store as many bits as the system dem but byte is having fixed amount of bits i.e; 8 bits.

Here $a = 2^k$

k = Address Locations or number of
b = Number of bits per word.

Eg =





12 = Number of words in memory
16 = Number of bits per word

$\therefore$ k = 12

Registers :

i) Address Registers :
→ Deals with the Address Locations or the total number of words.

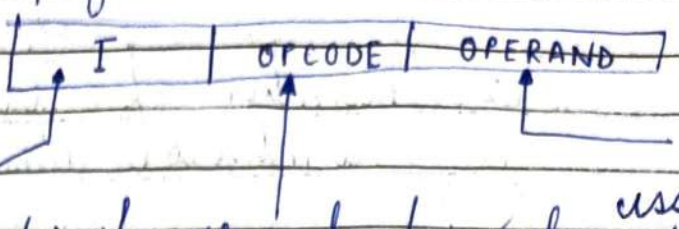
~~ii) Data Re~~
Working = Instruction (Address) → decode → Accumulator → Performance

ii) Data Registers :
→ Used to ~~is~~ store data on which operations is to be performed.

iii) Accumulator :
→ It is used to store Intermediate data As
When we fetch data then it gets stored in Accumulator before performing operations.

iv) Program Counter =
→ Used to store address of next Instruction

v) Instruction Register = Used to store Instruction to be performed on data.



→ 0 → direct Addressing.
→ 1 → Indirect Addressing

Note = Direct Addressing Refers to that operation is to be performed on given Address.
while
Indirect Addressing Refers to the Address location containing another address to location where the operation is to be performed.

vi) Temporary Register =
→ Used to store temporary data.

vii) Input Register =
→ Used to take Input of data from peripheral devices.
→ Not having any link with memory.

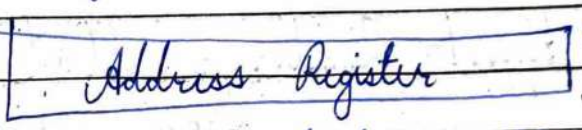
viii) Output Register =
→ Used to display the processed information by Arithmetical and Logical Unit.

Eg =

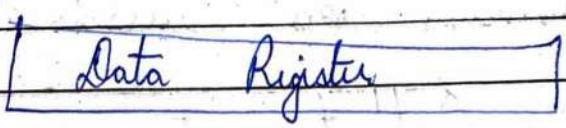
4096×16
$2^{12} \times 16$

Number of words or number of Address or Number of memory locations = 1
Number of bits per word or number of bits for Address or Number of bits for memory locations = 16

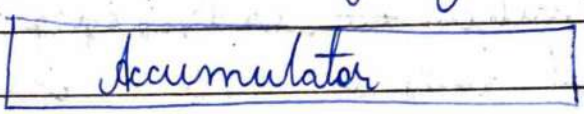
i) Address Register = deals with Address



ii) Data Register = deals with data



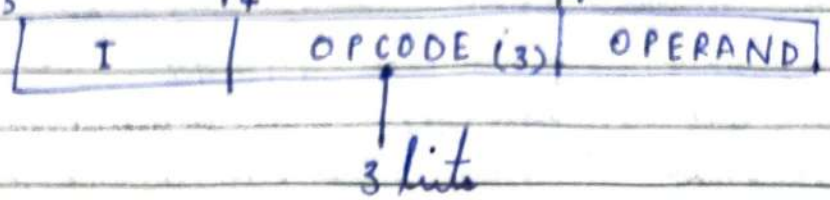
iii) Accumulator = deals with Intermediate storage of data.



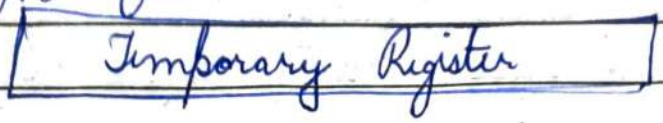
iv) Program Counter = deals with Address of next Instruction or

Operation = [Program Counter]

v) Instruction Register = deals with Address of operation and operation to be performed and MSB (Most Significant bit) (Direct Addressing or Indirect Addressing) in combination with data.



vi) Temporary Register = deals with temporary storage of data.

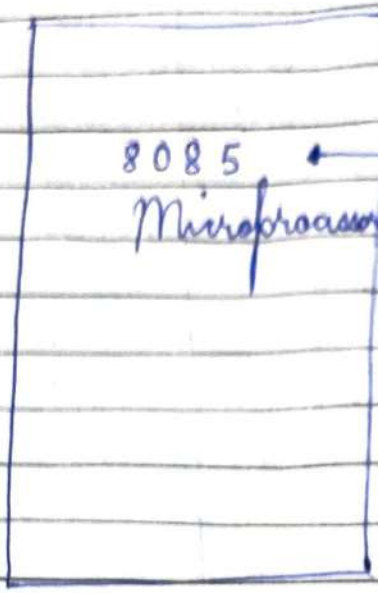


vii) Input Register = Used for taking Input from peripheral devices.
→ Not dependent on memory, hence not dependent on words.

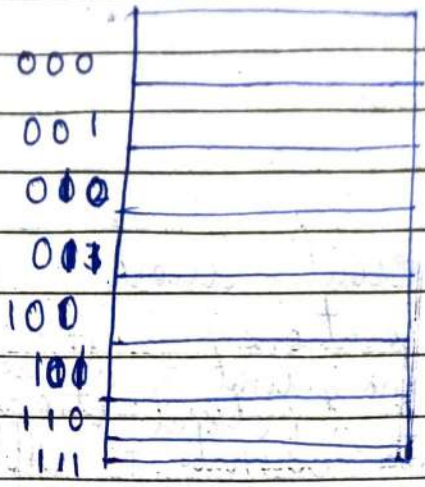
viii) Output Register = Used for displaying processed data to the output devices by fetching it from Arithmetical and Logical Unit through control signals and buses i.e. (Not word topology).
→ Not dependent on words or memory or address.

Lecture 4

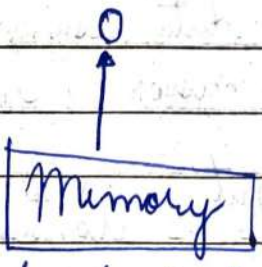
TYPES OF BUSES



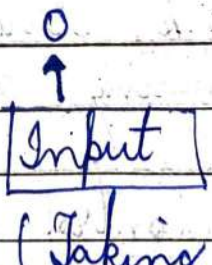
Eg: 2^8



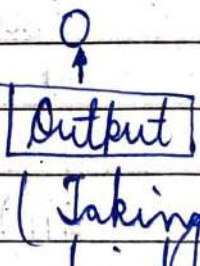
Here,



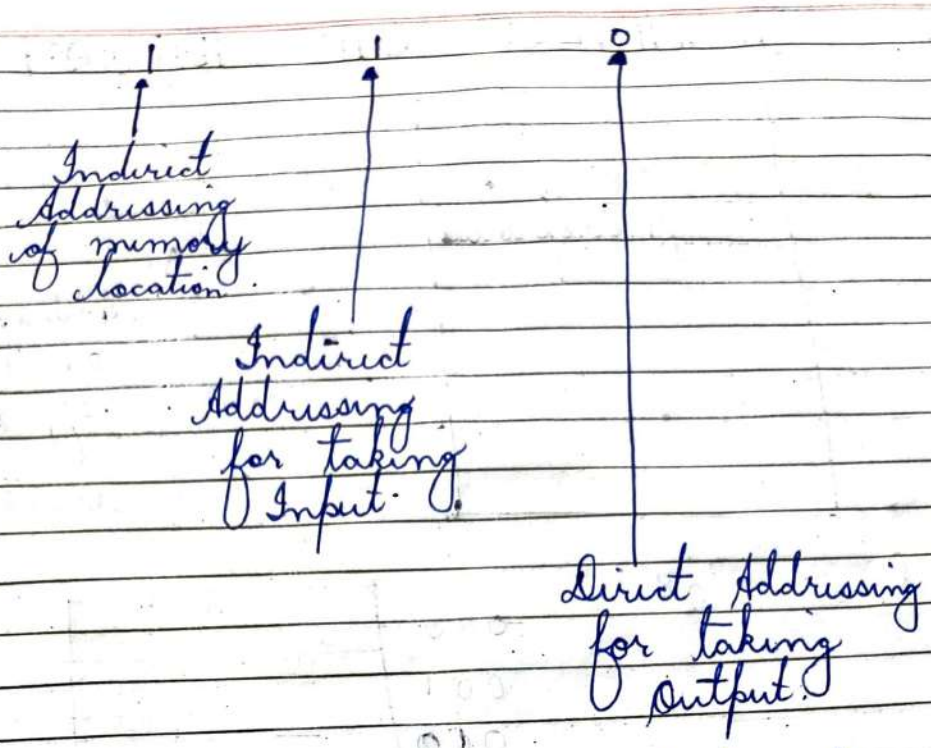
(direct Indexing / direct Addressing)



(Taking direct Input from memory location)



(Taking direct Output from memory loca)



and will work similarly for each memory locations.

i) Memory Unit = Stores address containing Instructions

ii) Data Signals = All data processed or unprocessed is stored in work through data signals.

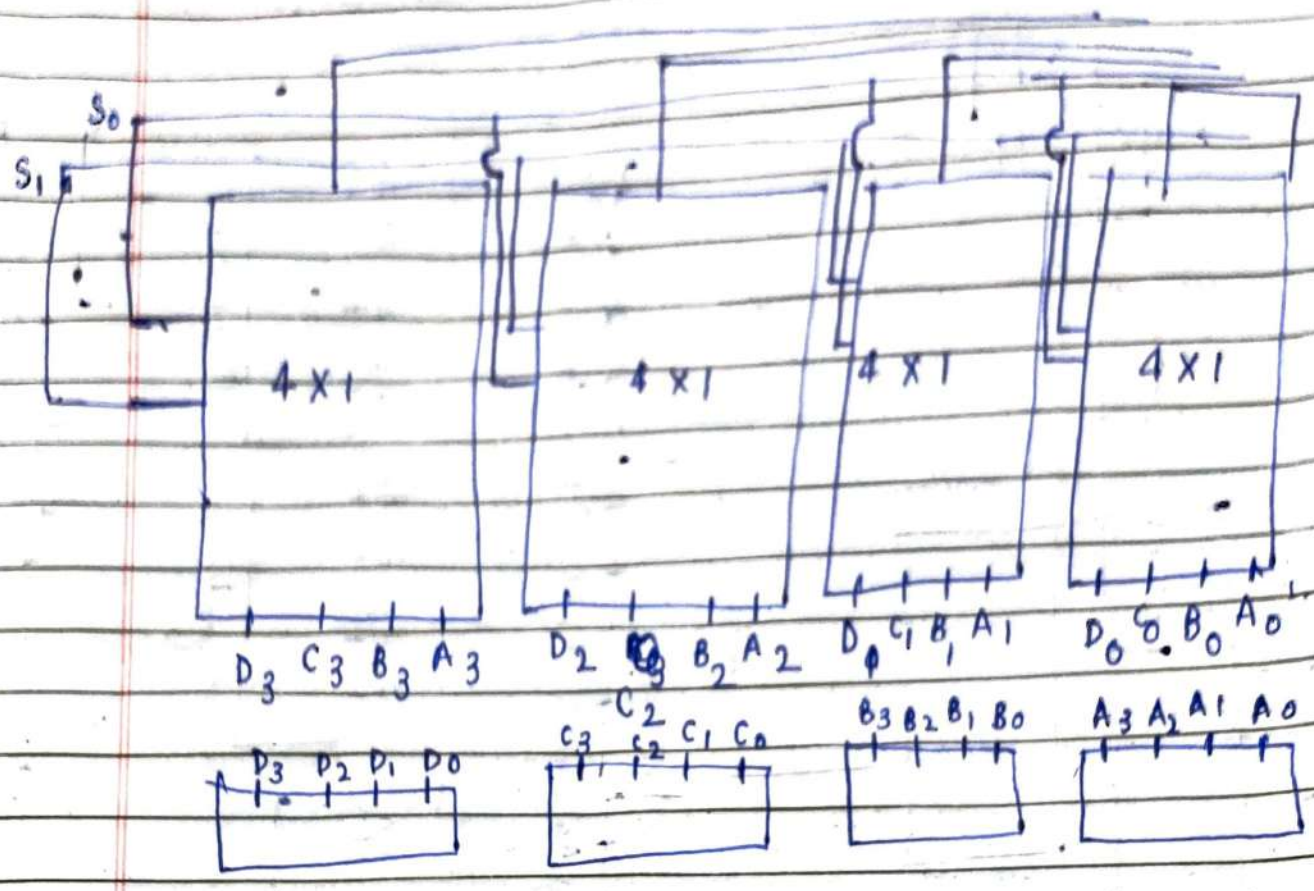
→ Having a processor Installed already

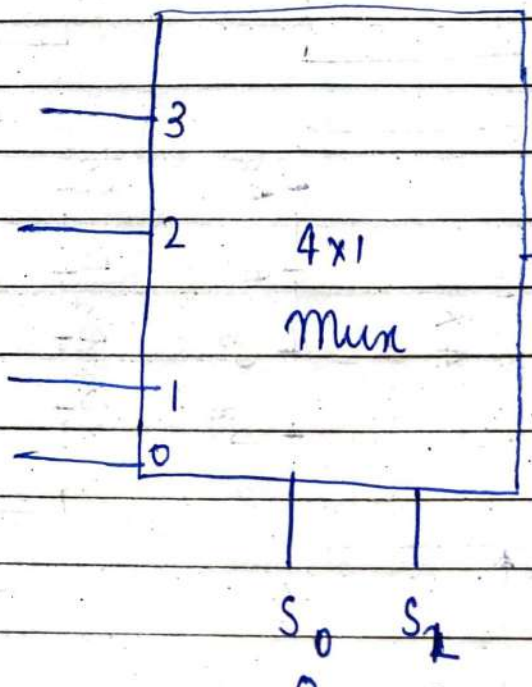
iii) Control Signals = Contains Timing Signals and Control Signals which are having Instructions set already to perform and Basics as networking or network topology

(1)

Lecture 5 ...

Bus architecture :



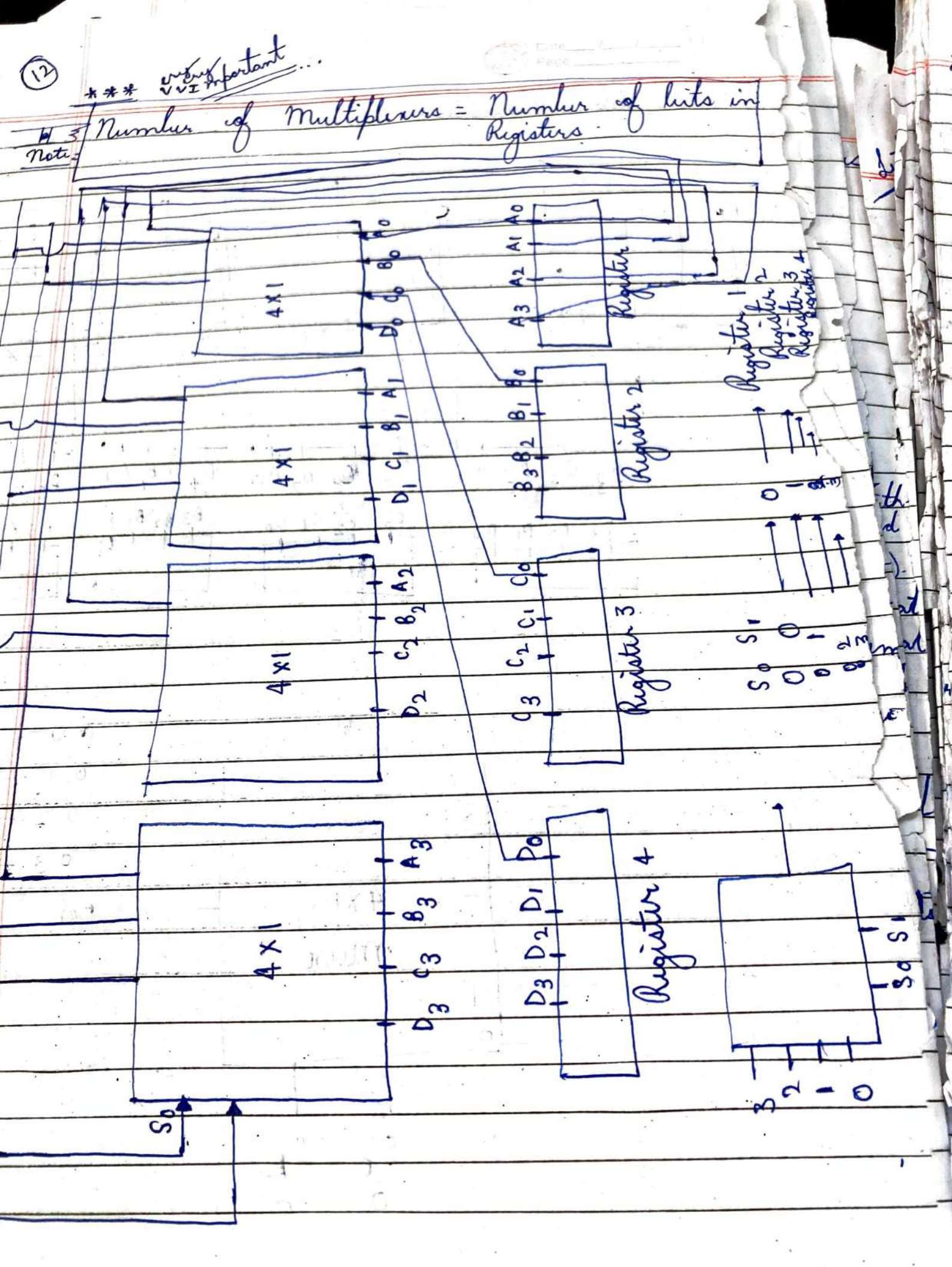


0 0 → 0
0 1 → 1
0 2 → 2
0 3 → 3

(12)

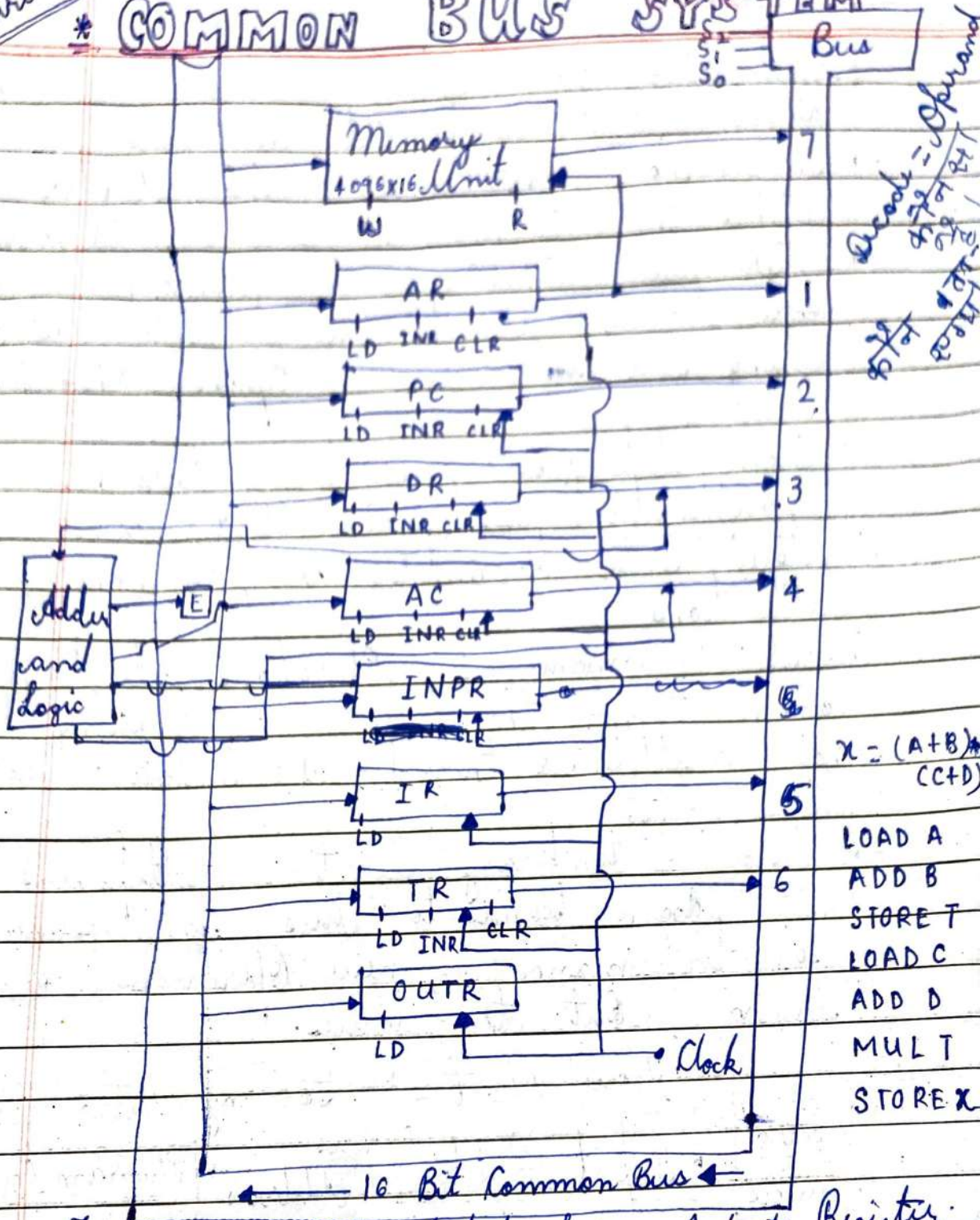
*** very very important ...

Note: Number of Multiplexers = Number of bits in Registers.



Lecture 6...

* COMMON BUS SYSTEM



$x = (A+B) * (C+D)$

LOAD A
ADD B
STORE T
LOAD C
ADD D
MUL T
STORE X

→ First we take Input from Input Register.
→ The process starts from program counter as it stores address of next location.
→ Then the address of that value transfers to Address register through bus by giving value to multiplexer by program counter.
→ By Address Register, it transfers to Memory

Unit by Again giving multiplier value through Address Register going into the bus.

→ The memory register is having decoder which fetches the data of that particular Address.

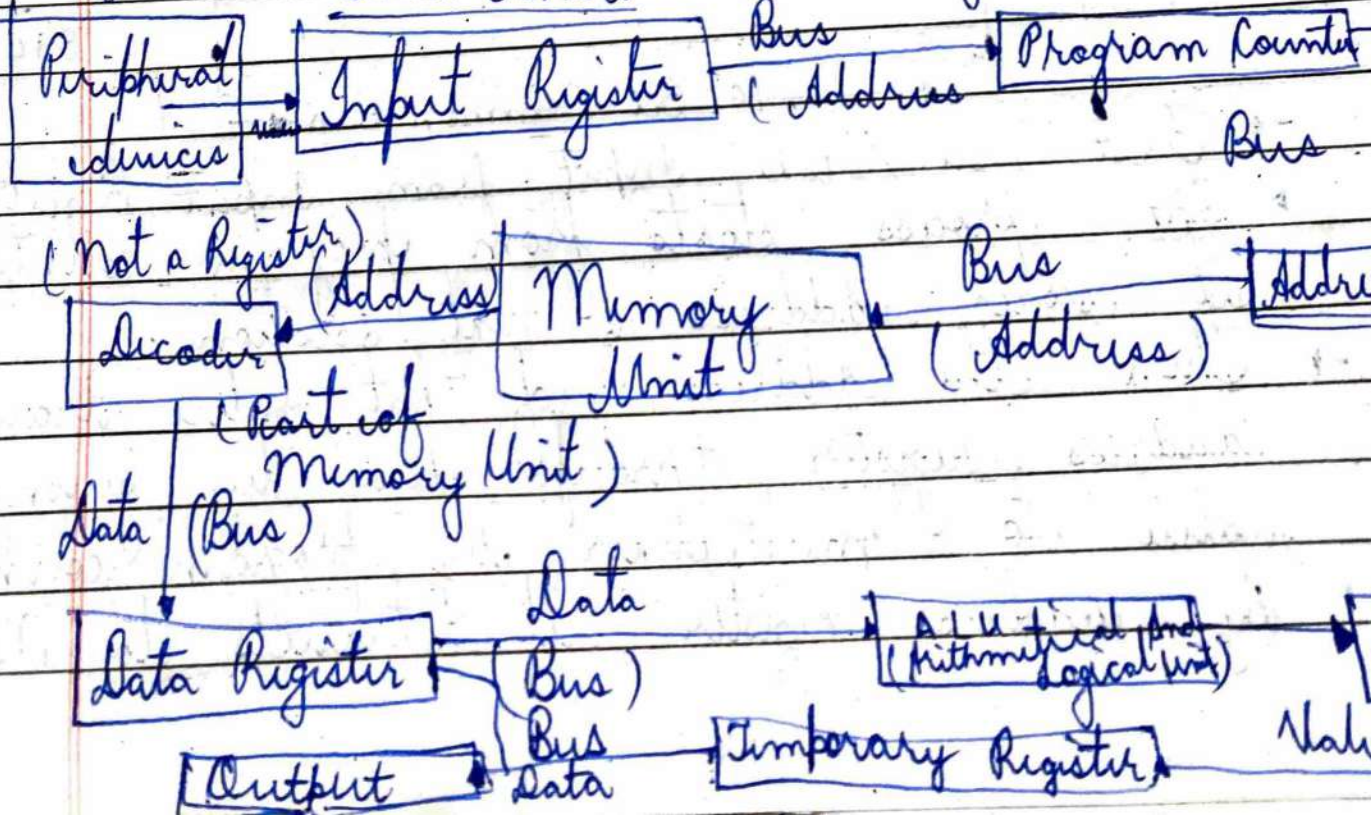
→ The data fetched by the memory register again travels through bus to data Register by giving the value of multiplier through Memo Unit.

→ From data register it passes through ALU (Arithmetical and Logical Unit) and get stored into Accumulator for performance of Operations.

→ After performance of Operations it the again travels through bus by giving result of multiplier through Accumulator. to tempo any register.

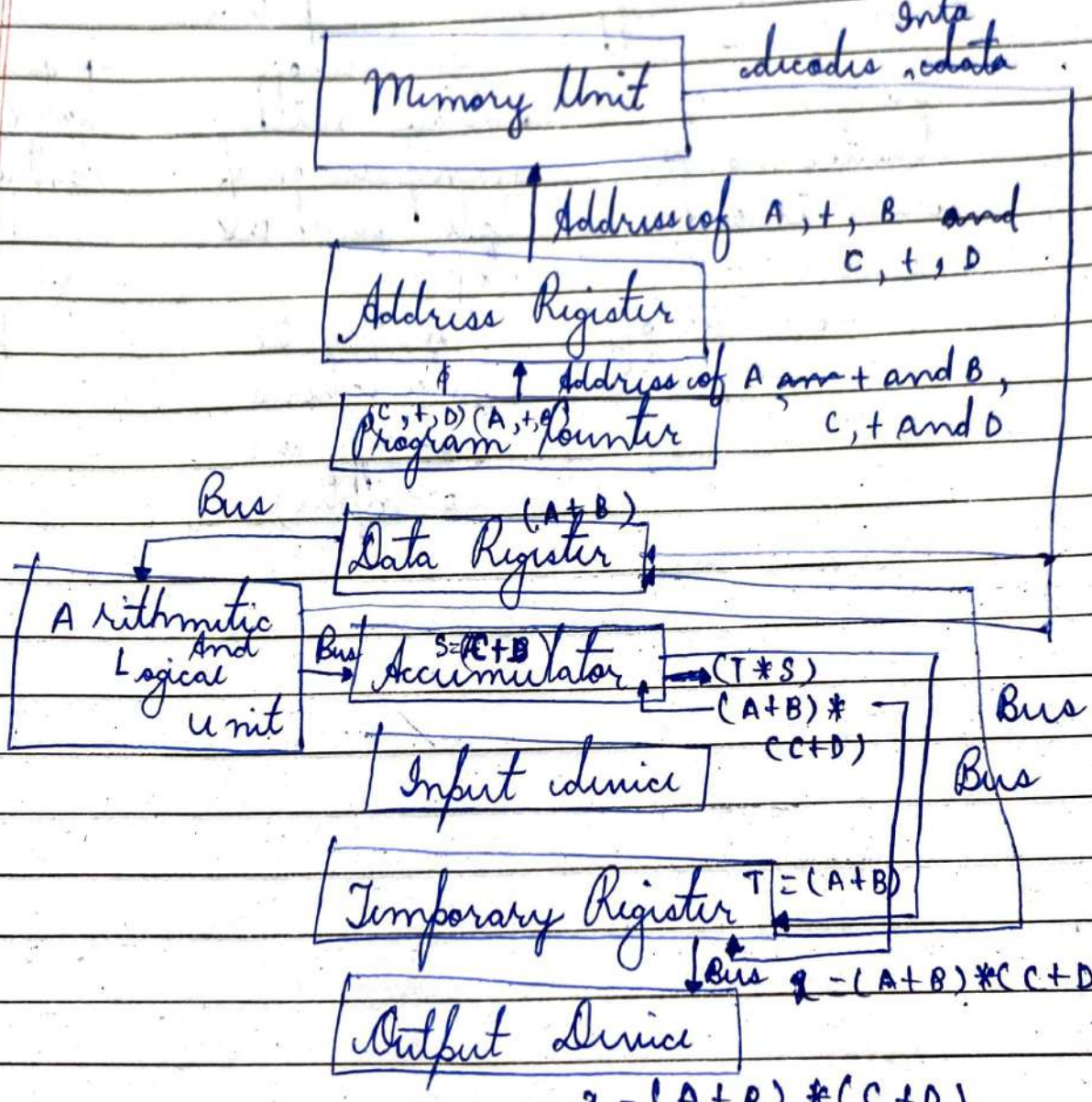
→ Through temporary register multiplier's valu it Again comes to bus for going to ALU for performance of new Operations or for giving out the values.

Flow Chart - (For my better Understa



(15)

Q LOAD A
ADD B
STORE T
LOAD C
ADD D
MUL S
STORE x



Hence, final Answer $\boxed{x = (A+B) * (C+D)}$

Lecture 7

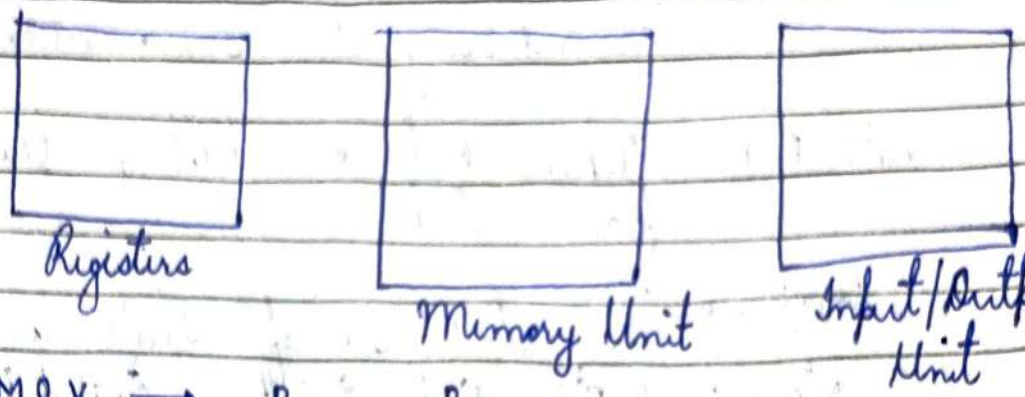
* TYPES OF INSTRUCTIONS

Instructions

- Data Transfer Instruction
  - → It deals with transfer or travelling of data through buses or network topology.
- Data Manipulation Instruction
  - → Deals with manipulation of data such as logical, arithmetic calculations (simple or complex).
- Program Control Instruction
  - → Deals with various program control statements such as If, else, for, while etc.

Unit 8 ...

DATA TRANSFER INSTRUCTIONS



i) MOV → R1 , R2
Means Giving value of R2 to R1.
Eg = MOV → R1 , 100
Giving value 100 to R1.

ii) LOAD → Loading the value from memory unit to register (Mostly used for Accumulator).

iii) STORE → Opposite of load, loading values from registers to memory Unit.

iv) EXCHANGE → Exchanging the Values of Registers

Eg : R1 R2

Q. R1 = 100 , R2 = 200
⇒ R1 = 200, R2 = 100

v) INPUT → Transfering Instructions from peripheral devices (Input devices) to Memory U

vi) OUTPUT = Taking the processed Information and transferring it to output devices.

vii) PUSH = Storing the Information into stack or transferring the Information into stack

viii) POP = Taking out or deleting the Information from stack.

* Conclusion = Data Transfer Instructions

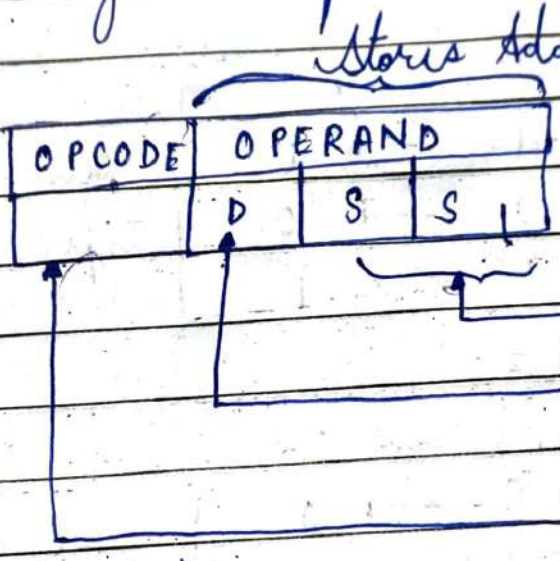
Move Load Store Exchang Input Output Push Pop

(19)

Lecture 9 ...

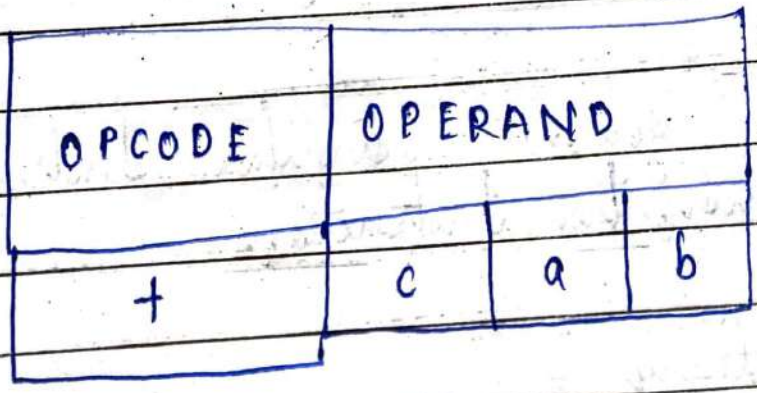
ARITHMETIC INSTRUCTIONS

i) Add → For adding Values
ii) Subtract → For Subtracting Values
iii) Multiplication → For multiplying Values
iv) Division → For dividing Values
v) Addition with Carry → Multipliers (Source)
vi) Subtraction with Carry → Multipliers (Source)
vii) Negate → Changing positive to negative and negative to positive.



Eg = O = (a+b)

⇒



Lecture 10

LOGICAL INSTRUCTIONS

i) Complement (COM or NOT) =
→ The value will become opposite of that value

T → F

$1^c = 0$

$0^c = 1$

ii) Clear =
→ Clears all the Information saved.

iii) Logical AND =
→

0	0	→	0
0	1	→	0
1	0	→	0
1	1	→	1

} Acts As multiplication type.

iv) Logical OR =

0	0	→	0
0	1	→	1
1	0	→	1
1	1	→	1

} Acts As Addition type.

v) Ex-OR =
→ Giving zero on similar Instructions and 1 on disimilar Instructions.

vi) Clear Carry =
→ Clearing the value of carry for calculating 1's compliment or two's compliment.

vii) Set Carry =
→ Setting the value of carry for taking one's compliment or two's compliment.

viii) Compliment Carry =
→ Taking out the carry of the compliment.

*** ix) Enable Interupt =
→ There is To Enable the Interupt function.

- Hardware Interupt
  → Using mouse, pressing keyboard and many more Interupts the working of system.
- Software Interupt
  → Using different memory locations and many more Interupts working of the system.

*** x) Disable Interupt =
→ To disable or stop Interupt function.

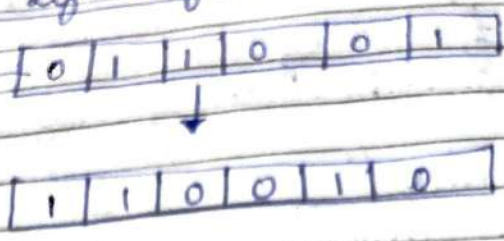
- Hardware Interupt
- Software Interupt

Lecture 11

SHIFT - INSTRUCTIONS

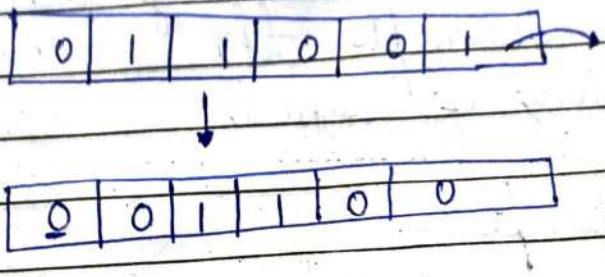
* → Loading operand → Register → Requires bits shifting either Left or Right

i) Logical Left Shift =

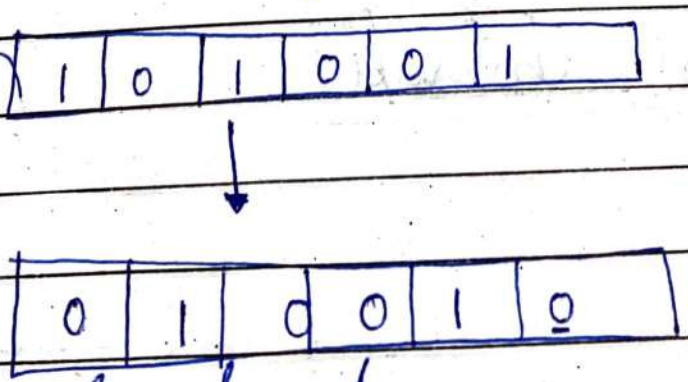


Division of 2

ii) Logical Right Shift =

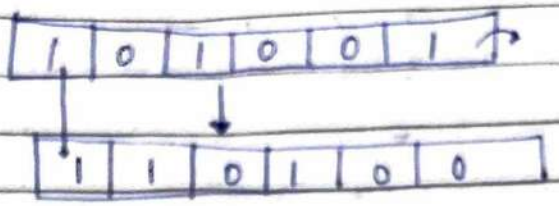


iii) Arithmetical Left Shift =



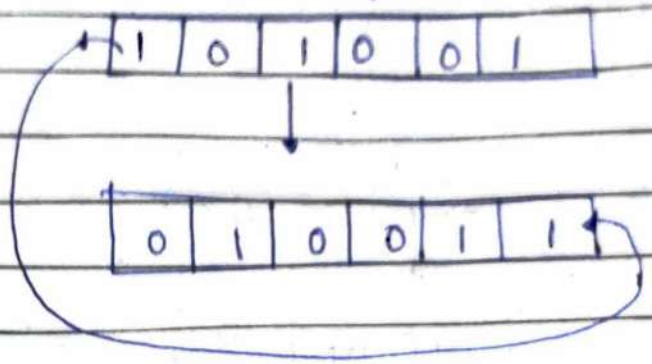
Note = Exact Same as Logical Left Sh

iv) Arithmetic Right Shift =

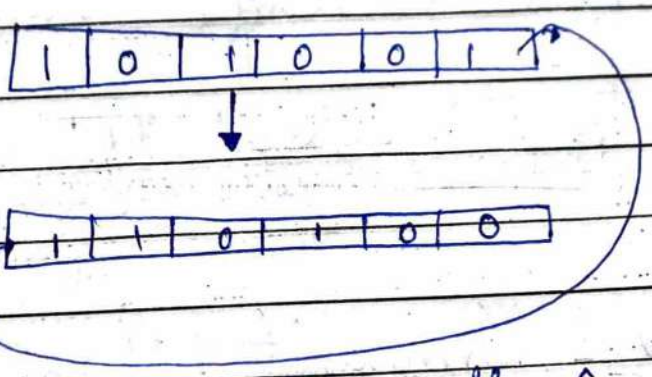


} Similar data was removed from vacant position

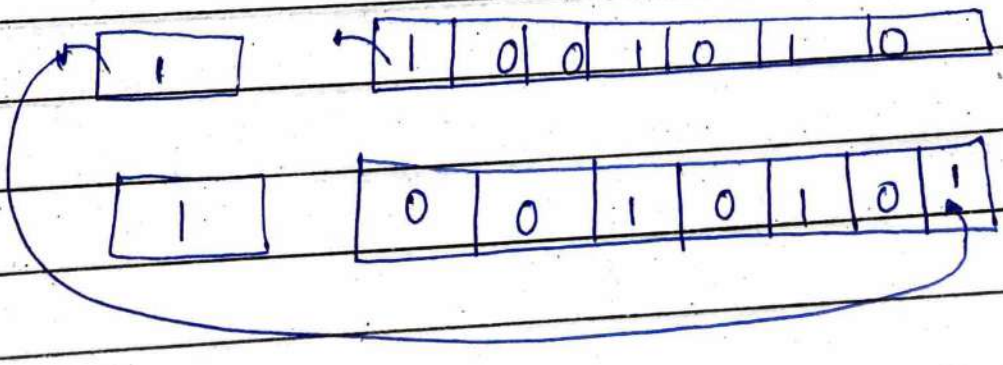
v) ~~Carry~~ Rotate Left Shift =



vi) Rotate Right Shift =



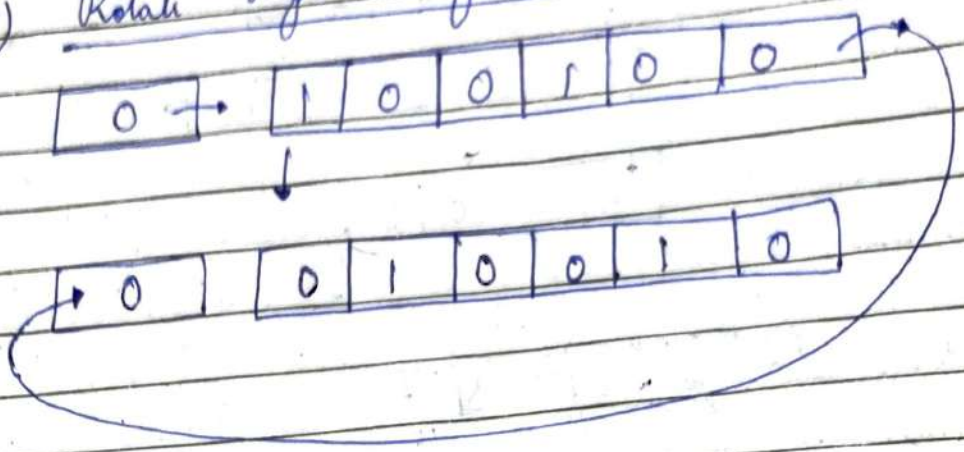
vii) Rotate Left Shift with Carry =



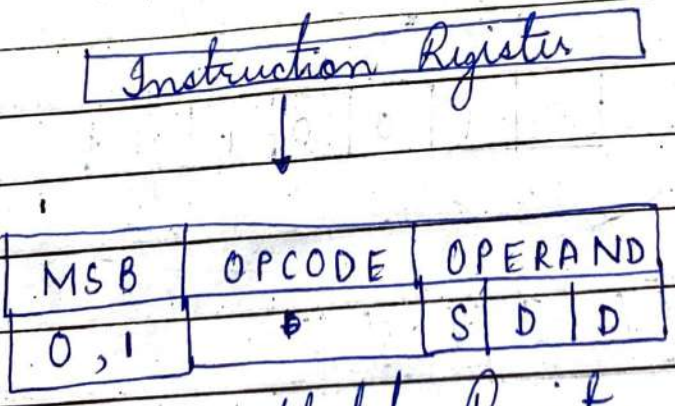
(22)

(24)

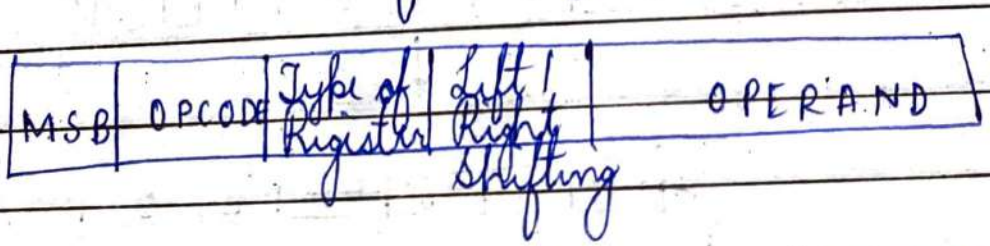
viii) Rotate Right Shift with Carry



☞ Note =



But Now in Shift Register



More Accura Struct

☞ Note = Immediate Operand = Direct Values

Lecture 12

(25)

PROGRAM CONTROL INSTRUCTIONS

Branched

Conditional

→ According to given conditions, it works.

Eg = BE R1 R2 2000.

Means if Branched Equal to

$R_1 = R_2$.

then program counter will have Address of Instruction as 2000 and will get Executed.

Unconditional

Jump =

→ Used to jump from one position to another, not necessarily in Sequential Order.

→ Program Counter will store the Address accordingly

Skip =

→ Skip the particular position to another given position.

→ Like In simple Memory Unit, Program Counter gives the Address of next location and hence gets decoded and transfers it i.e. (Network topology) Sequentially

→ But if we want to skip or jump

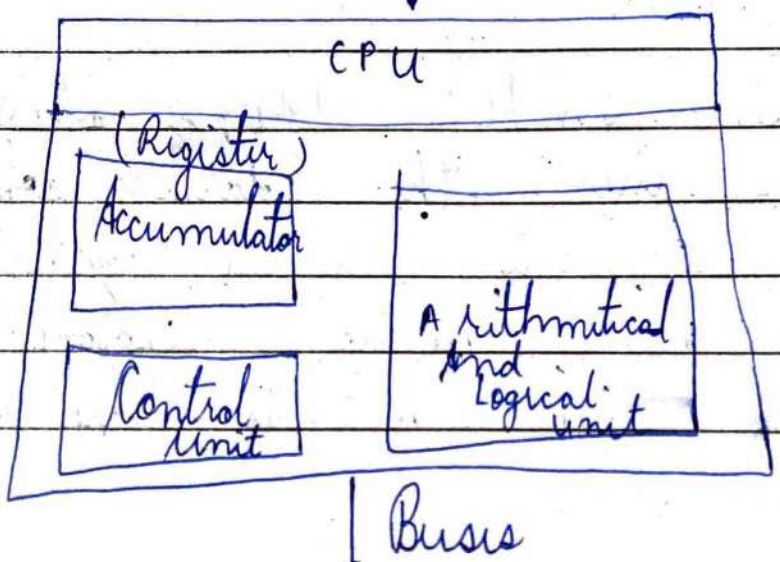
Lecture 13

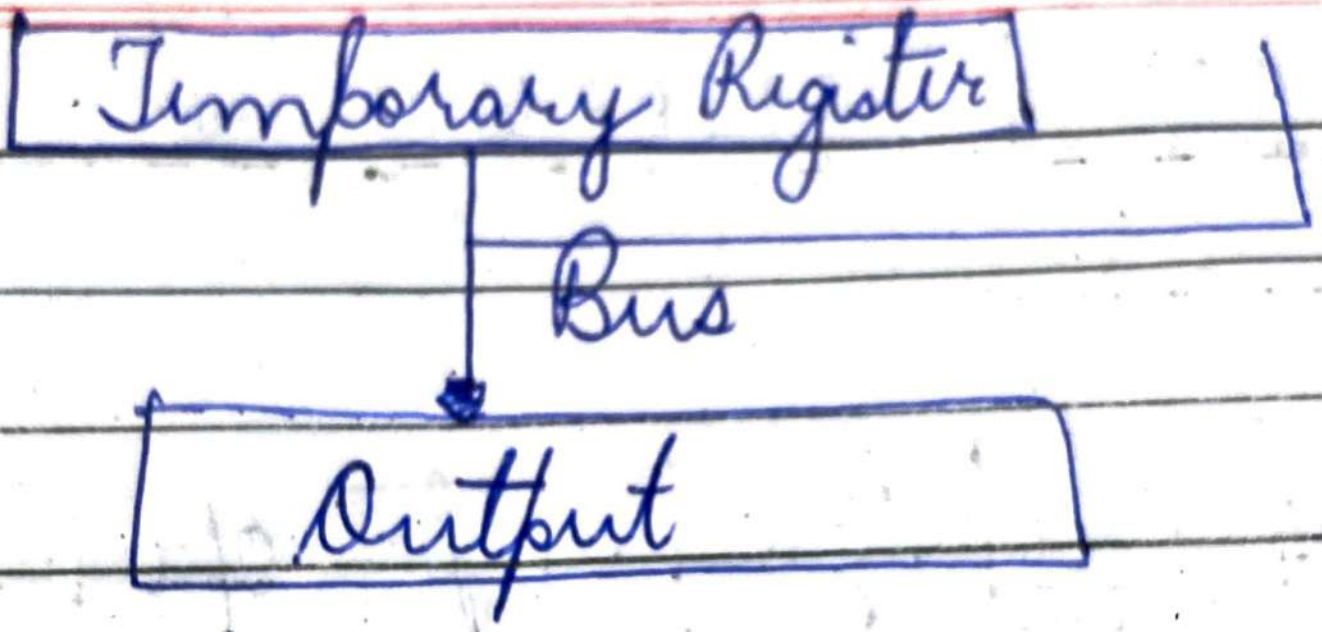
INSTRUCTION FORMAT

Program Counter
↓
Address Register
↓
Memory Unit
↓
Bus
↓
Instruction Register

MODE	OPCODE	OPERAND

- MODE → The Address of Operand / Address of Address where Operand is present / Address of Register where Operand is present.
- OPCODE → Arithmetic Instructions to be performed
- OPERAND → + Shifting if required / On which to be performed




Temporary Register
Bus
Output

Lecture 10

16 bit

6	OPCODE	OPERAND

QUESTION

Operand	Reg 1	Reg 2	Unused
6	3	3	4

16 - (6+3+3) : 4

$2^8 : 256$

$0 - 255$ } Signed Integer

-128 to 127 → Range

Conclusion =

Types Of Instructions

Data Transfer Instructions =

- MOV
- LOAD
- STORE
- EXCHANGE
- INPUT
- OUTPUT
- PUSH
- POP

Data Manipulation Instructions =

- Arithmetic
  - ADD
  - SUBTRACT
  - MULTIPLICATION
  - DIVISION
  - ADDITION USING CARRY
  - SUBTRACTION USING CARRY
  - NEGATE

Program Control Instructions =

- Branched
  - Conditional
    - Branched Equal → Jump
    - Branched → Skip Unequal
  - Unconditional

Logical data Manipulation Instructions

- COMPLEMENT
- CLEAR
- LOGICAL AND
- LOGICAL OR
- EX-OR
- CLEAR CARRY
- SET CARRY
- COMPLEMENT CARRY
- *** ENABLE INTERUPT
- *** DISABLE INTERUPT

Lecture 15

* TYPES OF CPU ORGINAZATION

i) Single Accumulator Organization

→ Last kaam se kaam Rakhni hai Isliye Register kaam

Question = Load ⇒ Value → Memory Unit

↓

Register

Store ⇒ Value → Register

↓

Memory Unit

Now Single Accumulator means Instructions are stored In Single Register and hence travelling in the buses.

PC

↓

AR

↓

Memory Unit

↓ Decoding

Instruction Register

↓

Data Register

↓

ALU

↓

Accumulator

↓

Buses

Eg : x = A + B

| Mode | OPCODE | OPERAND |

LOAD A
AC + B
A +

x = AC
x = (A + B)
Store x ← AC

and all arithmetical and Logical data Manipulation.

Note = Sara logh ek hii Register Mtha Raha hai, Isliye speed kaam ho jayegi.
Saasta Roye Baar Baar, Mehanga Roye Ekbar.

→ Single Address

Lecture 16 ...

(2) General Register Organization

→ Two or more registers for enhancing the speed and working of system.

MODE	OPCODE	OPERAND		
		D	R1	R2
	+	x = 24	8	16

$R_1 \leftarrow 8$

$R_2 \leftarrow 16$

$D \leftarrow x$

$x = R_1 + R_2$

$x = 24$

→ Note: Similar for two Registers, taking values from one One Register and making another Register as distination.

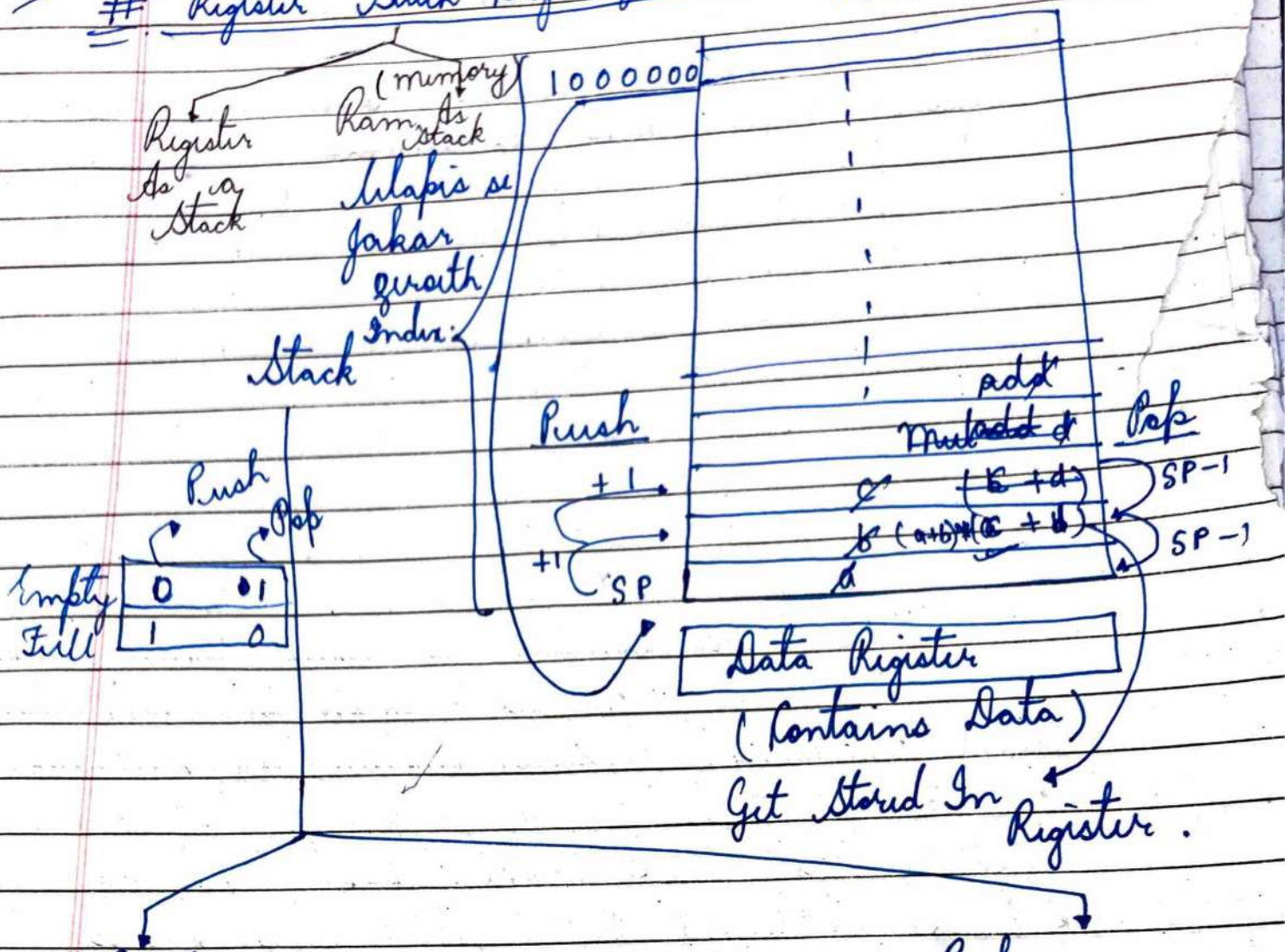
→ Sabki Size chadh jayegi kyoki Register Zyada use ho Rahi hai.

→ Saath me Cost bhi chadh jayega.

Registers ∝ Cost ∝ Size of Bus

Lecture 17

Register Stack Organization = 63 bit



Push =

SP ← SP+1 ← Increasing the Value

M[SP] = DR ← Storing the Value

IF (SP == 0) then (FULL ← 1)

EMPTY ← 0

Pop =

DR ← M[SP] ; Val Et

SP = SP-1 ; // Ek Ek positi

ghatiga

IF (SP == 0)

then (Empty

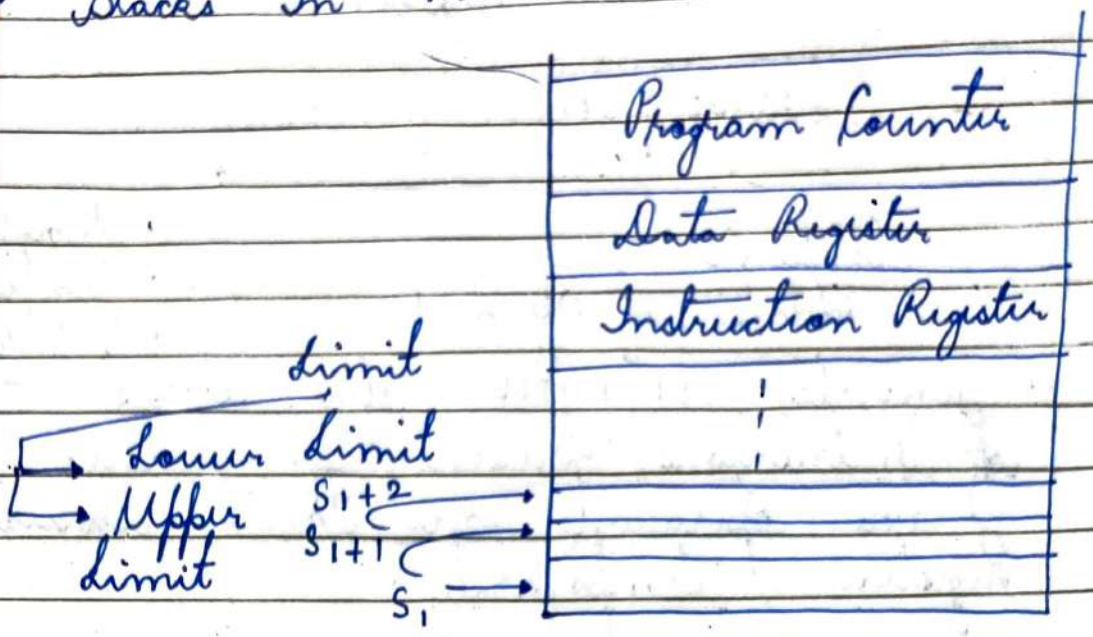
FULL ← 0

Note = Cost will be less as the Many Registers have been used In Comparision of Single Register Architecture and General Register Architecture.

Lecture 18 ....

Memory Stack Organization =

→ Stacks In RAM.



→ Here, program counter stores Address of next Instruction and gives it to memory Unit to decode.
→ After decoding it transfers to data Register and transfers Again to Buses to Instruction Register.

→

Address of Operand
→ direct Address
→ or Address in Register
→ Address Inside

The Operation which is to be performed

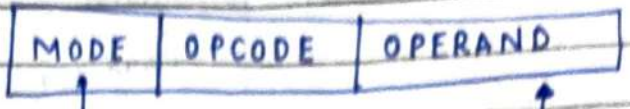
The Instructions on which the Operation is to be perform

→ Cost Effective
→ Now Address Incomparision of Stack Implementation using Register that are stored between Upper Limit and Lower Limit of the stack Implemented in RAM (Memory).

Lecture 19...

Addressing Modes :

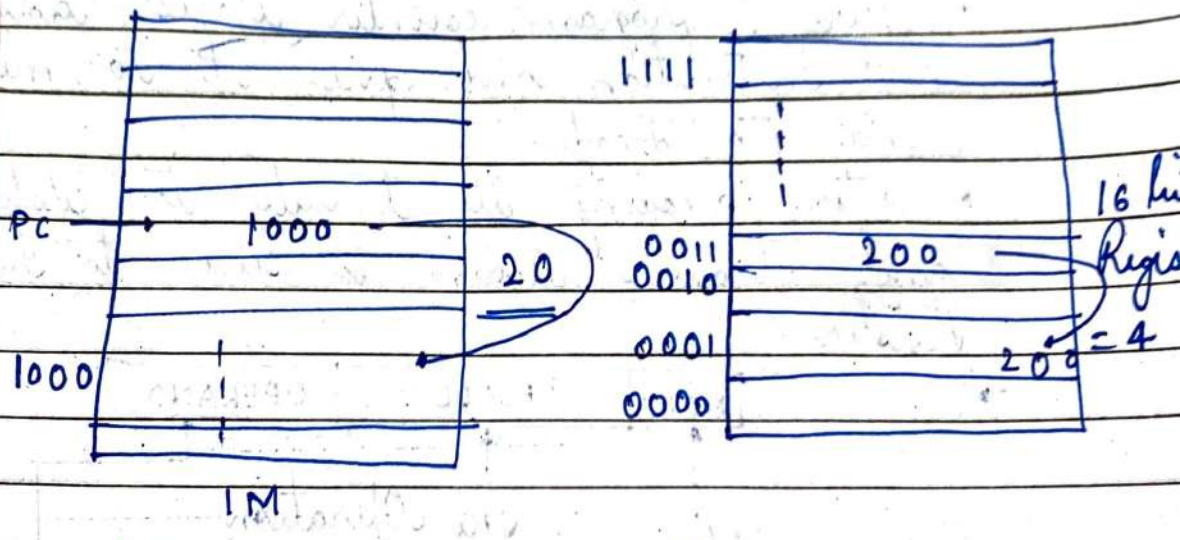
→ In Instruction Register



Contains Address of Operand

→ Register
→ Values
→ Address

→ Generally OPERAND also stores the address of Instructions present in Register because the Operations are to be saved in Registers Temporarily.



→ Jumping On given Address
→ Switching the values are also possible.

→ There are various kinds of addressing modes and we will be studying all of them in later lectures.

Lecture 20 ...

Types of Addressing Mode =

(i) Implied Mode =
→ That the Operand is called by the Opcode Implicitly.
→ Deals with zero value (Stacks) or one value (Accumulator) ster.

i) One Value (Accumulator) =

ADD R1, R2

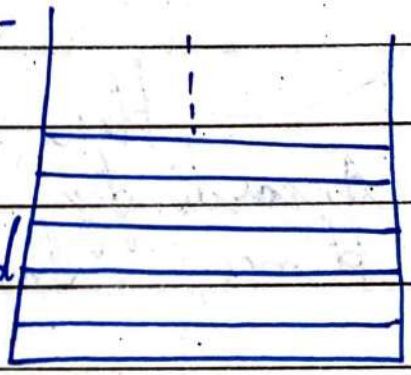
→ Here, ADD is the Opcode and R1, R2 is the Operand which is called Implicitly with Opcode.

R1 ← R1 + R2

Act like Accumulator

ii) Zero Value (Stacks) =

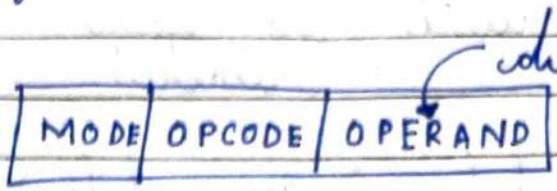
→ The last two Instructions will become Operand on which operation to be performed and getting stored in register.

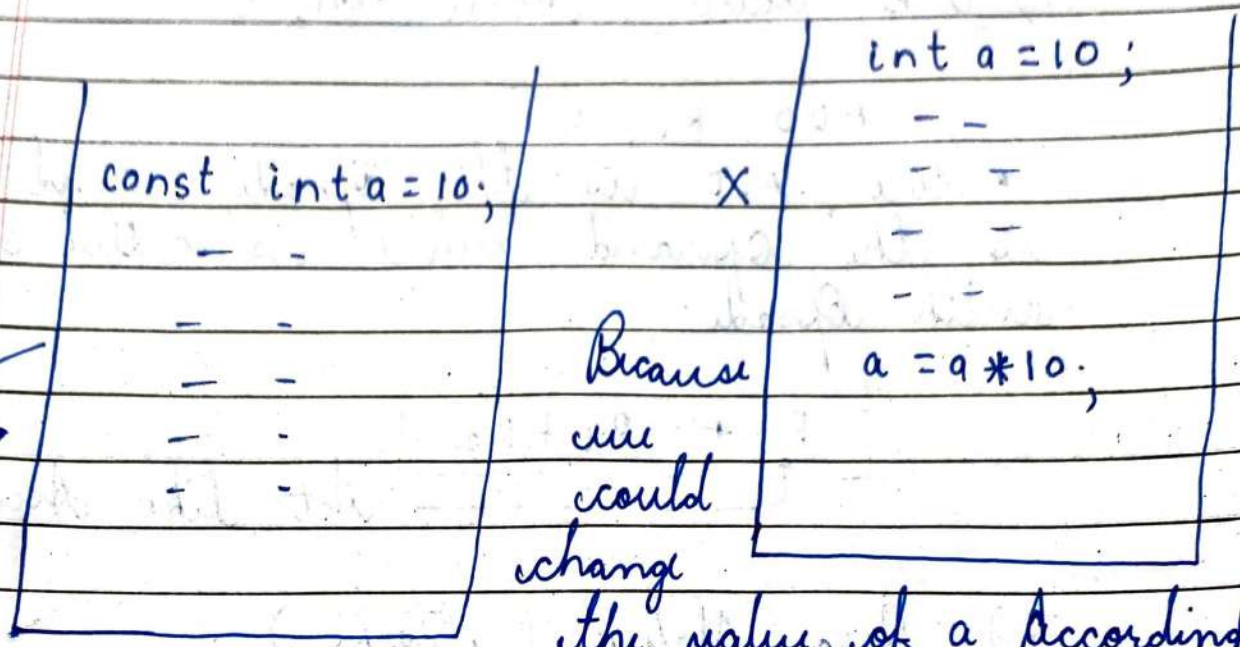


Lecture 21...

* Addressing Mode
Immediate Mode =

→ It gives direct value to the Operand.
→ The value cannot be changed further or no Addressing must be done.





Here we cannot change the value Accordingly.
Hence, Immediate Addressing mode.

Drawback =
→ The value of OPERAND depends on size or ~~num~~ we can say number of bits for words.

Lecture 22 ...

* Register Mode =
→ The Instructions are present in the Register

MODE	OPCODE	OPERAND

→ It will store the value of (Operand) the Regist
→ Instructions are stored In Register

Eg = | MODE | OPCODE | OPERAND |

LOAD R_1, R_2

Means R_1 and R_2 are Values of Register In which the storage of Valu both R_1 and R_2 will be in Register R_1.

$R_1 \leftarrow R_1, R_2$

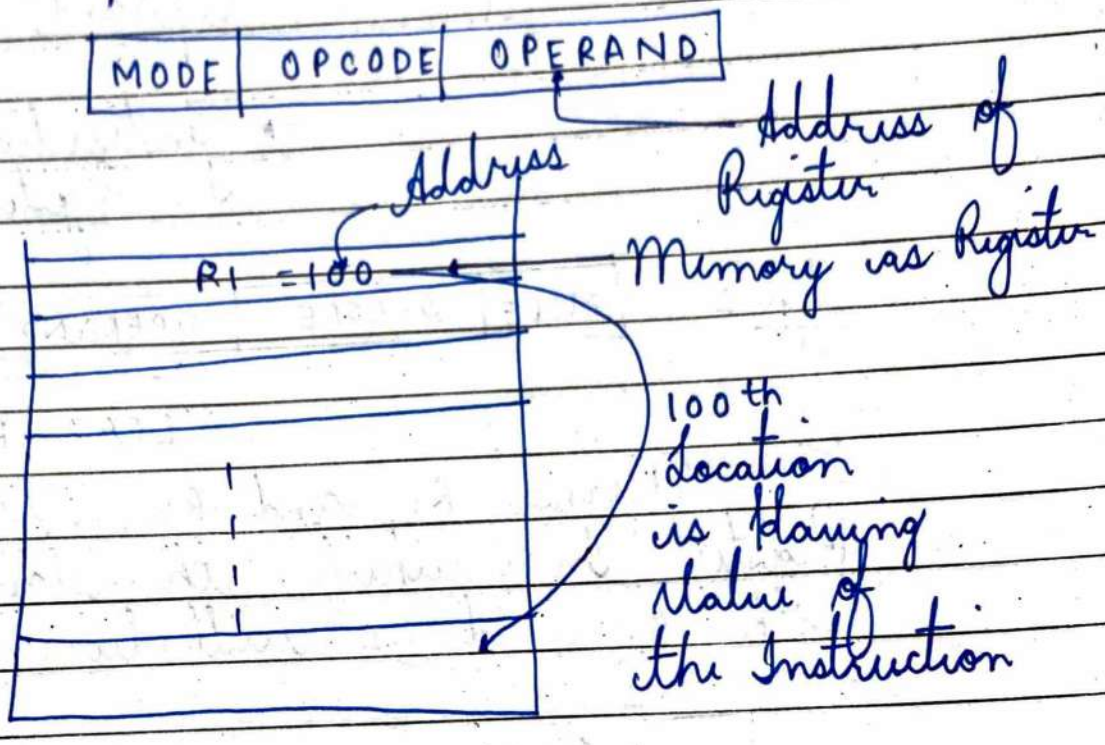
Q → LD R_1
AC ← R_1 (Value is getting stored in Accum

Advantages =
→ The size of Operand will get reduced
→ The process will become faster As re are fastest and timporary memory.

Lecture 23

* Register Indirect Mode

→ OPERAND having value of Register → Register having value Address of Instruction → Fetch Instruction to perform operations in OPCODE.



Q. LD R_1, (R_2)

Indirect Addressing of R_2

→ $R1 \leftarrow R_1, M[(R_2)]$

At memory location of value of R_2, Instruction is present.

Q. ADD (R_1), (R_2)

Register Indirect Addressing

→ $M[(R_1)] = M[(R_1)] + M[(R_2)]$

At The value must be stored at memory Address of R_1

→ The value at memory Address R_1 and Memory Address R_2 must be Added. (Before storing the value in R_1).

Advantages =

i) The size of OPERAND getting reduced.

ii) The process will be very fast. (Not As fast as direct because here, it is

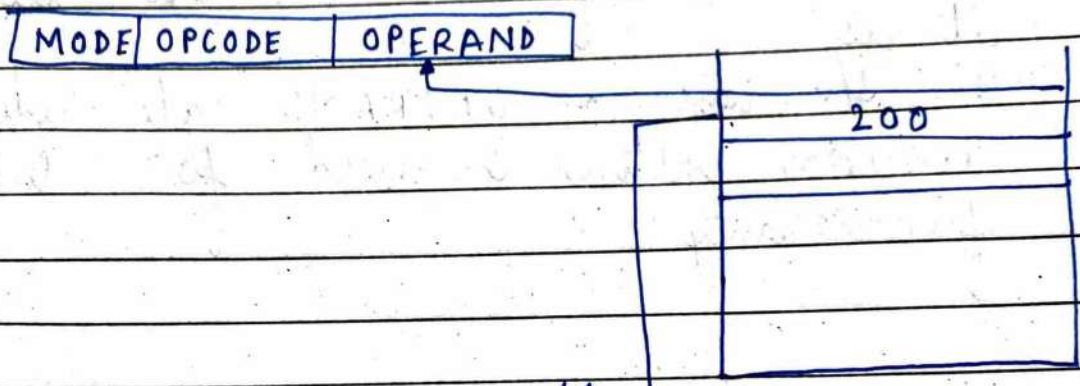
iii) The size of OPERAND gets reduced and memory getting Increased for better functionality.

Lecture 24

Auto - Increment and Auto - Decrement Addressing Modes =

→ The Extension concept of Register Indirect Addressing mode.

→ At particular time stamp, provides One memory location to the Register and the memory location will get Incremented or Decremented after the Instruction has been performed after particular time stamp.



LD ← R1 +; // Means the
AC ← AC + M[(R1)]; Address will get Incremented

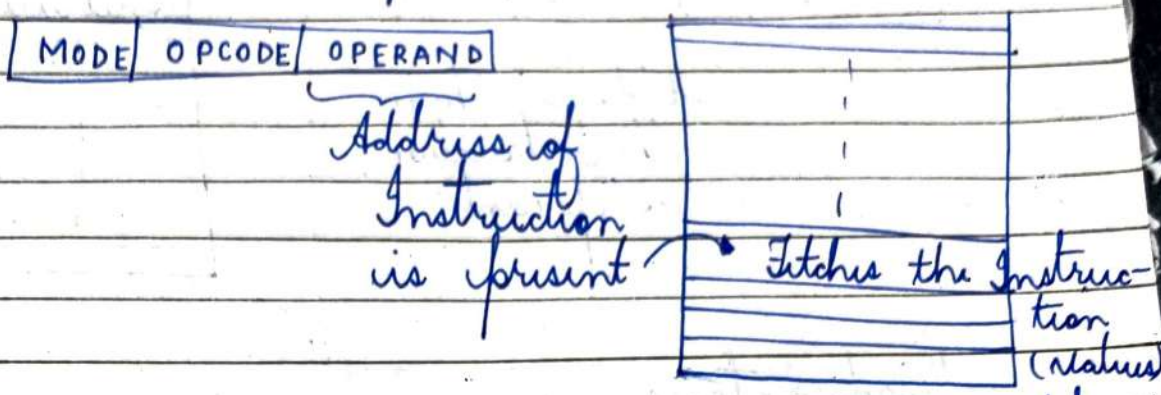
// Adding Value to the Register After performance of Instruction

Note = The Addressing is also done and will get Incremented by the time stamp after performance of Instruction without help of CPU or Extra micro and decrement is Also done the

Lecture 25

Direct Addressing Mode =

→ The OPERAND contains the Address of memory location in which the Instruction (values) are present.



Eg = OPERAND → x
and Addition to be done

⇒ AC ← AC + M[(x)]

Storage done in Accumulator

Value present in Accumulator

Value at Memory Location x.

Disadvantages =

→ Size Constraint.

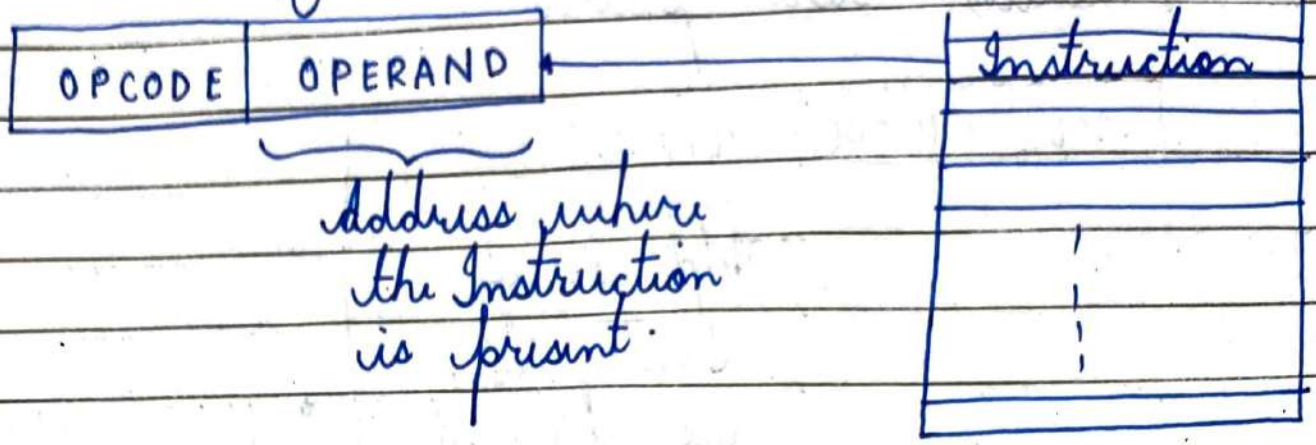
Lecture 26 ...

*** Most Important Topic

Direct Addressing Mode = (Absolute Addressing Mode) Address given in OPERAND.

→ The Instructions are present at Address

→ Use of Variables.



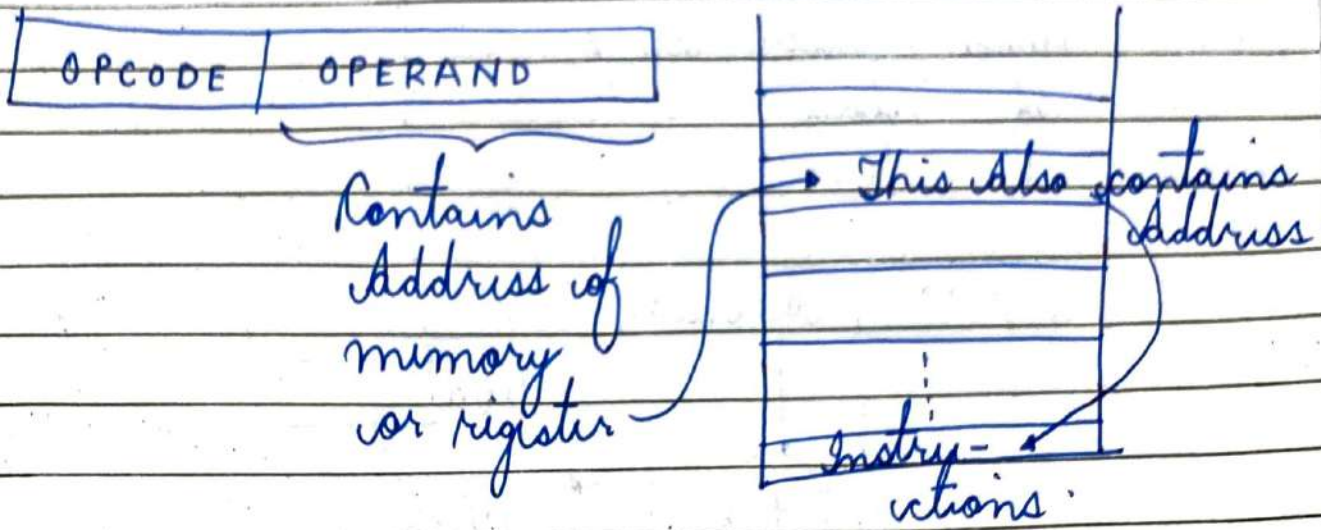
Disadvantages

→ Size is bounded according to the Address of the register.

Lecture 14

Indirect Addressing Mode :-

→ The OPCODE contains address which contains address of Instructions.
→ Basically the concept of pointers.



Q OPCODE → ADD
OPERAND → x

∴ Full Instruction becomes ADD x.

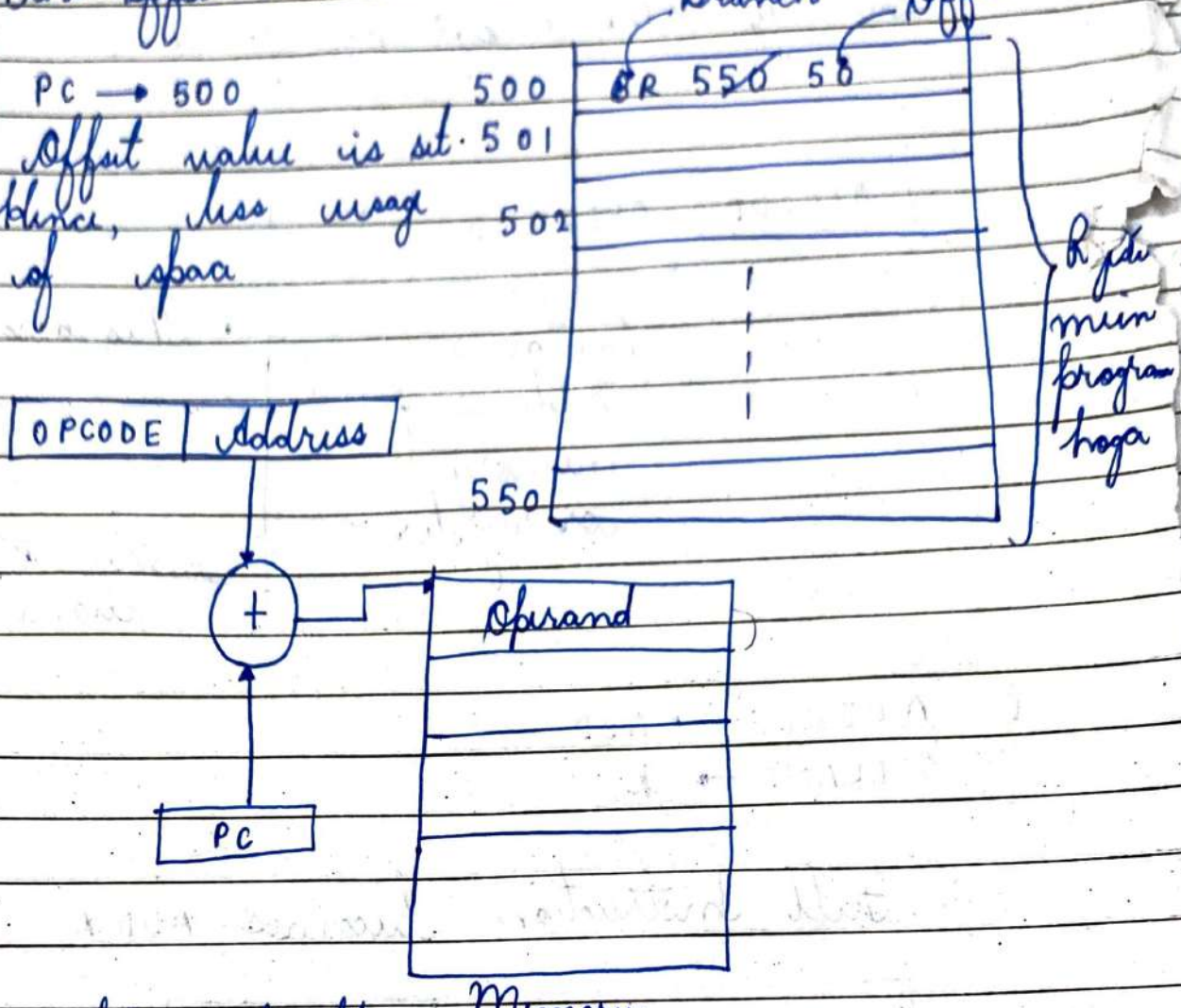
AC ← AC + M[M(x)]

All the working are done in accumulator.

Lecture 28

Relative Addressing Mode

→ The program counter works according to the offset value.



→ Offset value is set. less usage of space

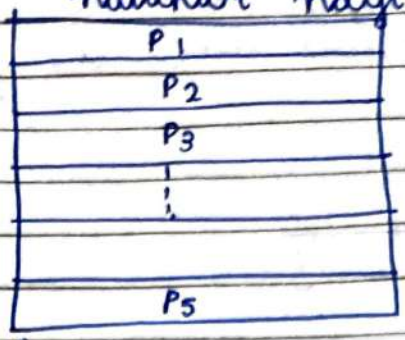
→ Jumping of the values.

EA = Base Register Value + displacement

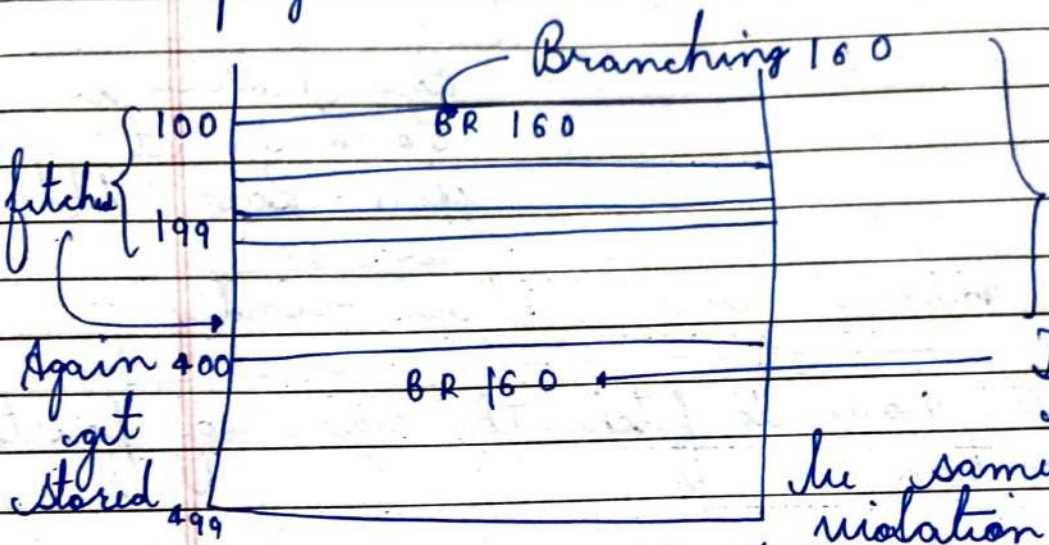
Effective Address where the actual Instruction is present.

Base Register Addressing Mode =

As Gandagi Yaa Unusu kii chiji hatakar nayi chiji dalo.



The phenomena is that when a Register or having program has performed all it's Instructions, then it is replaced by new program.



The two conditions cannot be same as the violation in security has been occured.

Hence, between memory locations 400 to 499 we should give offset Branched value as BR 60.

Note = Absolute Address ke bajaye sirf displacement deng.

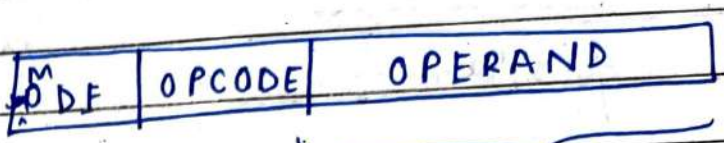
E-A = Base Register Value + Displacement

(400 + 60) = 460

Indexed Addressing Mode :-

→ Basically used for Implimentation of the Array Elements.

→ Use of multiple registers can be performed.



Base Address is stored in it here, Base Address is 100.

Effective Address = Base Address + Index

Like we have to fetch the value of fourth Index location

Effective Address = Base Address + Index

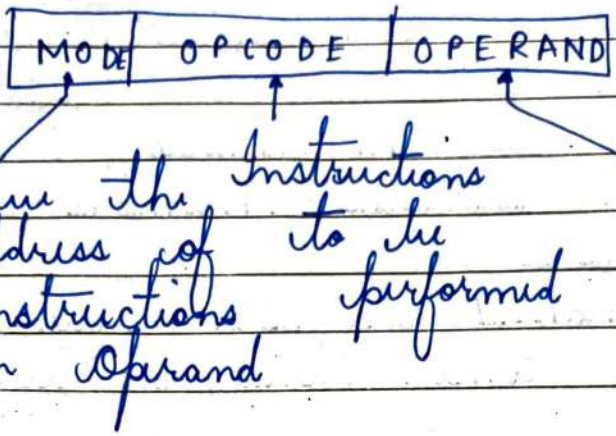
100 4

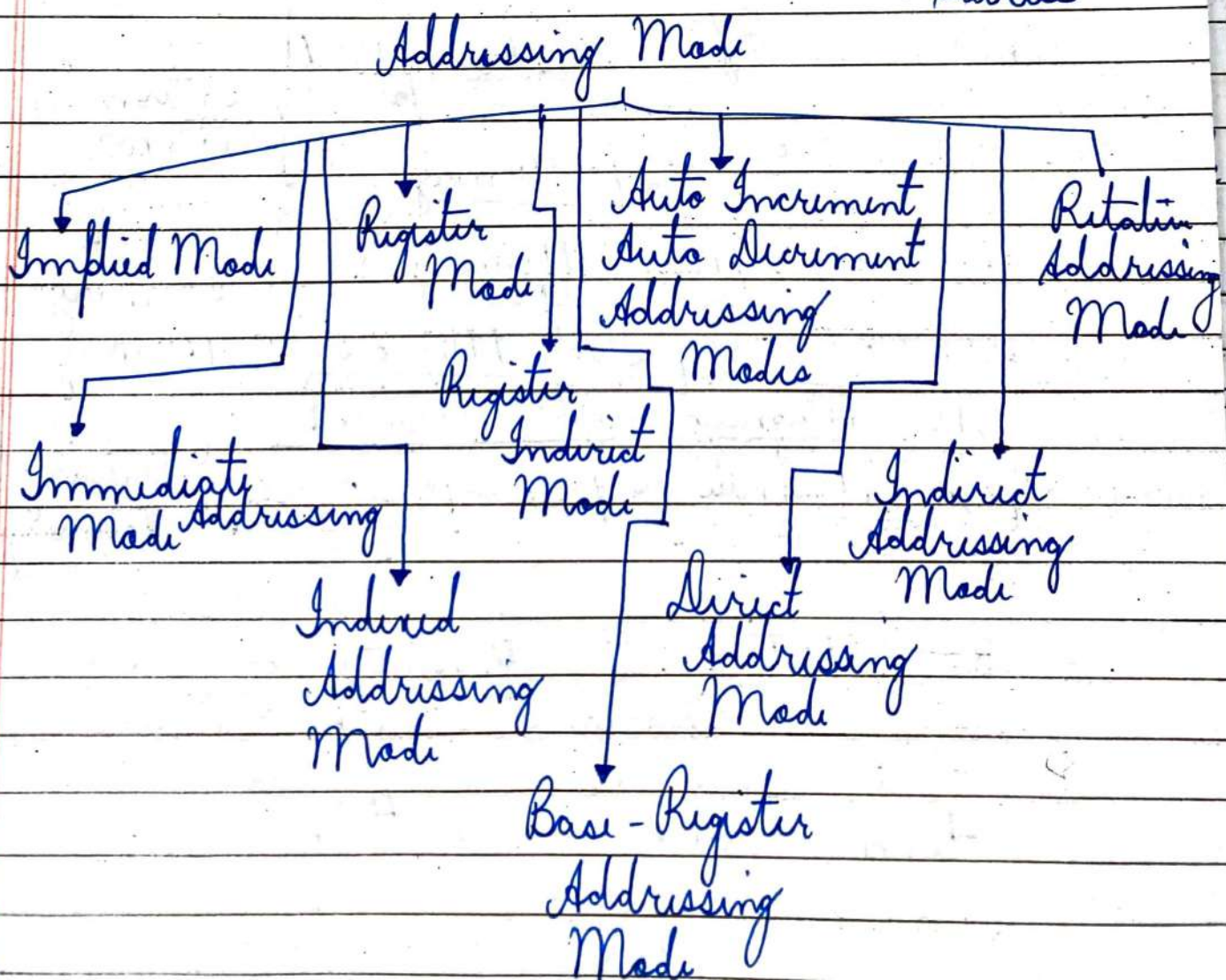
⇒ Effective Address = 100 + 4

→ Effective Address = 104

The address from where Instruction can be fetched.

Conclusion on Addressing Mode :-

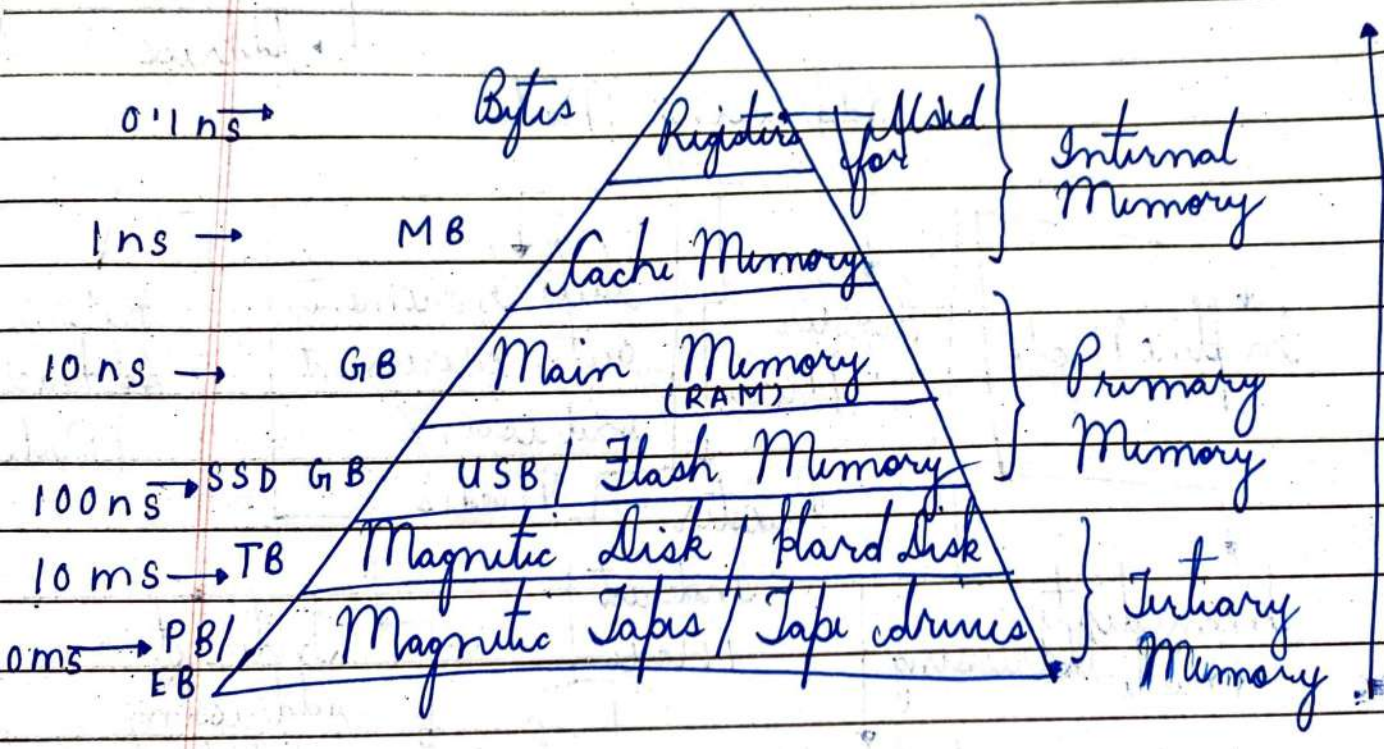




Lecture 31

MEMORY HIRARCHY IN COMPUTER ORGANIZATION

→ According to the theory, there are various memory storage based on the size.
→ The smallest in size will take less execution time. Hence, costlier.
→



Note = The frequency of usage also Increases with decrease in size and Time.
Hence, becomes costlier.

(i) Registers = Used for Temporary storage.
→ Connected to CPU.

ii) Cache Memory = Storage (SRAM)
iii) Main Memory = Used for processing. (DRAM)
iv) USB / Flash = Primary Memory.
v) Magnetic Disk / Hard Disk = Tertiary Memory
vi) Magnetic Tapes / Tape drives = Used to store backups, tertiary memory.

Since, backups are not used so often, hence the frequency of Usage is very low.

Lecture 32

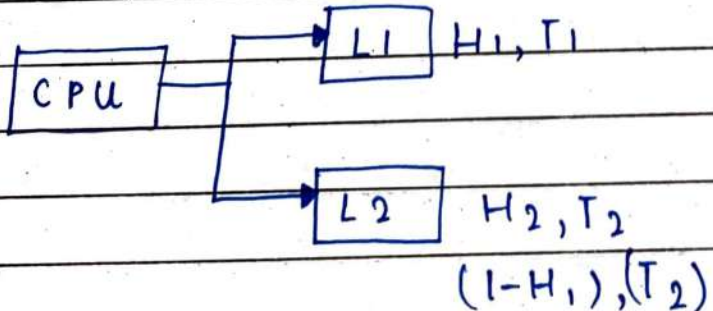
TWO-LEVEL MEMORY ORGANIZATION

→ Combination of Main Memory and Secondary Memory.

Independent Organization

→ It runs parallely as when it starts checking in first memory unit at the same time starts checking to another unit of memory.

→ Agar 1k mein nahi milega toh dusre mein hoga hii hoga.



$H_1 + H_2 = 1$

$H_2 = (1 - H_1)$

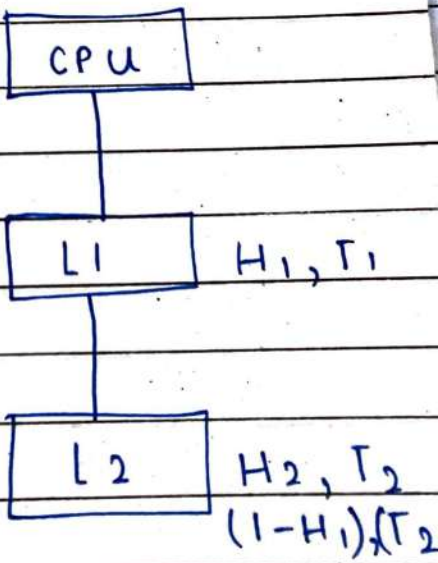
$T = H_1 \times T_1 + (1 - H_1) \times T_2$

as soon as successfull in ... then no need to proceed.

Hierarchical Organization

→ It runs in series.

If information is not present in one memory level then after one will move to another memory level.



$H_1 + H_2 = 1$

$H_2 = (1 - H_1)$

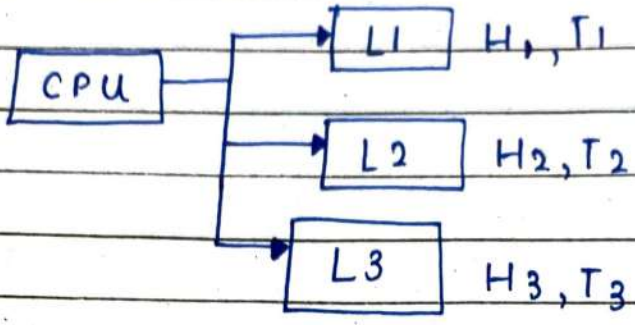
$T = H_1 \times T_1 + (1 - H_1)(T$

But, if not found in combination, one after
H1 and will be in another
H2.
then time accordingly.
(Worst Case)

Lecture 33 ...

THREE - LEVEL MEMORY ORGANIZATION

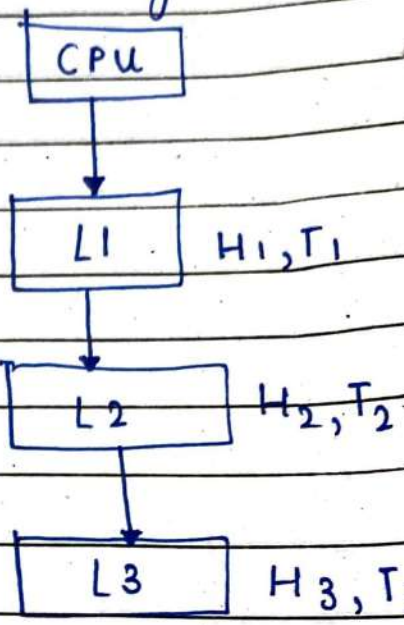
Independent Organization =



$$T = H_1 \times T_1 + (1 - H_1) \times H_2 \times T_2 + (1 - H_1)(1 - H_2) \times T_3$$

→ Program Running parallely.

Hierarchical Organization

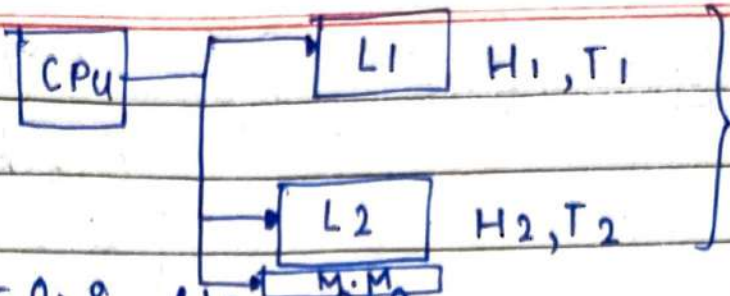


$$T = (1 - H_1) \times T_1 + (1 - H_1) \times H_2 \times (T_1 + T_2) + (1 - H_1) \times (1 - H_2) \times (T_1 + T_2 + T_3)$$

→ Program Running One After Another.

Lecture 34 ...

Q =.



$H_1 = 0.8$, $H_2 = 0.9$

$$\Rightarrow T_{avg.} = H_1 \times T_1 + (1-H_1) \times H_2 \times T_2 + (1-H_1) \times (1-H_2) \times T_3$$

$$\Rightarrow T_{avg.} = \{0.8 \times 1 + (1-0.8) \times 0.9 \times 10 + (1-0.8) \times (1-0.9) \times 500\}$$

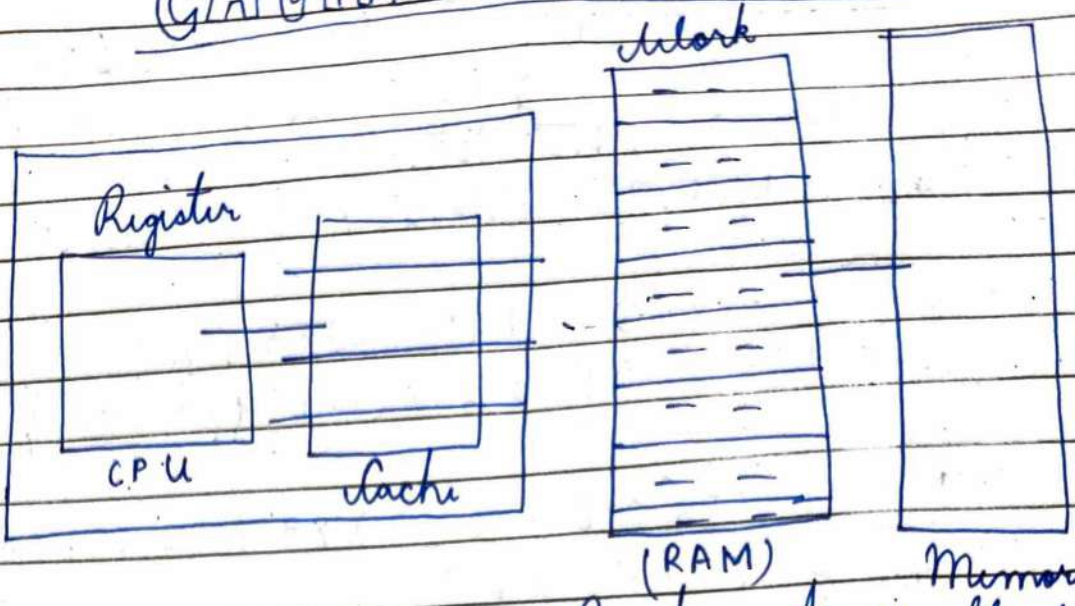
$$\Rightarrow T_{avg.} = (0.8 + (0.2) \times 9 + (0.2) \times (0.1) \times 500)$$

$$\Rightarrow T_{avg.} = 12.6 \text{ ns}$$

H.W. If Not = If there no Information is given then priority will be set to Independent Method.

Lecture 35 ...

CACHE MAPPING



→ The full program must be present in Memory Disk or Hard disk according to the size.

→ Then some of the Required Instructions are transfered to the words of Primary Memory (RAM = Random Access Memory).

→ Then the suitable Instructions or data from RAM is getting transfered to cache as per the suitable location.

→ They Gets Stored in register and hence the Instruction having data gets processed in CPU.

Conclusion =

→ Memory Disk / Hard Disk (Secondary Memory) → RAM (Random Access Memory) Primary Memory

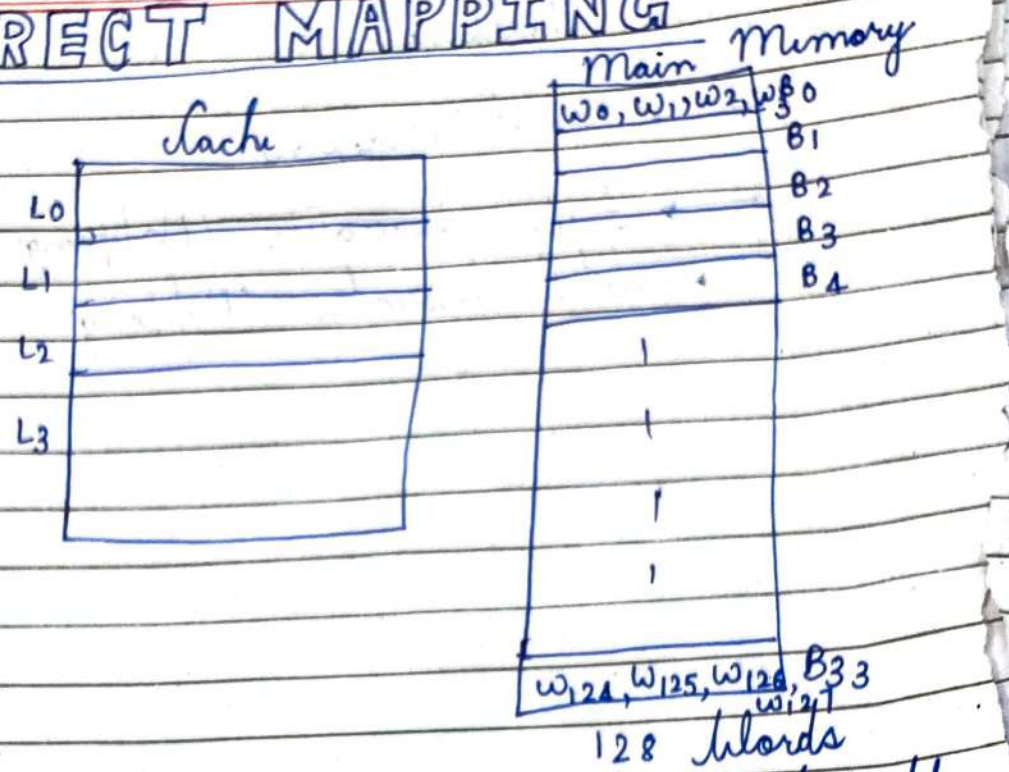
Registers (Internal Memory) ← Cache (Internal Me

Cache Mapping =

- Direct Mapping
- Fully Associative Mapping
- Set Associative Mapping

Lecture 36 ....

* DIRECT MAPPING



Let us assume 4 bits per words, then memory locations in main memory = $\frac{128}{4} = \frac{32}{}$

In cache memory =

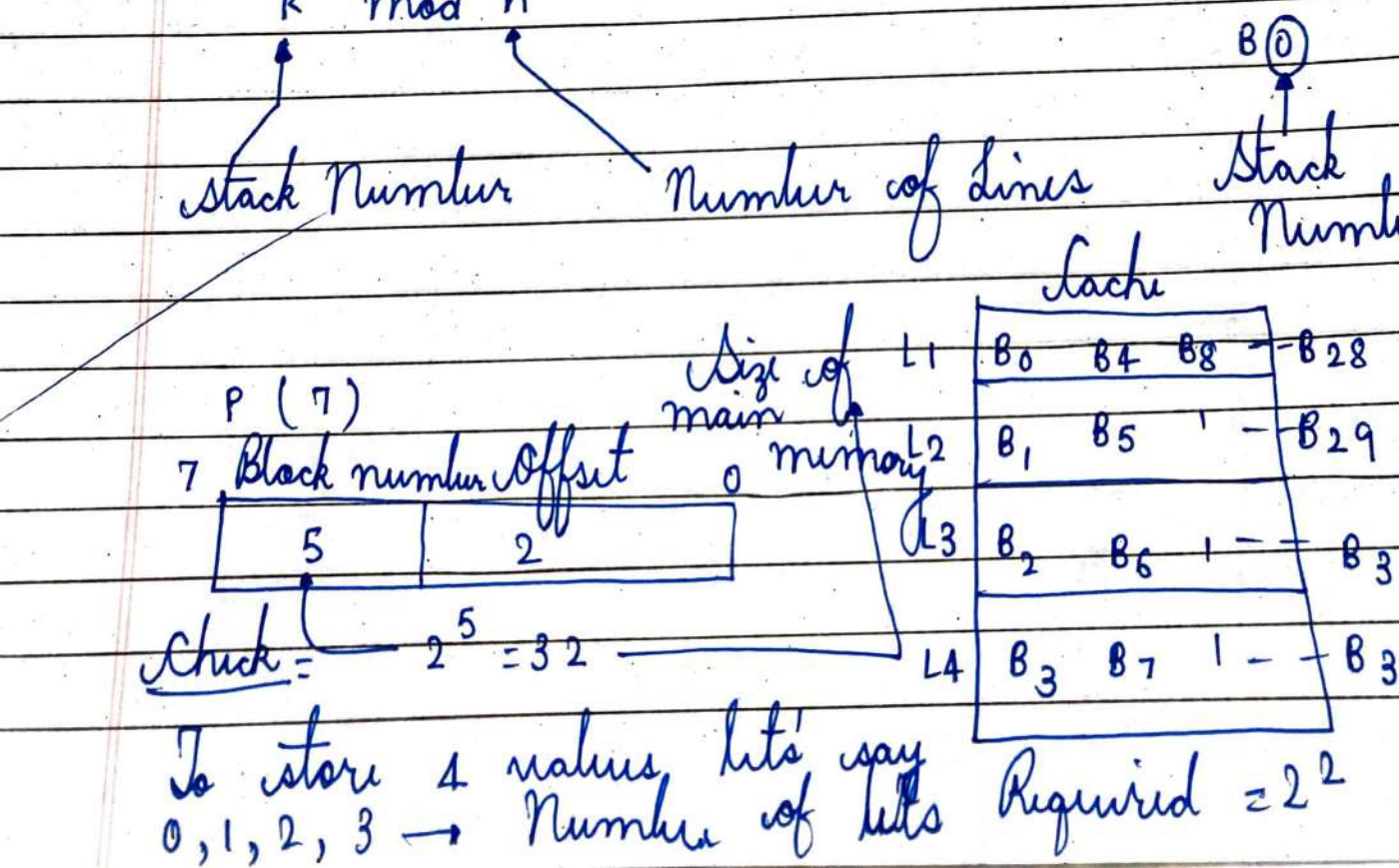
K mod n

Stack Number

Number of Lines

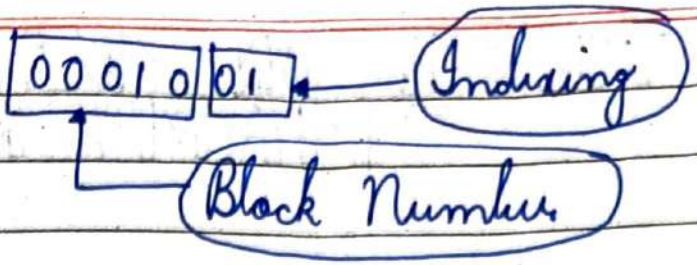
B(0)

Stack Numb

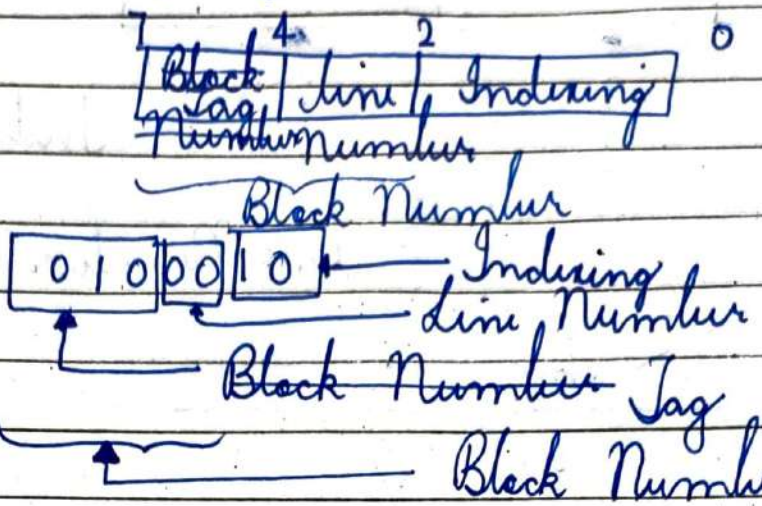


Chuck = $2^5 = 32$

To store 4 values, lets say 0, 1, 2, 3 → Number of bits Required = 2^2



Similarly for cache memory

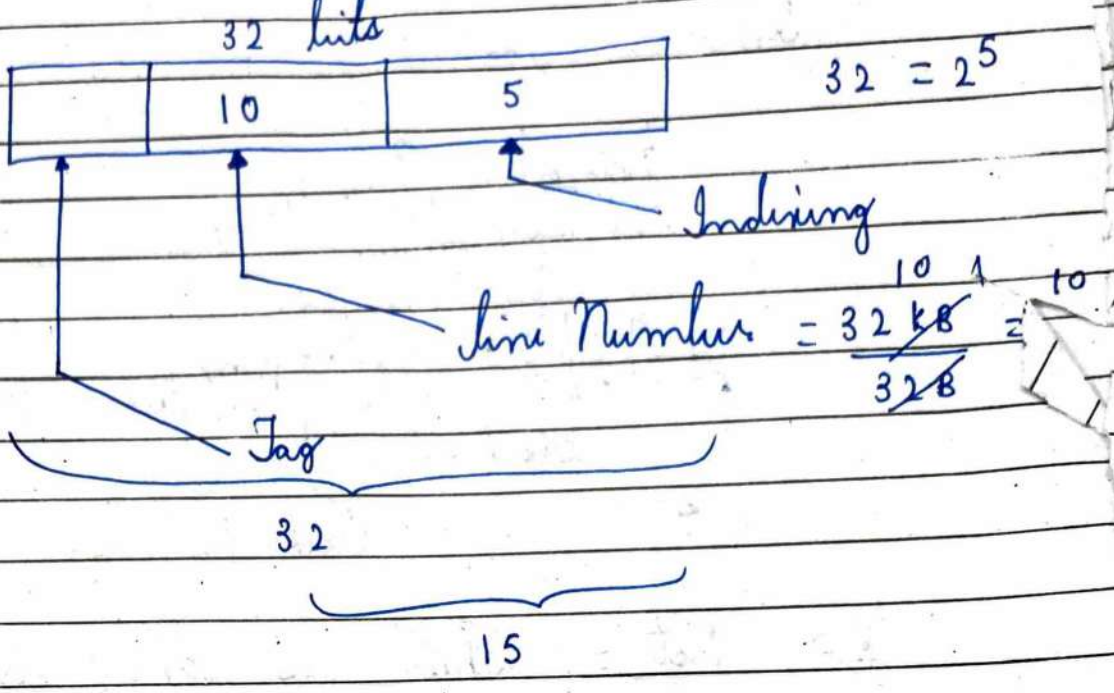


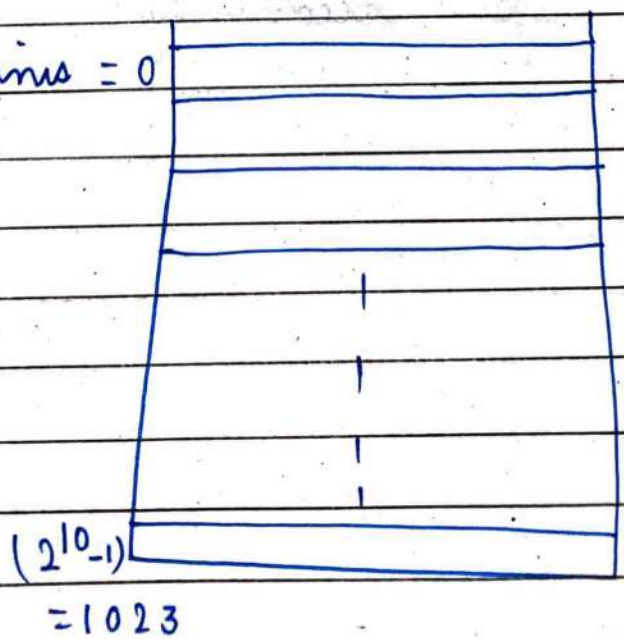
☞ Note = There is one major drawback that we are storing blocks at fixed memory locations (lines) in cache memory.

Eg = B0 → L0 and the nomenclature goes accordingly.

Lecture 37 ...

Q. QUESTION ON DIRECT MAPPING

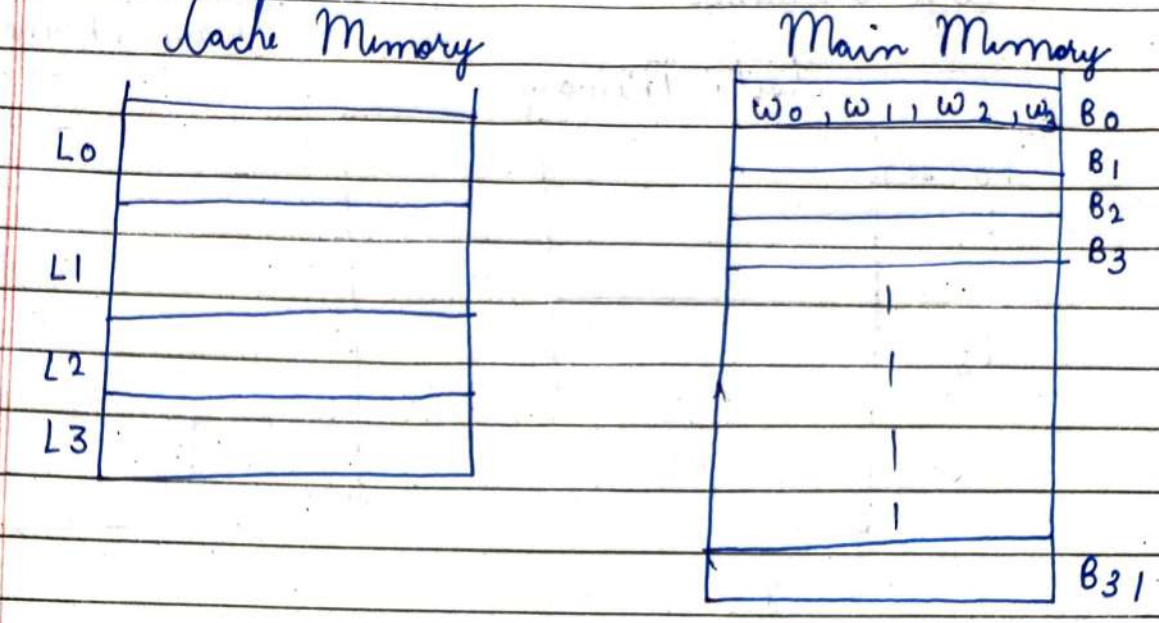




Number of lines = 10
Number of Tags = 17

Lecture 38 ...

FULLY ASSOCIATIVE MAPPING



128
4 bits per blocks
Number of lines = $\frac{128}{4} = 32$

Now, here the blocks can be placed at any lines.

→ If blocks are placed anywhere, then line bits can be removed.

Tag	Offset

Block Number

Here, Block Number = Tag

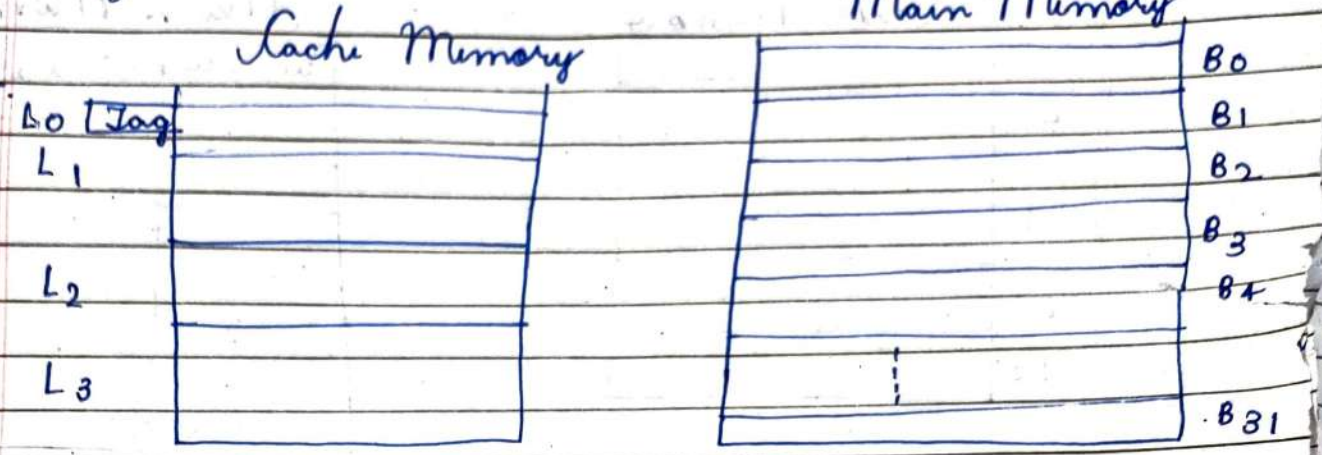
Block Capacity = Number of lines

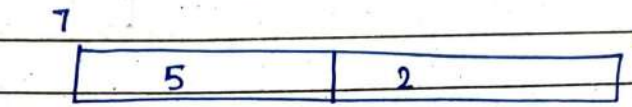
• Drawbacks =

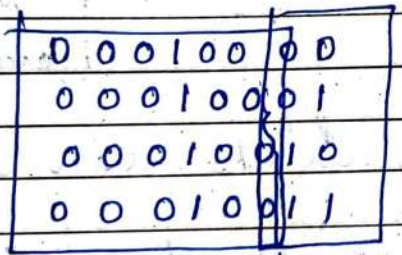
→ Comparisions becomes more. Hence, time taking.

Advantages =
→ Number of hit and trials becomes more.
→ Size Increases

Q =







Tag offset (Number of Comparisons

00 =
01 =
10 =
11 =

0 0 0 1 0 0 → 5th Block → B4 and having 4 number of comparisions.

ecture 39 ...

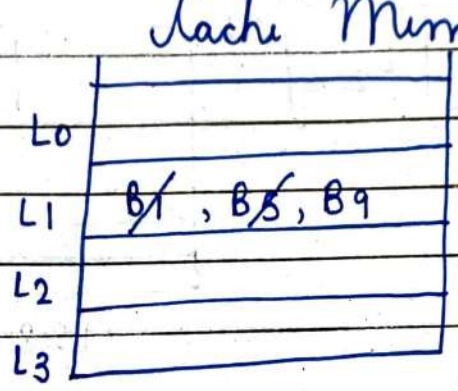
ADVANTAGES AND DISADVANTAGES

Advantages =

→ Searching the blocks becomes easier.

Disadvantages =

→ The hit and miss process occurs very often



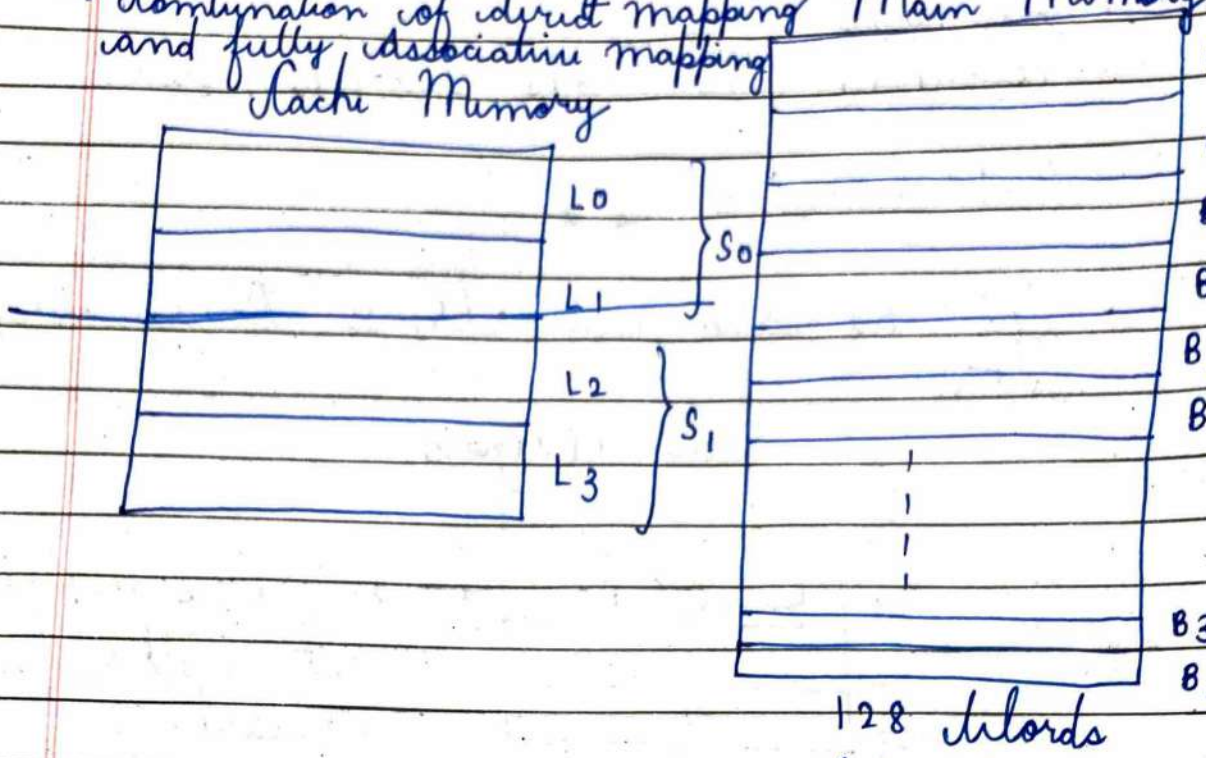
← If we search B1, I
It will get missed.

→ Hence, positioning is defined.
→ We could not place the blocks anywhere
Instead of the empty position.

Lecture 40 ...

SET ASSOCIATIVE MAPPING

→ The Cache Memory is divided into sets
→ Combination of direct mapping and fully associative mapping



4 lines per words
⇒ Number of Blocks = $\frac{128}{4} = 2^5$

Number of blocks
0 − 31

Number of sets = $\frac{\text{Number of lines}}{\text{Number of ways}}$

Lets Say we have 2-way Method
i.e. A/q,

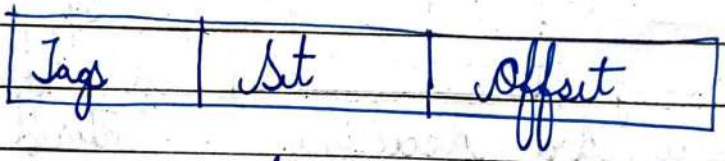
Number of sets = $\frac{4}{2} = 2^2$ sets

Now,
Mapping

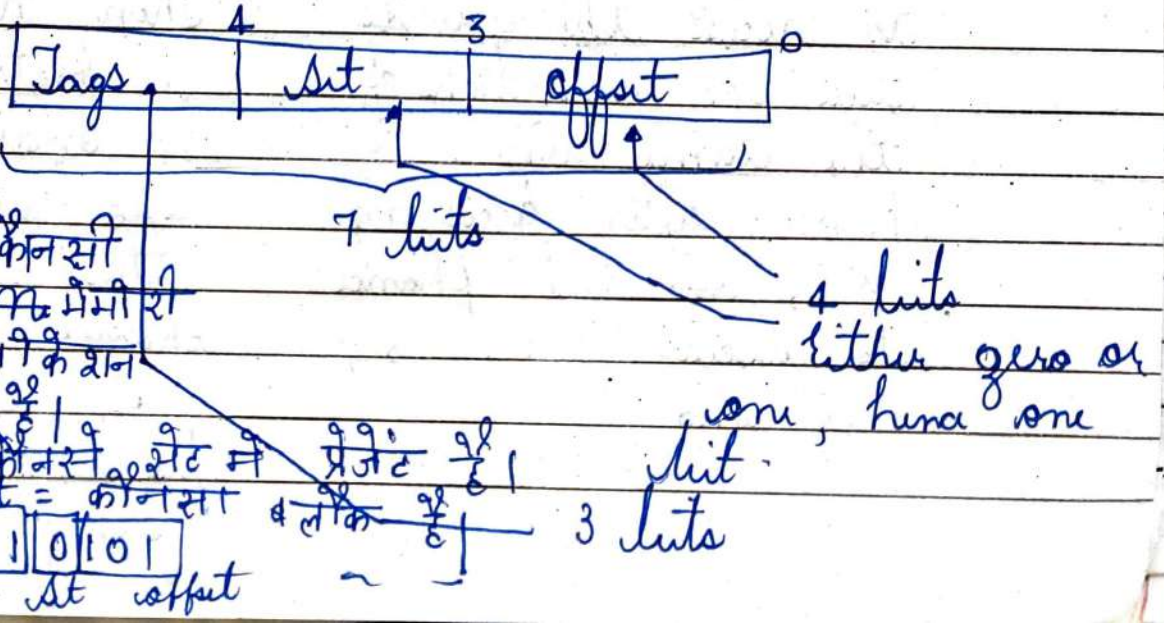
Direct Mithod
k map n
0 map 4 = 0 ← which means Anywhere In set 0.
(Hina, Here here comes fully Associative mapping)

Similarly for B1
k map n
1 map 4 = 1 ← Anywhere in set 1.

→ Hina, by direct mapping mithod we will decide whether it is set hit or set miss.



Q. Let's say we have an Instruction Register of 7 bits and 4 bits In each block

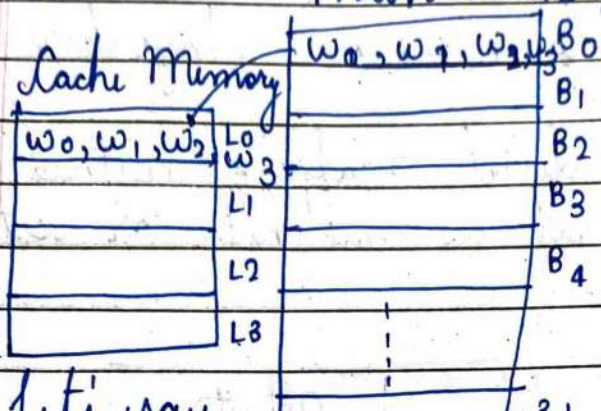


Lecture 41

LOCALITY OF REFERENCE

Spatial Locality = (Close proximity)

→ If the word is accessed now, then there is high priority that the words adjacent to it will be accessed next.

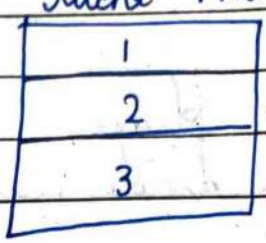


Let's say if we are accessing w_2, then there is very high chance to access the words next or instructions of the same line. Hence, after accessing w_2, we high chance to access w_1, w_3, w_0 and w_1.

Temporal Locality:

→ If any word is referenced now, then same word will be referenced Again.

Eg = If

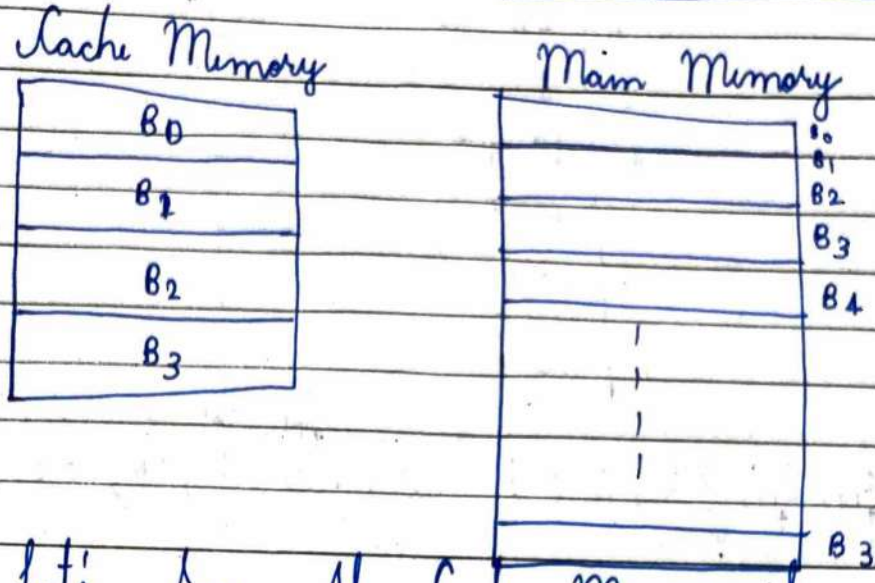


If One is Entered at first then Two is Entered or Accessed and then three is Entered or Accessed. Then next word or Instruction will be Inserted at block 1, not here at 3 because there is very high chance that words of block 3 will be getting Accessed or Accessed.

→ Funda of LRU (Least Recently Used).

Lecture 42 ...

CACHE REPLACEMENT POLICIES



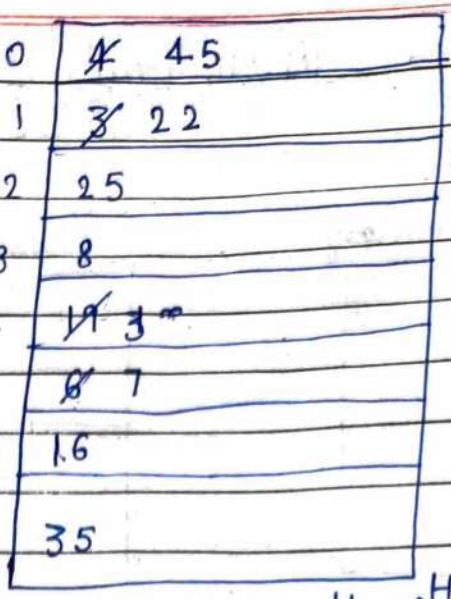
→ Let's Say If Cache Memory becomes full and gets filled by limited Amount of blocks.

→ If we want to transfer some more blocks from main memory to cache memory then we need to empty cache memory. So, Now, this can be done on the basis of various factor such As =

- FIFO → (First In First Out)
- LRU → (Least Recently Used)
- MRU → (Most Recently Used)
- (Random)

Lecture 43

Q.



4, 3, 25, 8, 19, 6, 25, 8, 16, 35, 45, 22, 8, 3,
16, 25, 7

At highest 4th distance is 4. Hence, will replace 4

→ Steps =

(i) First we will fill according to direct cache Access

(ii) Then we will check hit or not.

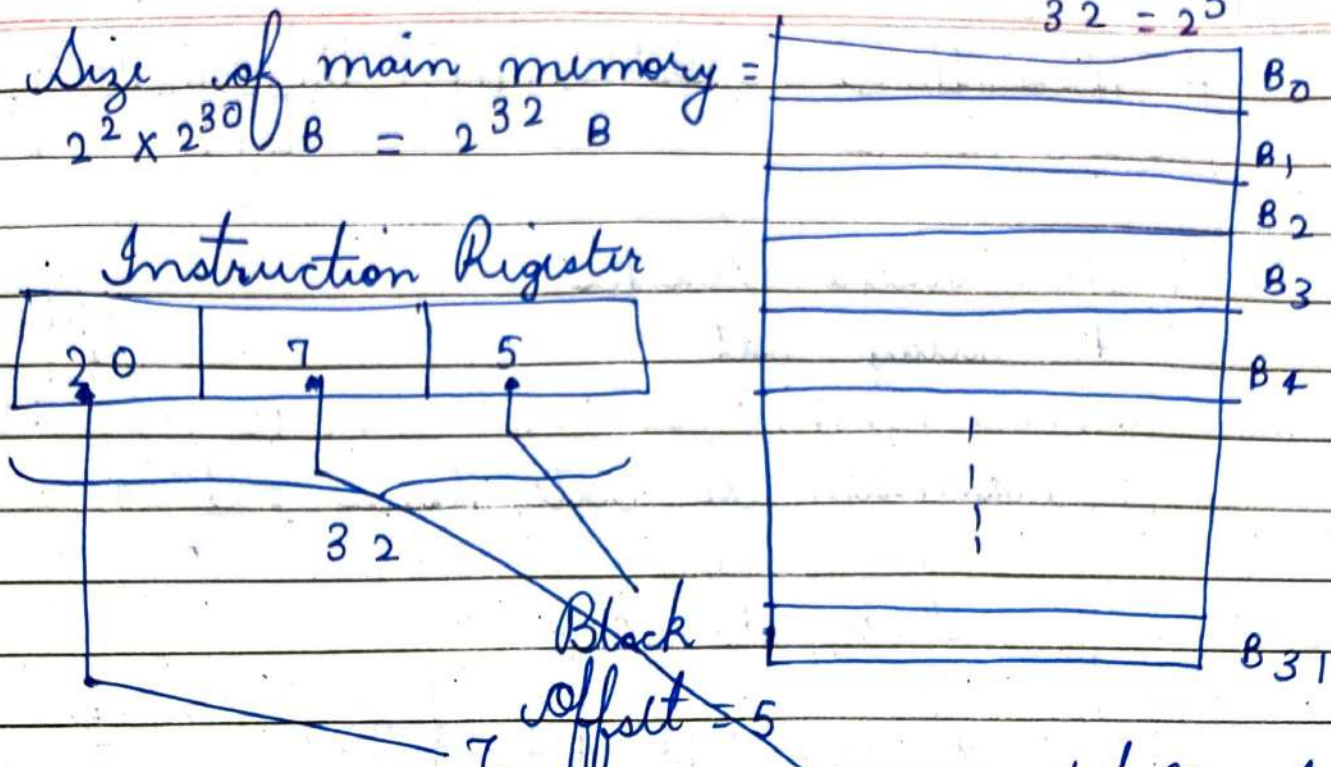
(iii) Then after we will check maximum distance element from the element which is to be placed and will remove that and place a new.

→ Hence, 7 will occur at Sixth block i.e., (that is) Line 85.

Note = $1\,GB = 2^{30}\,B$, $1\,MB = 2^{20}\,B$, $1\,KB = 2^{10}\,B$, $1\,TB = 2^{50}\,B$

Lecture 44

Q. Size of main memory = $2^2 \times 2^{30}$ B = 2^{32} B



$$\frac{16\text{ kB}}{32\text{ B}} = \frac{2^4 \times 2^{10}}{2^5} = 2^9$$

$$\frac{2^9}{4} = \frac{2^9}{2^2} = 2^7 \rightarrow \text{Number of Set}$$

$$32 - (7+5) = (32 - 12) = 20$$

Lecture 45

FIFO CACHE REPLACEMENT POLICY

Q 16 Cache blocks
4 way set
0, 255, 1, 4, 3, 8, 133, 159, 216, 129, 63, 8, 48, 32, 73, 92, 155, 1
∴ Number of set lines = 16/4 = 4 set lines

Hit

Cache Memory

S_0	0	4	8	216	48	32	92
S_1	1		133	129	73		
S_2	2						
S_3	255		3	159	63	155	

B mod n

Block

Num of lin

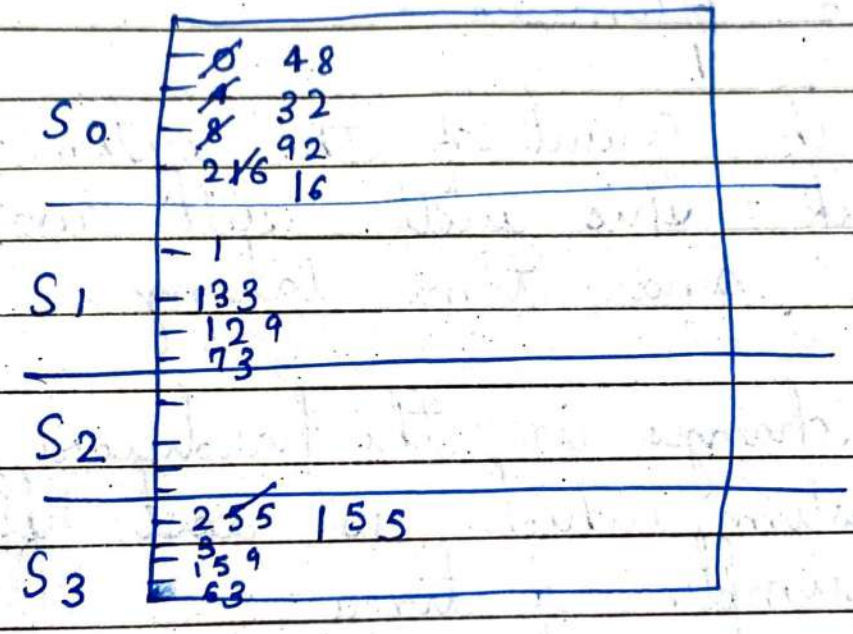
Steps =
→ Find count mods and then place in set lines according to capacity.
→ If capacity is full then replace an El according to FIFO.

70

Lecture 46

LEAST RECENTLY USED POLICY

Q. 0, 255, 1, 4, 3, 8, 133, 159, 216, 129, 63, 8, 48, 32, 73, 92, 155, 16



Lecture 47...

NEED OF PIPELINING

Advantages :=

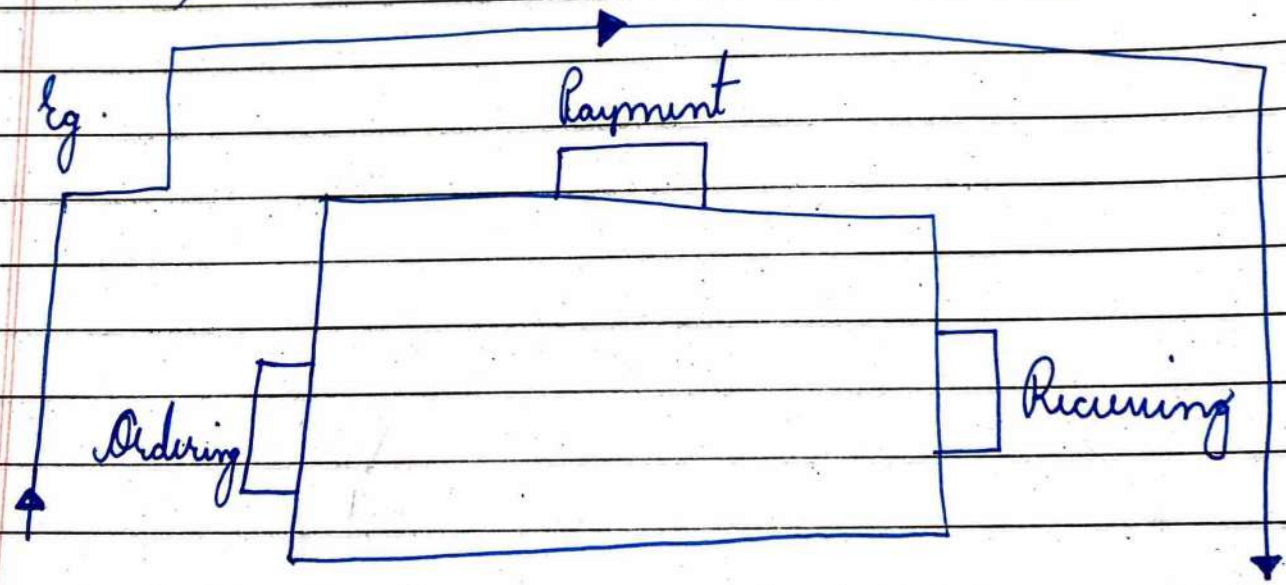
→ Reduces time of processing.

There are two methods to make our System faster and Optimistic :=

i) Change the circuit of the system.

→ Drawback = This will result more costlier and time taking.

ii) Doing changes in the hardware and software of the system which is cost effective and less consumption of time.
(Here, the concept of pipelining has been used.)



If we the performance is in pipelining then

Time Consumption	C_1	O	P	R		
	C_2	–	O	P	R	
	C_3	–	–	O	P	R

(12)

Date / /
Page

Within performance of not pipelining

		O	P	R			
Time	C_1	O	P	R			
Non-	C_2	—	—	—	O	P	R
consumption	C_3	—	—	—	—	—	—
high							C_P

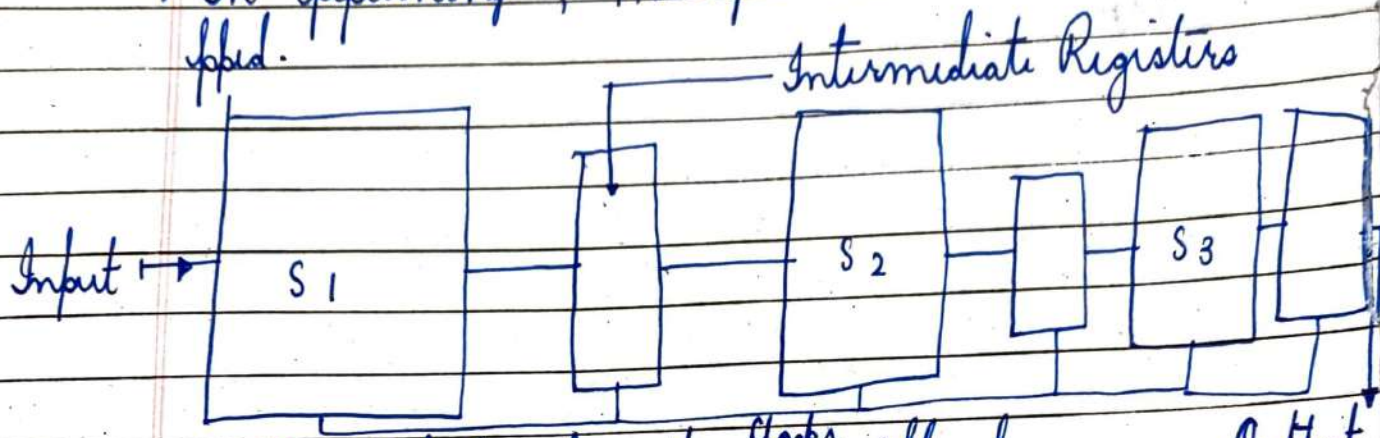
Lecture 48 ...

DEFINITION OF PIPELINING

→ The setup of hardware and Operating system such that working of the system becomes faster.

→ The simultaneous working of the system.

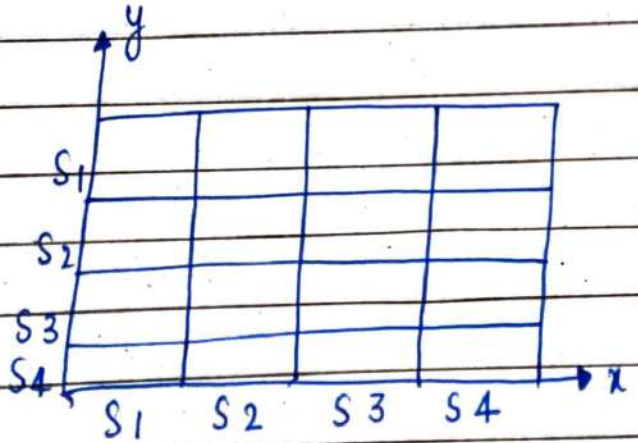
→ In pipelining, multiple Instructions are overlapped.



→ The Input of stage 2 will be the Output of stage 1.

→ For further processing, the Input is first stored at Intermediate Registers.

→ The clock gives the signals that when the Instructions has to get Executed.



Further it is divided into space signals.

Lecture 49

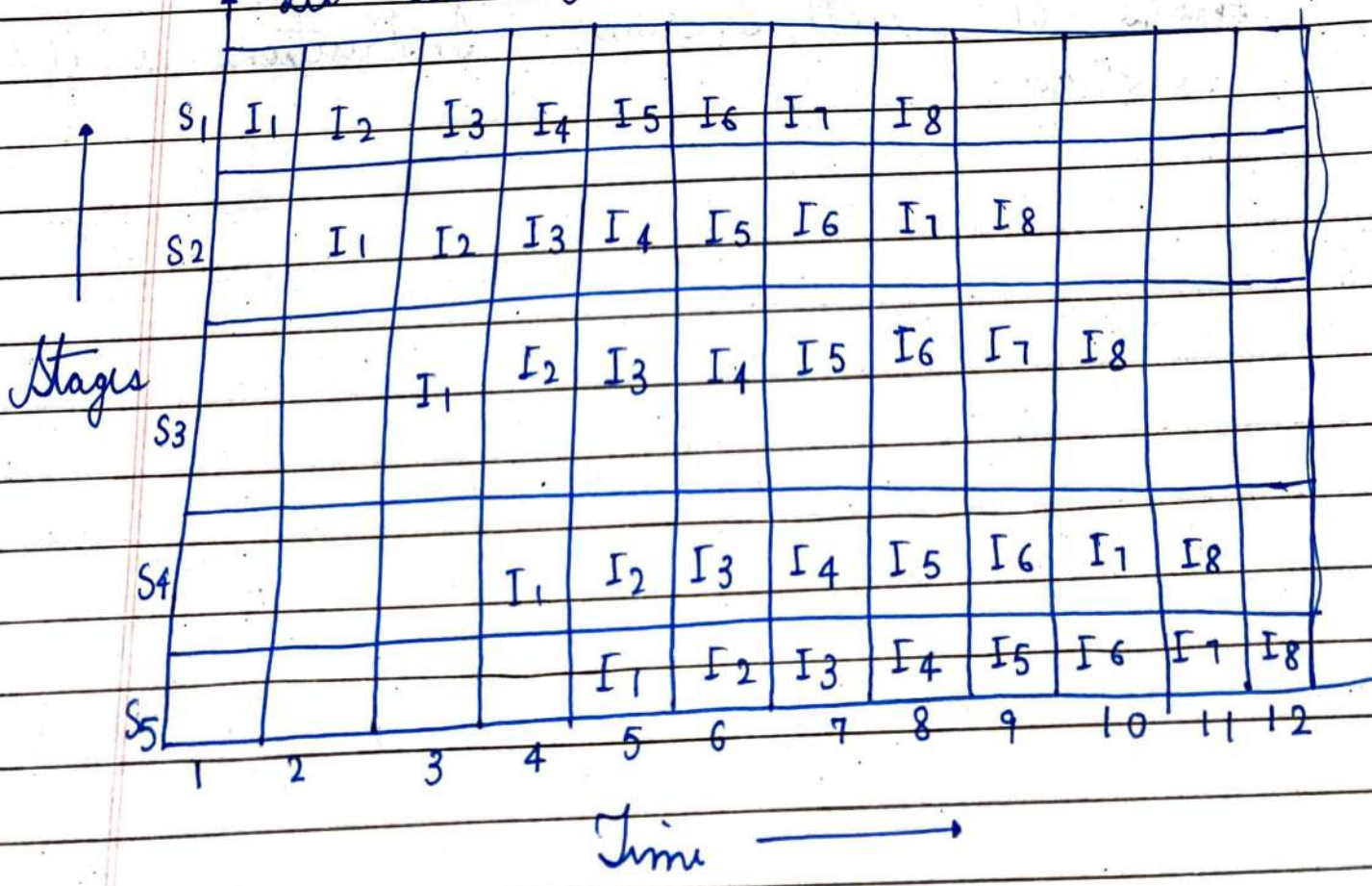
PIPELINING VS NON-PIPELINING

Q. P = 8I , $I_1 - I_8$

Input Function → IF
Input Decoding → ID
Execution → Ex
Memory → MEM
Write Back → WB

If By Non-Pipelining then
Clock Pulse = (8 X 5) cc
⇒ Clock Pulse = 40 cc

By Pipelining
Let Us Assume Clock pulse per Instruction ≃ 1



Note = Speed Up = $\frac{NP}{P}$

Speed Up $= \frac{40}{12} \frac{20}{6} \frac{10}{3} = \frac{10}{3}$

Time Taken In Pipelining $= n$

$\swarrow \quad \searrow (n-1)$

$1 \times k$

$= \boxed{k + (n-1)}$

Where, k is number of sets and n is number of Instructions.

* Efficiency $= \frac{\text{Used Boxes (Sets} \times \text{Number of Instructions)}}{\text{Total Number of Boxes}}$

$\Rightarrow \boxed{\text{Efficiency} = \left(\frac{8 \times 5}{60}\right)^{2} \frac{}{30} \frac{}{15} \frac{}{3} = \frac{2}{3}}$

Note: As CPI will Increase, Instructions will also Increase.

Lecture 51

NUMERICAL QUESTION ON PIPELINING

Q. $$\text{Speed up} = \frac{T_{NP}}{T_P}$$

$$\therefore \text{Speed up} = \frac{T_{NP}}{T_P} = \frac{\left(\frac{1}{F_{NP}} / \frac{1}{F_P}\right)}{\frac{4 \times 1}{2.5 \times 10^9} \Big/ \frac{1}{2 \times 10^9}}$$

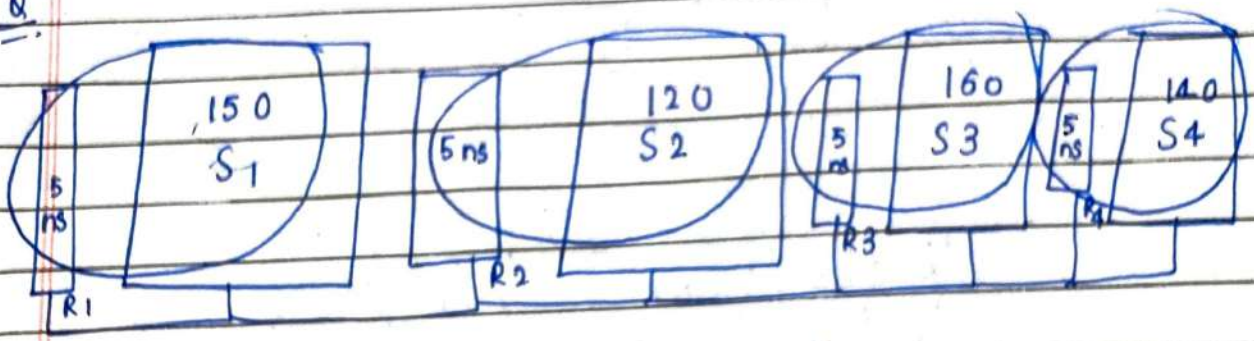
$$\Rightarrow \text{Speed up} = \left(\frac{4 \times 2}{\frac{5}{5}}\right) = \frac{16}{5} = 3.2$$

74

Lecture 50...

STAGE DELAY IN PIPELINE

Q.



Maximum Time = 165

∴ The most frequency Required = $\frac{1}{t} = \frac{1}{165}$

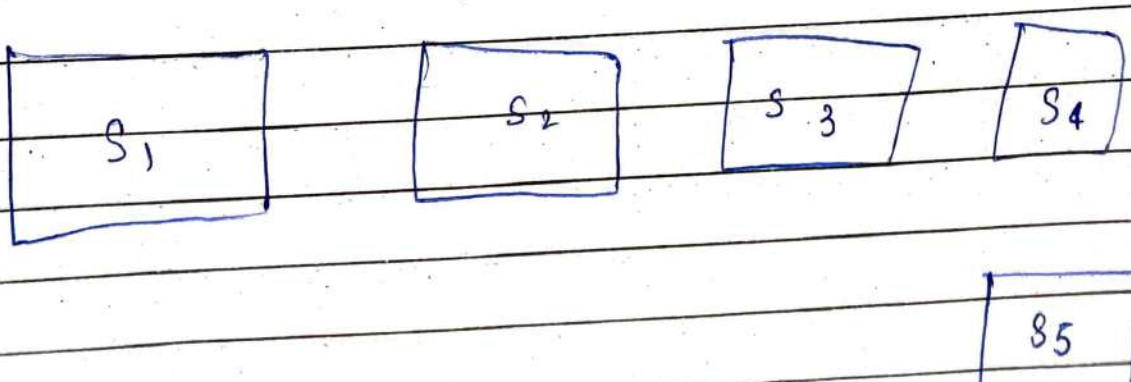
~~In~~ Clock Rate = $(1 \times 4 \times 165 + 1 \times 999 \times 165)$

= 165.50 ns

~~Bar~~ द्वा ६ 311 x

~~Total~~ Maximum

$(k + n - 1)\, t_p = (8 + 99 - 1)\, 2$ = 214

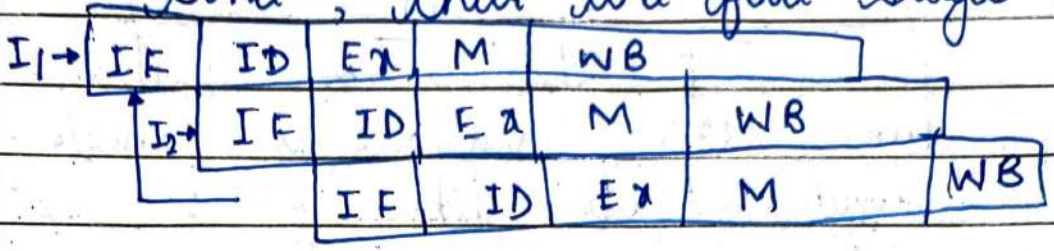


Lecture 53 ...

STRUCTURAL HAZARDS

→ When Instructions requires various resources like Registers and many more hardware resources for processing, then there structural hazards occurs.

Since, there are five stages in Pipelining



Now, Here The contradictions in performing same functions Instructions occurs, then, their either we have to change the whole circuitry or needs various registers so that Information can get stored.

Otherwise, cloud will be getting occured which will perform continuous miss. Rarely a hit will be occuring.

(18) Lecture 56 ...

(17) Lecture 55 ...

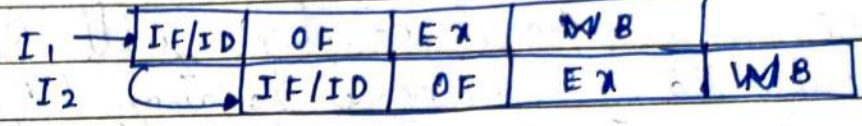
READ AFTER WRITE HAZARD

→ Let's Say

$R_3 \leftarrow R_1 + R_2$

$R_5 \leftarrow R_3 + R_4$

Four Phase



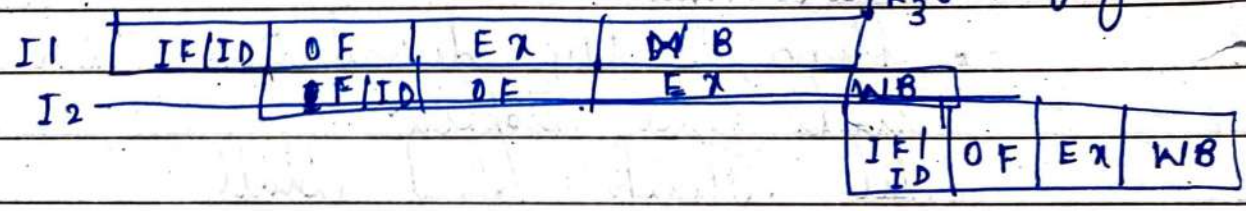
But,
Before the Final Execution of R_3 after the starting of Instruction I2.

→ In Instruction I_2, R_3 has been used.

→ Hence, this will produce contradiction.

Note = To resolve this contradiction Stall has been done

→ Instruction 2 (I2) will be getting flushed.



Branching In the Instruction has been used, stall.

→ Can get rectified by operator shifting method.

Lecture 54...

CONTROL HAZARD

500	IF	ID	EX	M	WB	
501	IF	ID	EX	M	WB	

Here while Execution of Memory Address 500, Program counter (PC) will automatically shift to Memory Location (501), Suppose If we want to Execute the Instruction of Memory Location 700.
Then that will perform contradictions. Hence, Pipelining raises contradictions for performance of Instructions.

Note = As soon as we detect branching we can shift the Instructions accordingly.
→ Hence, the degradation in our performance will be stopped.

Flush and Stall

Remove number of Instruction to reach.

If Program Counter (PC) in Register the Instruction to be displayed of

500	IF	ID	EX	M	WB					
		IF	ID	EX	M	WB				
			IF	ID	EX	M	WB			
								IF	ID	EX	M	WB

Let's say 2000th memory location, then flush

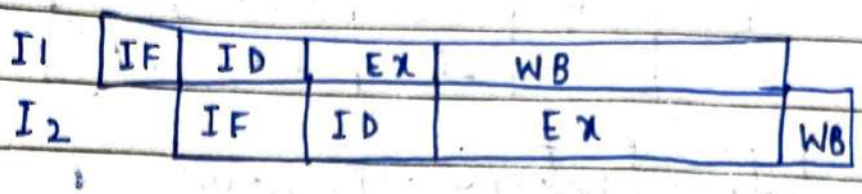
Lecture 56

WRITE AFTER READ HAZARD

Q

I_1 : $R_1 \leftarrow R_2 * R_3$ (700, 70, 20)

I_2 : $R_2 \leftarrow R_4 + R_5$ (70, 30, 40)



Here, Before giving the value of $R_2 * R_3$ Into R_1 Register of Instruction 1, The R_2 of Instruction 2 is reading the value of R_4 and R_5.

Hence, here write after Read Hazard Occurs

Rang $(I_1) = R_1$

Domain $(I_1) = R_2, R_3$

Rang $(I_2) = R_2$

Domain $(I_2) = R_4, R_5$

$\therefore$ Domain of $(I_1) \cap$ Rang of $(I_2) = R_2 \neq \phi$

Hence, Write after read Instruction is existing here.

Lecture 57...

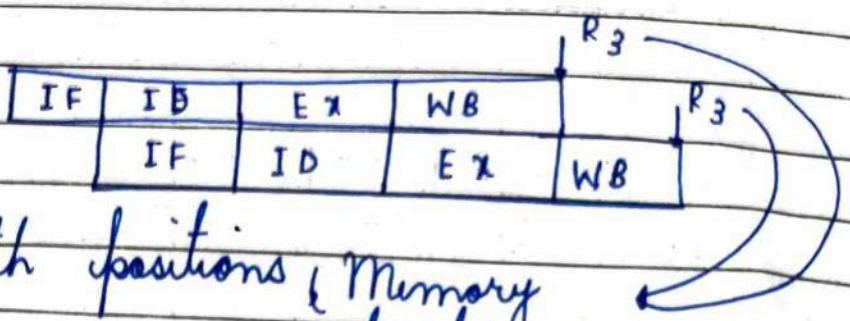
WRITE AFTER WRITE HAZARD

200 10 20

I1 = R3 ← R1 * R2

I2 = R3 ← R4 + R5

100 50 50



At both positions (Memory locations), the value of R_3 has been differing.

This is perfect example of Lost Update Problem.

LOST UPDATE PROBLEM =

→ When two or more transactions gets overriden then the error in output has been changes occured.

Check = Rang $(I_1) = R_3$

Domain $(I_1) = R1, R2$

Rang $(I_2) = R3$

Domain $(I_2) = R4, R5$

In Write After Write Hazard

$$\text{Read}(I_1) = \text{Read}(I_2) = R3 \neq \phi$$

Hence, Existence of Write After Write Hazard.

Note = Data Hazards

- Read After Write
  - → Can get rectified by shifting Operator
- Write after Write
  - → Can be rectified by Register Renaming.
- Write After R
  - → Can be done by register renami

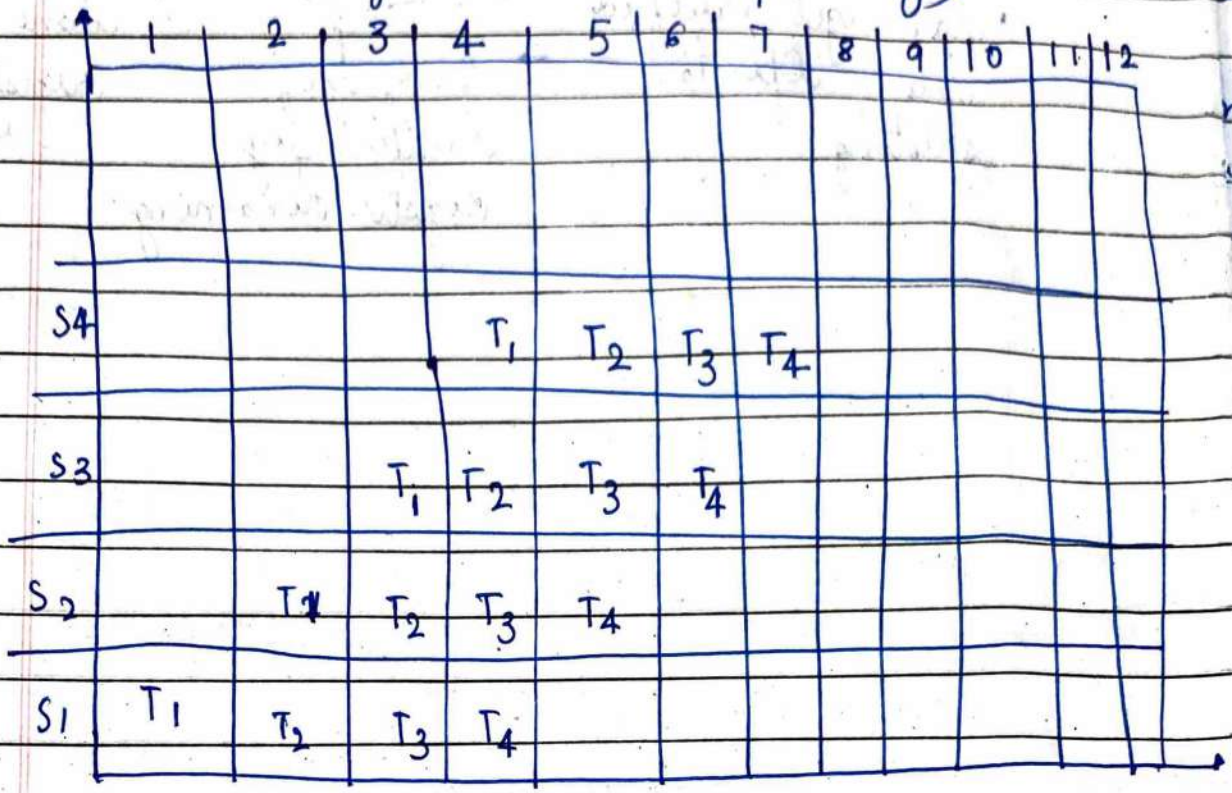
(21)

(Depth of pipelining = k

Q. Consider a 4-Segment pipeline for 4 tasks, compare the number of cycle required in pipeline model and non-pipeline model.

⇒ $k + (n-1) = 4 + (4-1) = 7$ Cycle (Pipelining)

$4 \times 4 = 16$ Cycles (Non-Pipelining)



(16-7) = 9 Stages Saved

⇒ $\{k+(n-1)\} \times t_p$ ← Pipelining

⇒ $n t_n$ ← Non-Pipelining

$$\eta = \frac{S}{R}$$

Sta