REGISTER TRANSFER LANGUAGE

* A digital system is an interconnection of digital hardware module to complete a specific information processing task
* It have a modular approach. i.e, it have a lots of modules constructed from digital Systems such as, Registers, decoders, control logic etc and are connected through common data & control path.

* A MICROOPERATION is an elementary operation performed on information stored in register. Eg - shift, count, clear, load etc.

* It may be possible that the new result replaces the previous result stored in Register or may be transferred to other register.

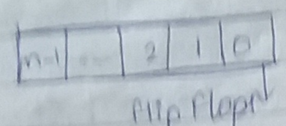
* The internal hardware of any Digital System takes 3 things :-
  (i) Registers & their functions
  (ii) Sequence of Microoperation
  (iii) control to initiate Microoperation.

* sequence of microoperations are specified by Symbolic Language known as Register Transfer Language
  ↳ expressing in symbolic form the microoperation sequence among the registers of a digital module.

REGISTER TRANSFER

Registers name are represented by capital letters to denote their functions

eg - PC (Program Counter)
MAR (Memory Address Register)



$R_2 \leftarrow R_1$

Transfer of information from R_1 to R_2

If we want transfer to occur under some defined condition, then use (if then) statement.

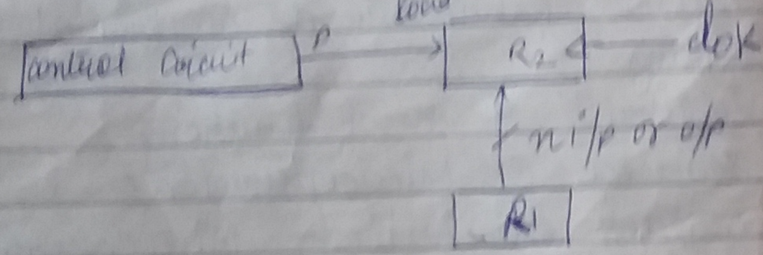
if $(P=1)$ then $(R_2 \leftarrow R_1)$
↳ control signal

Or we can use control function which is true if

i.e, $P: R_2 \leftarrow R_1$

control signal determine which register is to be selected for next

BLOCK DIAGRM



Basic Symbol for Register Transfer

Letters	Denotes a Register	MAR, R2
Parenthesis	" a part of "	R2(0-7), R2(1)
Arrow	Transfer of Information	$R2 \leftarrow R1$
Comma	Seperates two Microoperation	$R2 \leftarrow R1$, $R1 \leftarrow R2$

eg - $T: R2 \leftarrow R1, R1 \leftarrow R2$

It implies at T=1, exchange of information from 2 different bits take place simultaneously

If it is assumed that all transfers occurs during a clock edge Transition.

BUS AND MEMORY TRANSFERS

Generally, k registers → n bits → Produce n common bus lines

BUS SYSTEM basically used to transfer information between various registers

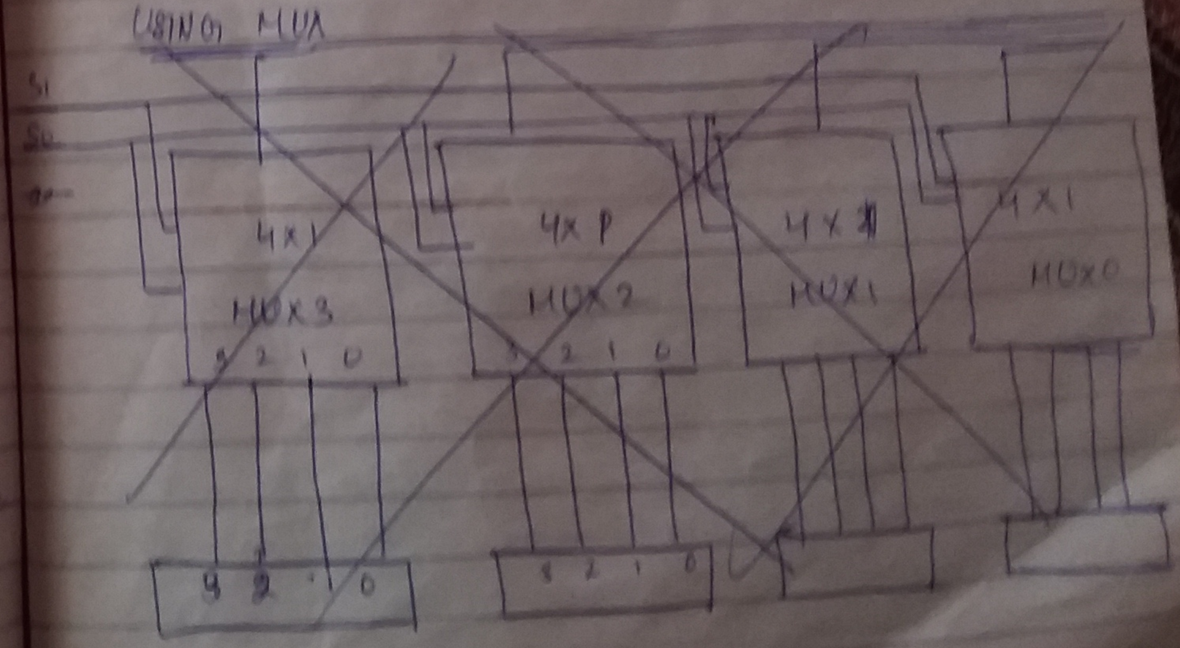
ADVANTAGE

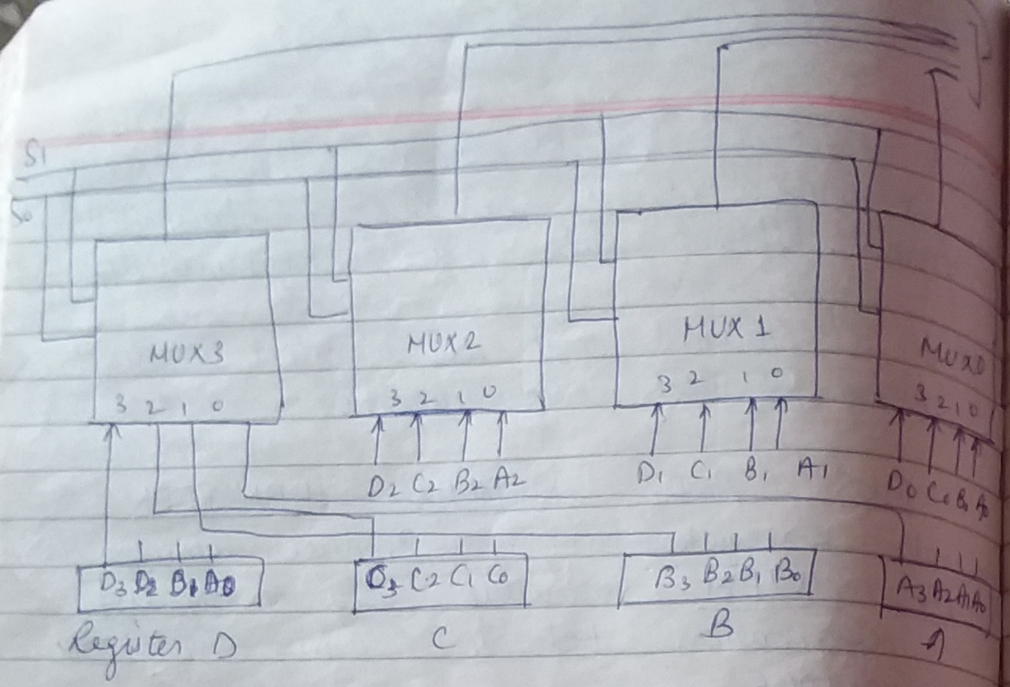
More convinient than using wires for seperate lines b/w registers

Thus, a bus structure consists of a common set of lines, one for each register to transfer information at a time.

CONSTRUCTION

USING MUX





NOTE

If there are K registers of n bits then,

→ the no. of mux needed to construct bus is equal to 'n'.

→ the size of each mux is 'K x 1'

WORKING

0 1 2 3 4 5 4 3 2 1 0

Transfer of info. of from 1 register to other is done with the help of bus.

Firstly, from Register 1 to Bus, then Bus to Register. 2

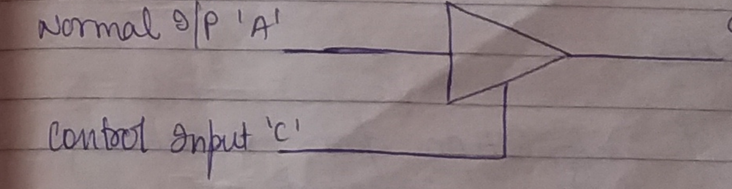
Symbolic

BUS ← C, R1 ← BUS

or, if Bus is known to exist direct method, R1 ← C

(2) BUS SYSTEM USING 3-STATE BUS BUFFER

instead of multiplexer we can use 3-state gates.



Three-state buffer Gate

is a digital circuit in which output depends upon the control signal 'C'.

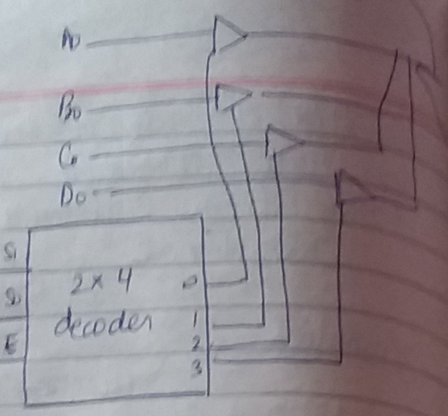
o/p : Y = A ; C = 1

o/p : Y = High impedance state ; C = 0

Boox of high impedence feature availability we can connect, a large no. of 3-State gate to get a common bus line.

we can explain its construction in paper

common bus line for bit 0



No more than 1 buffer gate can be in active state. Since they are connected in a way such that only 1 buffer has access to bus line, others are maintained at high ~~frequency~~ impedance state.

Thus, to make only 1 control point active, we use decoders.
Enable should be 1

This whole circuit is only for 1 bit.
For n bits we need n circuits.

MEMORY TRANSFER

READ : transfer of information from a memory word to outside environment.
WRITE : transfer of new information to be stored.

READ

Command :- Read : DR ← M[AR]
It will take an address from AR [ADDRESS REGISTER] and transfer the info stored at the address in Memory to a DR.

WRITE

command : Write : M[AR] ← DR.
It will transfer the content of Data Register to Memory at some address.

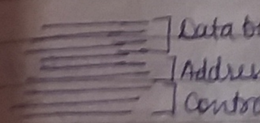
BUS ARCHITECTURE

↳ A communication pathway connecting 2 or more devices.

A Bus that connects major components [Processor, Memory, I/O] is known as System Bus. A System Bus is nothing but a group of buses having seperate func.

System bus usually seprated into 3 functional grps:

⇒ DATA BUS :- It carries data from 1 device to the. It consist of 8, 32, 64, 128 ... etc more buses.

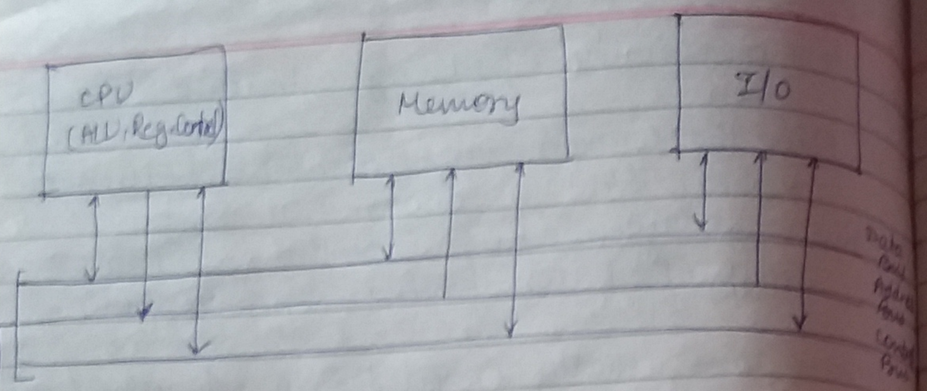


The number of lines/buses referred to as width of Data bus.

⇒ ADDRESS BUS :- It is having collection of wires/buses used to identify particular location in main memory.
It is used to identify the source & destination of data.
Its Bus width ~~determine~~ determine max. memory capacity of system.

⇒ CONTROL BUS :- It regulates activity on the Bus
- It control Timing information.
- It carries signals that report the status of various devices.

Types
Memory Read — I/O Read
" Write — " Write



BUS ARBITRATION

It decides which function to perform first. Since multiple devices are connected, thus it decides which device is to first served

It usually try to balance
(i) Bus priority (ii) fairness

Definition :- To decide which component can transmit over bus, when multiple components want to gain control of the bus.

Methods of Arbitration

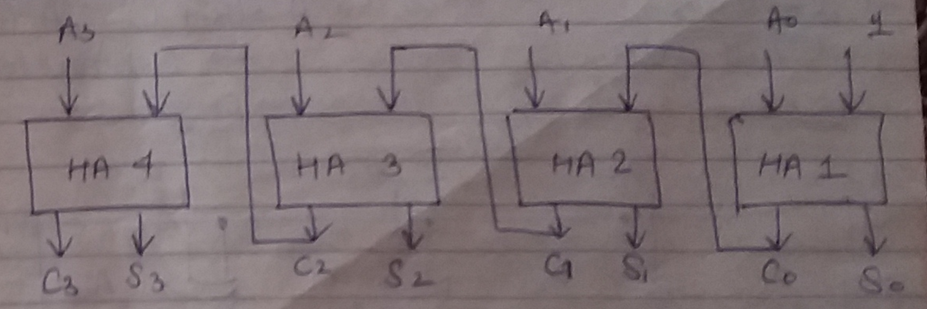
① Centralized	② Distributed
• A single hardware device allocates time and control of bus to other devices. • It may be a seperate module or a part of processor # Bus Controller / Arbitrator	• No central Authority • Each module has a code for access control and modules act together to share bus

Purpose of Arbitration

Both these methods aim to assign one device as Master (I/O Module / Processor)
↳ initiates a read/write operation with some other device [Slave]

~~SHIFT MICROOPERATIONS~~ ARITHMETIC MICROOPERATIONS

BINARY INCREMENTER



Incrementer → Increase value by 1
↙ ↘ can be constructed by
Counter Half Adder

Then explain the circuit as in - $A_3 A_2 A_1 A_0$: 1100 + 1 = 1101

circuit can be extended to n-bit incrementer by including n-half adder

BINARY ~~DECREMENTOR~~ ADDER-SUBTRACTOR

Subtraction → can be done by 1's complement

i.e, $A-B = A+(\overline{B}+1)$

2's complement

Addition & Subtraction can be combined into one circuit by using EX-OR.

Mode input M controls the operation.

M=0 → adder

$B \oplus 0 = B$

↳ A + B + C_0

↳ Full Adder.

$C_0 = 0$

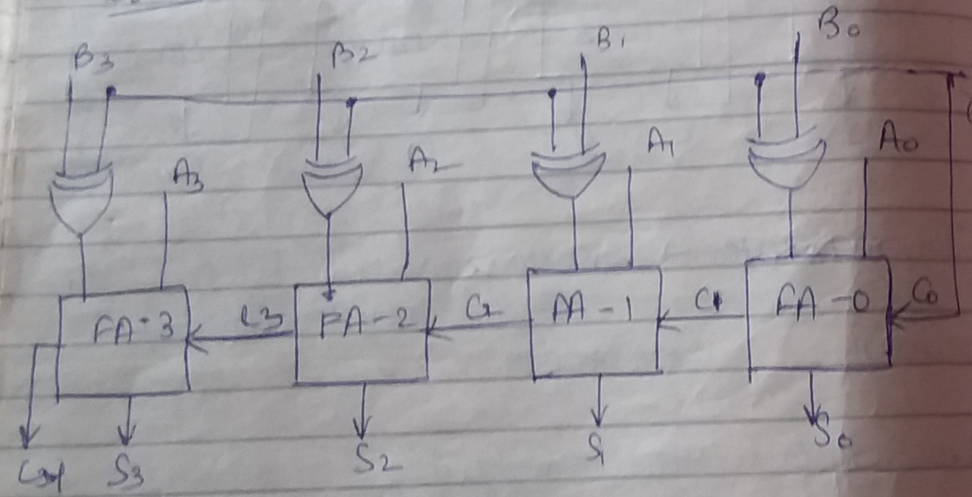
M=1 → Subtractor

$B \oplus 1 = B'$

↳ A + (B' + 1) = A - B

↳ Subtractor

↳ $C_0 = 1$



For Unsigned Number

gives A-B if $A \geq B$

For Signed Number

gives A-B, no overflow

MICROOPERATION

Any operation performed on the content of Register is called μoperation

↓

Arithmetic Operation | Shift Operation | Logical Operation

1. A = A + B (Addition)
2. A = A - B (Subtraction)
3. A = A's (1's complement)
4. A = A's + 1 (2's ")
5. A = A + $\overline{B}$ + 1 (Subtraction using 2's Complement)
6. A = A + 1 (Incrementor)
7. A = A - 1 (Decrementor)

↳ Circuits

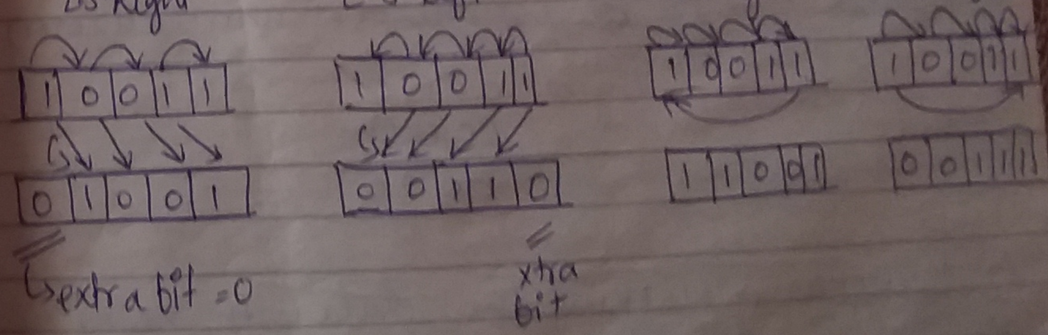
Shift OPERATION

↳ Used to shift the information in a Register

→ Arithmetic Shift

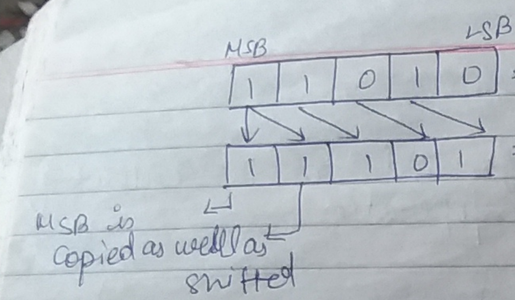
LOGICAL SHIFTS → L.S. Right, L.S. Left

CIRCULAR SHIFTS → C.S. Right, C.S. Left



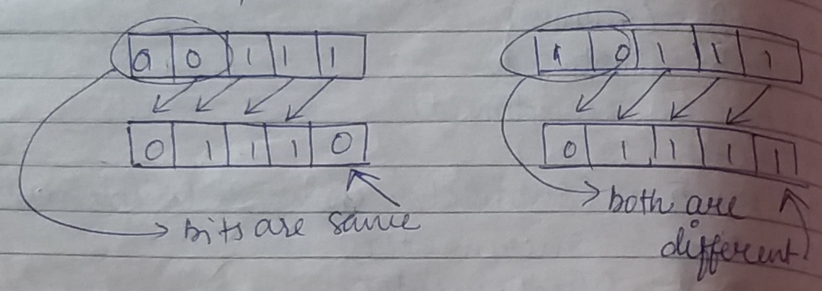
Arithmetic Shift Right

EXOR 1⊕1 = 0 1⊕0 = 1
      0⊕0 = 0 0⊕1 = 1



Basically used for Division

Arithmetic Shift Left



LOGIC MICROOPERATION

Types

① → AND
② → OR
③ → NOT
④ → EX-OR
⑤ → CLEAR → set all bits to zero
⑥ → MASK → Do and operation with all zeros

Eg → 0 1 1 0
     0 0 0 0
     0 0 0 0

⑦ → INSERT → Masking the data with zero's
→ 'OR' the data with Required

Eg → 01010 i/p, we want 01001 o/p

01010 — I/P
00000 — Mask
00000 — OR
01001 — O/P

Paint the world — Painting
Paint Mark — face Painting
Fashionisto — Newspaper dress Making
— Rangoli

⑧ SELECTIVE SET

A = 1 0 1 0 1
B = 0 1 0 1 0
o/p = 1 1 1 1 0

dP = B=1, I
→ B=0, A

⑨ SELECTIVE COMPLEMENT

A = 10101
B = 01001
o/p 11100

B	o/p
1	A
0	A

⑩ SELECTIVE CLEAR

A = 10101
B = 01001
o/p 10100

B	o/p
1	0
0	A

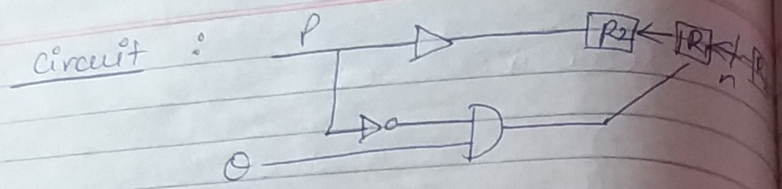
REGISTER TRANSFER LANGUAGE

$P: R_2 \xleftarrow{n} R_1$

// n bit of R_1 as transfer to n bit of R_2

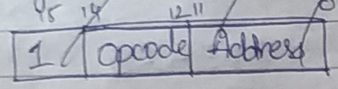
Eg If P=1 then $R_2 \leftarrow R_1$ else Q=1 then $R_1 \leftarrow R_2$

$P: R_2 \leftarrow R_1$
$\bar{P}Q: R_1 \leftarrow R_2$
$P''=0$



Instruction codes are of 3 basic types:-

(1) Memory Reference
takes single memory address as an operand



Instruction Cycle

→ fig no. 11 of 2017

Addressig Mode

(5) Relative Addressing Mode
The effective address is determined by indexed mode using the program counter, in place of general purpose Register.

(6) Auto - Increment Mode

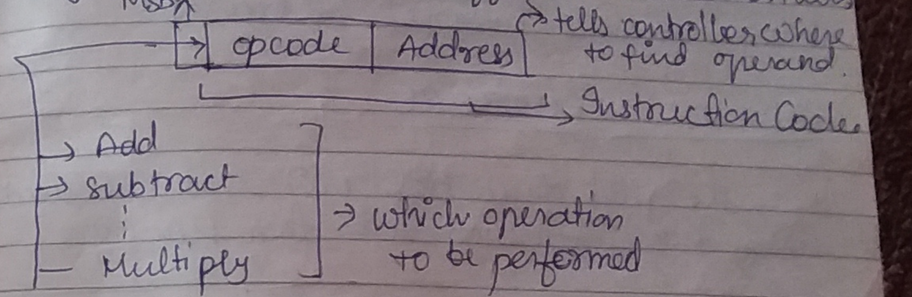
(7) Stack addressing Mode

UNIT - 2

Istruction Codes

↳ of

It is a group of bit that instruct comp to perform the specific operation
Every computer have different Instruction Codes



2^n distinct operation, 'n' bits

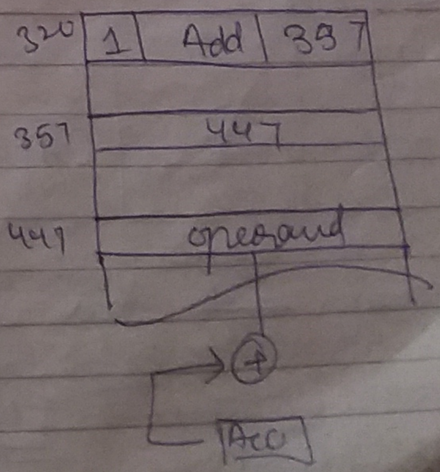
Effective Address - Address where operand is present.

DIRECT ADDRESS

Add R1, (1001)

INDIRECT ADDRESS

Add R1, (1001)
↳ (3287)



(7)

DIRECT AND INDIRECT Address

- When the second part of instruction code specifies the adress of the operand, the instruction is said to be direct address
- vice versa indirect addressable

COMPUTER REGISTERS

It is the part of control unit that are used for storing the instruction code from Memory

REGISTER SYMBOL	No. of BITS	REGISTER NAME	FUNCTION
DR	16	Data Register	Holds Memory operands
AR	12	Address "	Holds Address for Memory
AC	16	Accumulator	Processor Register
IR	16	Instruction Reg	Holds Ins. Code
PC	12	Program Counter	Holds the addres of next Instruction
TR	16	Temporary Register	Holds Temporary data
INPR	8	Input Register	Holds Input character
OUTR	8	Output Register	Holds o/p "

Memory ~~length~~ width max = 4096 word.
$= 2^{16}$ B
= 16 bits

Thus a Register can have max. 16 bit

adding content of 2 registers

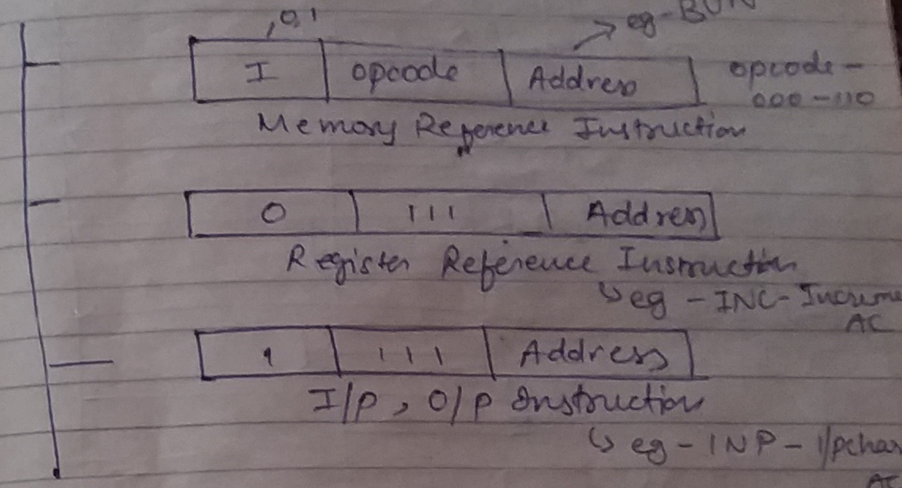
④ Add $R_1, (R_2+R_3)$

⑧ Add $R_1, \#3$ ← Immediate value to add

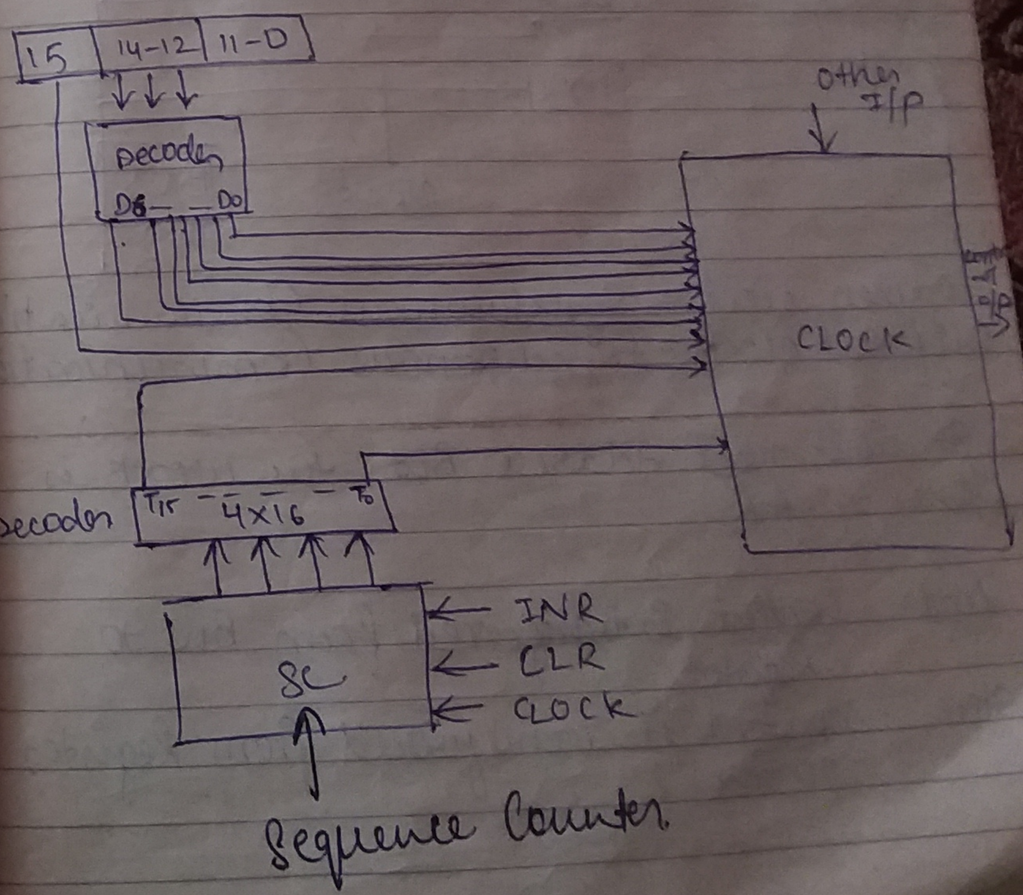
Computer Instructions

Types of Instruction Codes :-

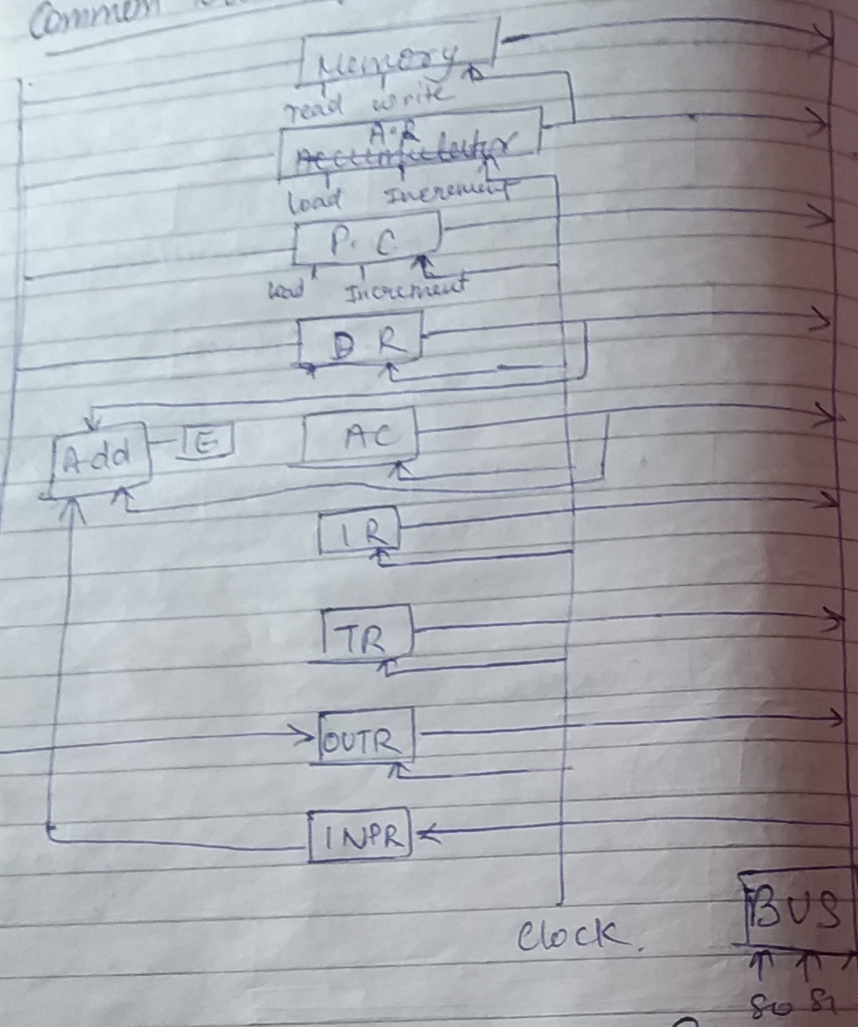
- Mode – 15th (1 bit)
- Opcode – 12-14th (3bit)
- Address – 0-11th (12 bit)



TIMING & CONTROL



Common Bus System [16 Bit]

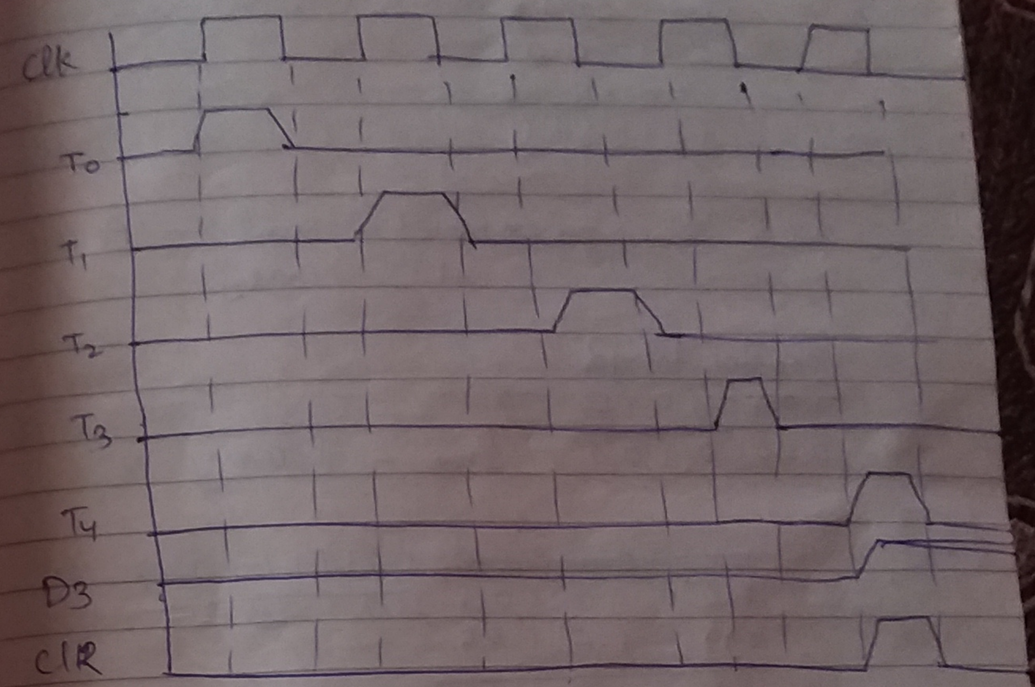


common clock = Synchronous Communication
Different clock = Asynchronous Communication

It don't need Address But, the work is done by AR

Load = content is transferred from Bus to Register.
Store = content is transferred from Register to Bus.

D3 T4 SC ← clr } D3 signal of decoder is enabled at T4 time pulse



CDA

UNIT-I

Instruction Sequencing & Execution/Interpretation

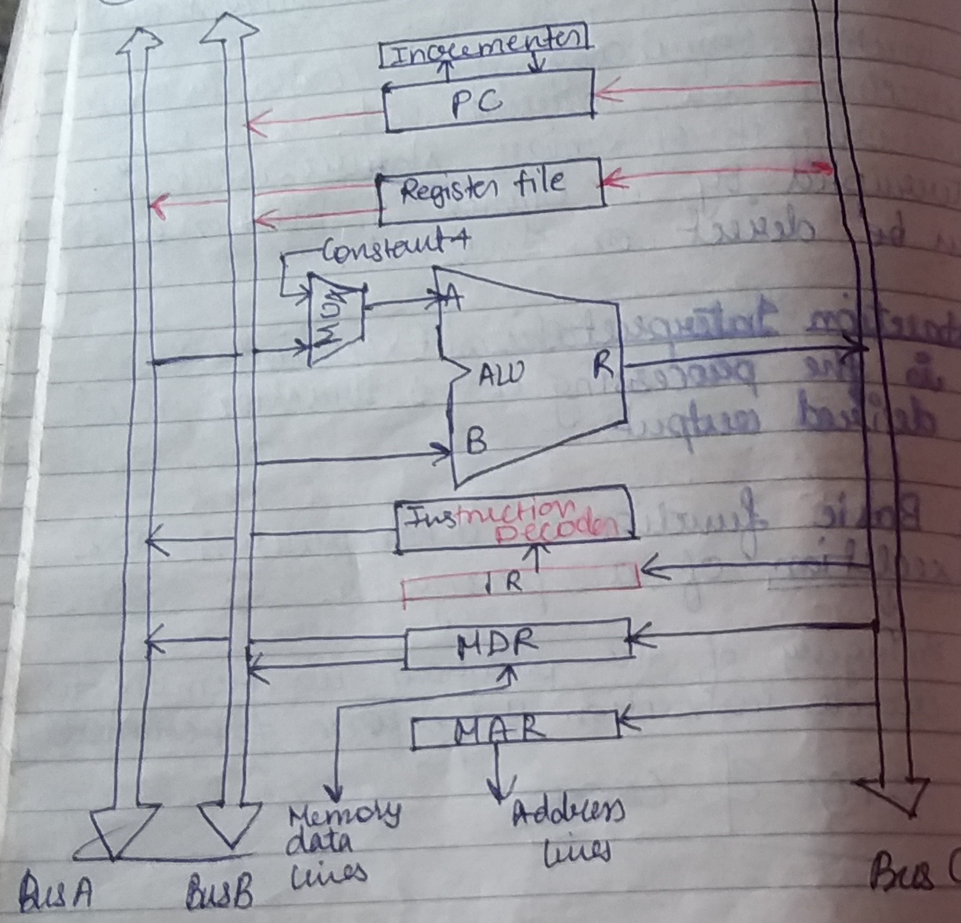
Instruction Sequencing is the order in which instruction are carried out.
Generally, it is of linear nature, which is interrupted by a Branch instruction, which can be direct or indirect.

Instruction Interpretation/Execution -
It is the processing of instruction to get a desired output.

The Basic function of a computer is the execution of a program.
↓
Bringing of the program in Memory to decode instruction & perform functionally

⇒ PDF Copy

(2) MULTIPLE BUS ORGANISATION



CONTROL UNIT ORGANISATION

To execute instructions, processor must have some means of generating the control signal needed in proper sequence.

Therefore, we can generate control signals either by

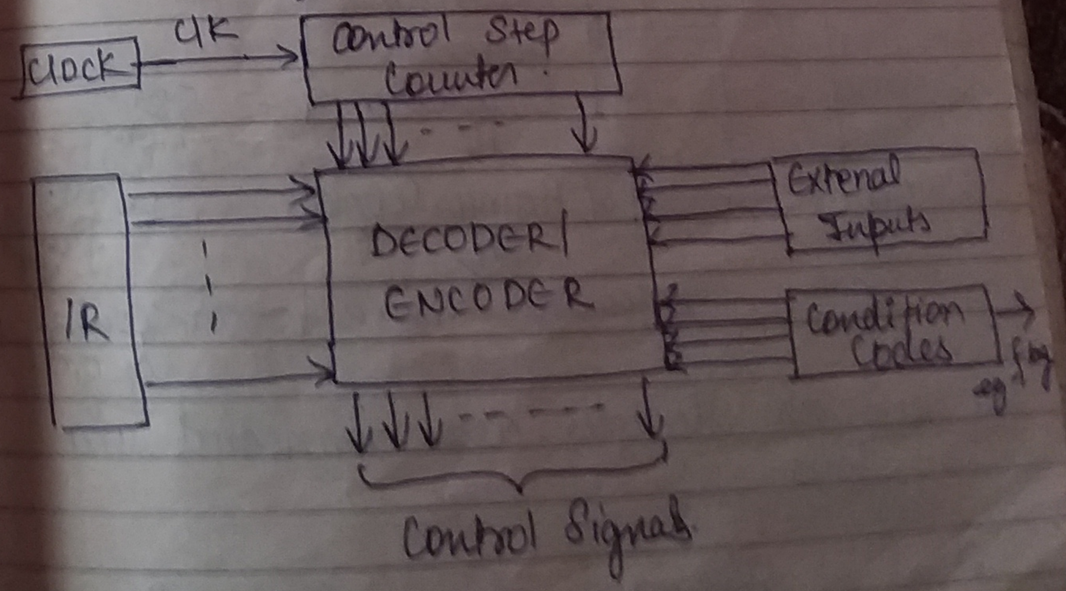
(i) Hardwired Control Unit

or

(ii) Micro-Programmed control Unit

(1) HARDWIRED CONTROL UNIT

A unit that uses combinational logic units, featuring a finite no. of gates that can generate a specific results based on the instruction to invoke responses i-e; control signal.



→ {wired Medium - hardwired {

+ components :-

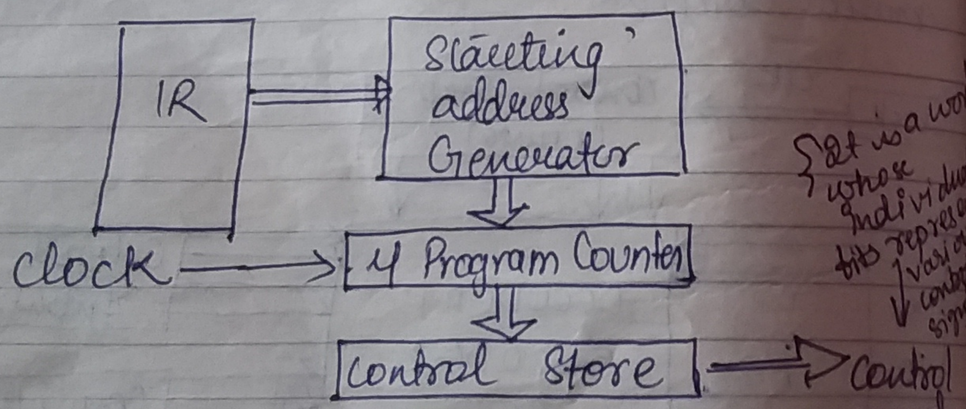
(i) contents of control Step Counter
(ii) Contents of Instruction Register
(iii) Contents of Condition Code.
(iv) Contents of External Input Signal.
↳ eg - MFC, Interrupts etc

2) MICROPROGRAMMED CONTROL UNIT

A unit that contains micro instructions in the control Memory to produce control signal.

→ {Instruction Coding - Software {

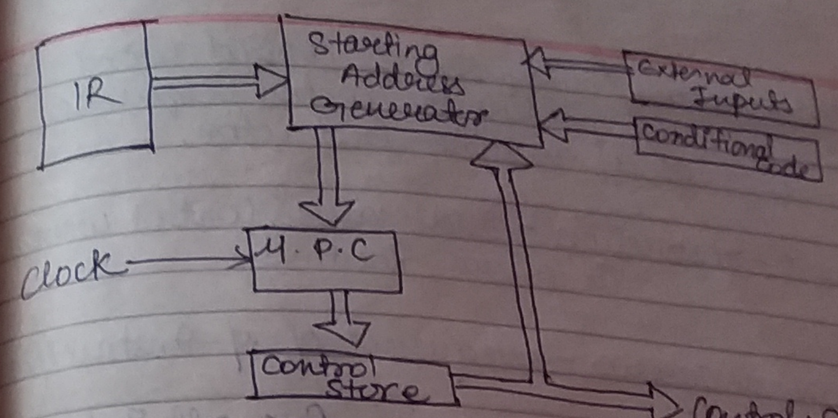
It don't have clock And flags as Hardwired CU.



{ It is a word whose individual bits represent various control signal

In this all Modules are connected via Bus.

But of Program also contains Branch Instruction.



Diff. b/w Hardwired C.U & Microprogram C.U

Hardwired C.U	Microprogrammed C.U
(i) fast operation	(i) slow because it requires memory access
(ii) more costly to implement	(ii) less costly to implement
(iii) To do Modification, entire unit should be redesigned.	(iii) Modification can be done by changing Microinstruction.
(iv) Diff. to perform instruction decoding	(iv) Easy to Perform
(v) No usage	(v) Use control Memory
(vi) Small instruction Set	(vi) Large Instruction Set.

Micro-routine – Sequence of control word corresponding to control sequence of a machine instruction.

Micro-Instruction – Individual control words in micro-Routine.

Micro-program – Sequence of μ-Instructions

Control Store – μ-routine for all instruction in instruction set of a comp-er stored in it.

ADDRESSING MODES

The way the operands are chosen during program execution is specified by addressing mode of the instruction.

> helps in writing program more efficient with no. of instructions and execution time.

The different addressing modes are basically used in Instruction cycle.

(i) Fetch – for every fetch instruct PC is incremented

(ii) Decode – determines operation to be performed, addressing mode, location of operands.

(iii) Execute – gives output and return for other fetch instruction.

INSTRUCTION FORMAT

Mode	Opcode	Address
← used to locate the operands needed for operation	↓ operation code specifies operation to be performed	→ there may or may not be an address field. It designates a memory address or a processor Regr

The instruction may have more than one addressing field, & each addressing field is associated with its addressing mode

Different Types are

(i) **Implied Mode** :- In this mode, operands are specified implicitly in definition of instruction.
eg - " CMP " → Compliment Accumulator
All Register-Reference instruction that use accumilator are of Implied mode ↳ अस्पष्ट
Their operand is implied.

(ii) **Immediate Mode** :- In the operand is specified in instruction itself.
These are useful for initializing Registers to a constant value.
eg - MVI C, 00

(iii) **Register Mode** :- When address field specifies a processor-Register. In this operands are in registers.

(iv) **Register-Indirect Mode** :- In this A reference to register is then equivalent to specifying a Memory addr

(v) Auto-Increment or Auto-Decrement Mode -
Similar to Register Indirect Mode. except ~~after/before~~ register is incremented or decremented after/before it value is used to access memory.
i.e;
Autoincrement - after operand addressing, contents of Register is incremented
Autodecrement - before operand addressing contents of Register is decremented

(vi) DIRECT Addressing Mode - In this effective address is the address field of instruction
eg - Add R1, R2

(vii) Indirect Addressing Mode - In this mode address-field gives the address where effective address is stored.
* Control fetches the instruction from memory and uses its address part to access memory again to read effective address.

eg - Add (R3), R2
↑ indirect

(viii) Relative Address Mode - In the content of P.C is added to address part for effec. add
It is often used with branch-type instruction

(ix) Index Addressing Mode -
effec. add. = address part + index Register

Effective address - It is defined to be the Memory address obtained from computation dictated by addressing Mode.

Index Register - is a special CPU Register that Contains Index value.

(x) Base Register Addressing Modes -
effective address = address field + Base Register
contains Base address

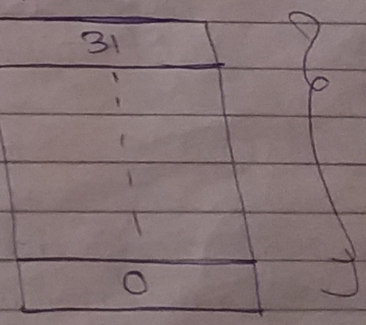
STACK ORGANISATION

→ collection of Registers.
A stack is also used in CPU's as a storage Device. Works on 'LIFO'
Register that holds the address for stack is called Stack Pointer (always points at top)

Operations
Push — Pop

- Stack pointer

SP = Address of Top
DR = Data
Flags → Full
→ Empty



① PUSH

```
if (Full = 0)
SP ← SP+1
M[SP] ← DR
If (SP = 0) then (Full ← 1)
EMPTY ← 0
```

Memory Size = 2

POP

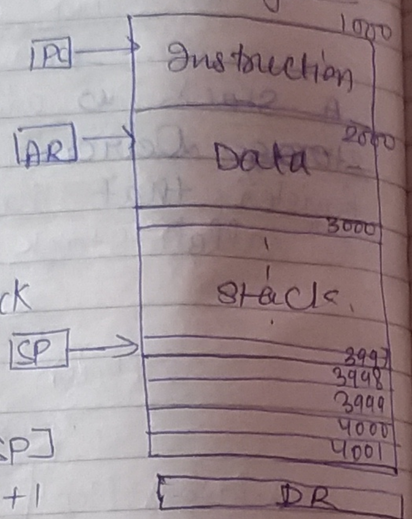
if (EMPTY = 0)
DR ← M[SP]
SP ← SP - 1
if (SP = 0) then (EMPTY ← 1)
FULL ← 0

Memory

A stack can exist as a stand-alone unit or can be implemented in RAM. In CPU, a portion of memory is assigned to stack operation and a processor Register as stack Pointer.

PUSH
SP ← SP - 1
M[SP] ← DR

POP
DR ← M[SP]
SP ← SP + 1



4000 → first entry, 3000 = top

Most comp. don't provide hardware to check for stack overflow / underflow. The stack limits can be checked by 2 processor Registers :-
(i) one to hold upper limit
(ii) " " " lower "
SP compares with both & give result

INSTRUCTION FORMATS

MODE	OPCODE	ADDRESS

Basilly divided in 3 formats :-

(i) General Register Organisation → Three address Instruction
→ Two address Instruction.
(ii) Single Accumulator Organisation → One Address Instruction
(iii) Stack Organisation → Zero address Instruction.

(a) ZERO ADDRESS INSTRUCTION
Use PUSH and POP instructions with an address field to specify the operand.

(b) ONE-ADDRESS INSTRUCTION
uses an Accumulator along with one address field. All the results are stored in AC.

(c) TWO ADDRESS INSTRUCTION
most common in commercial computers. Use 2 address field for doing operation as well as storage.
ADD R1, B } R1 ← R1 + M[B]

(d) THREE ADDRESS INSTRUCTION
Use 3 address field, 2 for doing operation & 1 as their storing location.
ADD R1, A, B } R1 ← M[A] + M[B]

UNIT-IV

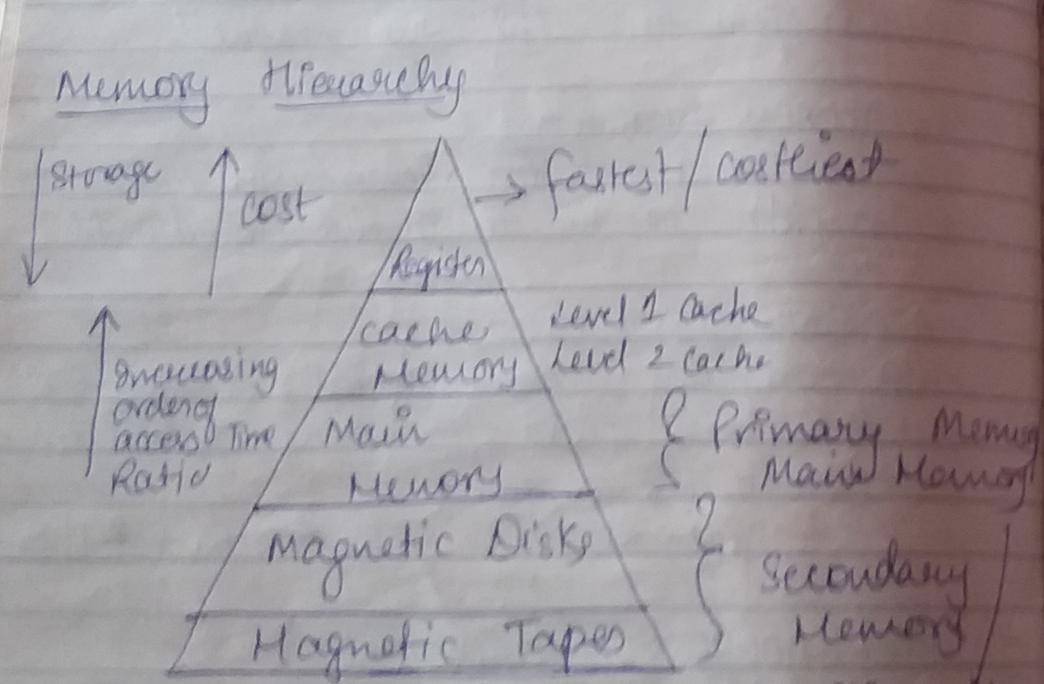
MEMORY

A Memory unit is the collection of storage units or devices together. Store bits

Volatile
loses data, when power is off.
Temporary storage
Also known as Primary Memory [RAM]
- SRAM
- DRAM

Non-Volatile
permanent storage
does not lose any data.
Also known as Secondary [ROM] Memory
- PROM
- EPROM
- EEPROM

Memory Hierarchy



Main Memory is in b/w bcoz it directly communicates with CPU & auxiliary

Cache Memory store program which is currently being executed.

MAIN MEMORY
directly communicates with CPU, Cache & Auxiliary Memory. Large and fast

① RAM – Random Access Memory
Read / Write

DRAM	SRAM
ⓐ Dynamic RAM	ⓐ Static RAM
ⓑ constructed of Capacitor	ⓑ Constructed of D-flipflop
ⓒ Requires recharge after every few milli seconds	ⓒ Holds content as long as power is available
ⓓ Used for main Memory	ⓓ used for Cache

② ROM – Read only Memory
Stores critical info as program to boot the comp
used in calculators / peripheral devices

PROM	EPROM	EEPROM
Programmable Read-Only Memory	Erasable PROM	Electrically EPROM
Once Programmed, can't be changed.	Changed by exposing to UV Rays.	changed by electrical field

RAM/ROM - Diff - Akaash.

Auxiliary - Not directly accessed by CPU, accessed using Input/Output channels.

CACHE MEMORY

Locality of Reference - Tendency of processor to access the same set of memory location repetitively over a short period of time.

Cache works on Principle of Locality. Thus active portion of programs are placed here for reducing access time. Cache is placed b/w CPU & main Memory.

WORKING

When CPU needs to access memory, cache is examined. If word is found in cache, it is returned to CPU. If not, then main memory is searched. A Block of words, containing that is then transferred from main Memory to Cache Memory for future References.

If word is found in cache, its cache Hit
If word is not found in cache, its Cache Miss.

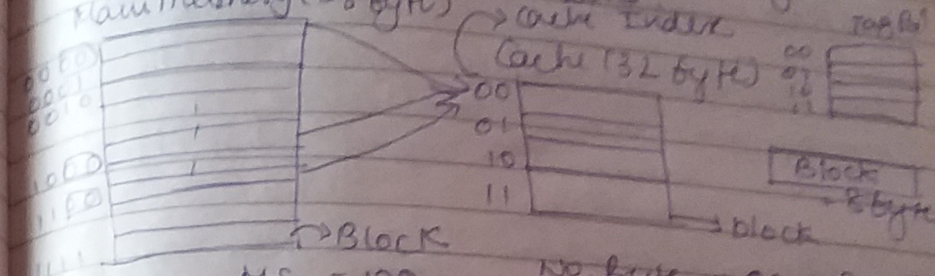
$$\text{Hit Ratio} = \frac{\text{no. of Hits}}{\text{no. of attempts}} \approx 0.8$$

Cache Mapping :- Transformation of data from Main Memory to Cache Memory. Types:-

(i) Direct Mapping
(ii) Associative Mapping
(iii) Set-Associative Mapping

Many to one function

(i) DIRECT MAPPING - Each memory block maps with exactly one cache line



No. Byte = $\frac{MS}{BS} = \frac{128}{8}$ = 16 = 2^4 blocks = 4 bits

No. Byte = $\frac{CS}{BS} = \frac{32}{8} = 4 = 2^2$ block

Block Index

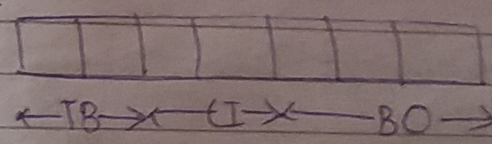
It does not mean all block with LSB 00 stored in Cache Index 00, any one of it.

CPU searches, Thus for clearity it holds Tag Bits

Tag Size = B1 - C1 = 4 - 2 = 2

Memory Configuration

128 byte = 2^7
PA = 7

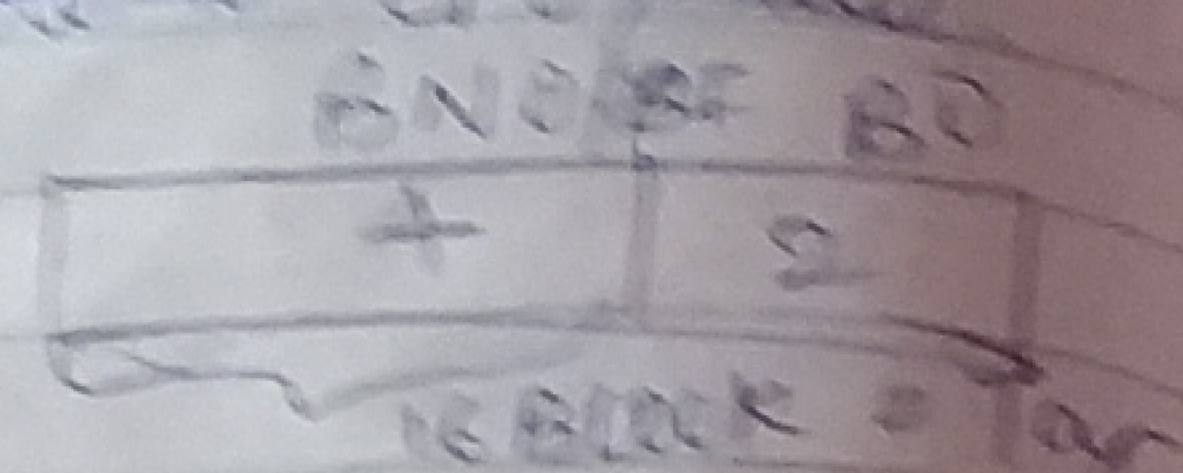


∴ In PA (Physical Address) firstly CI is matched with Cache, then there corresponding Tag bits are matched, & its Catch hit.

For any case if any inf. is not match it is searched in Memory., & copied at the CI.

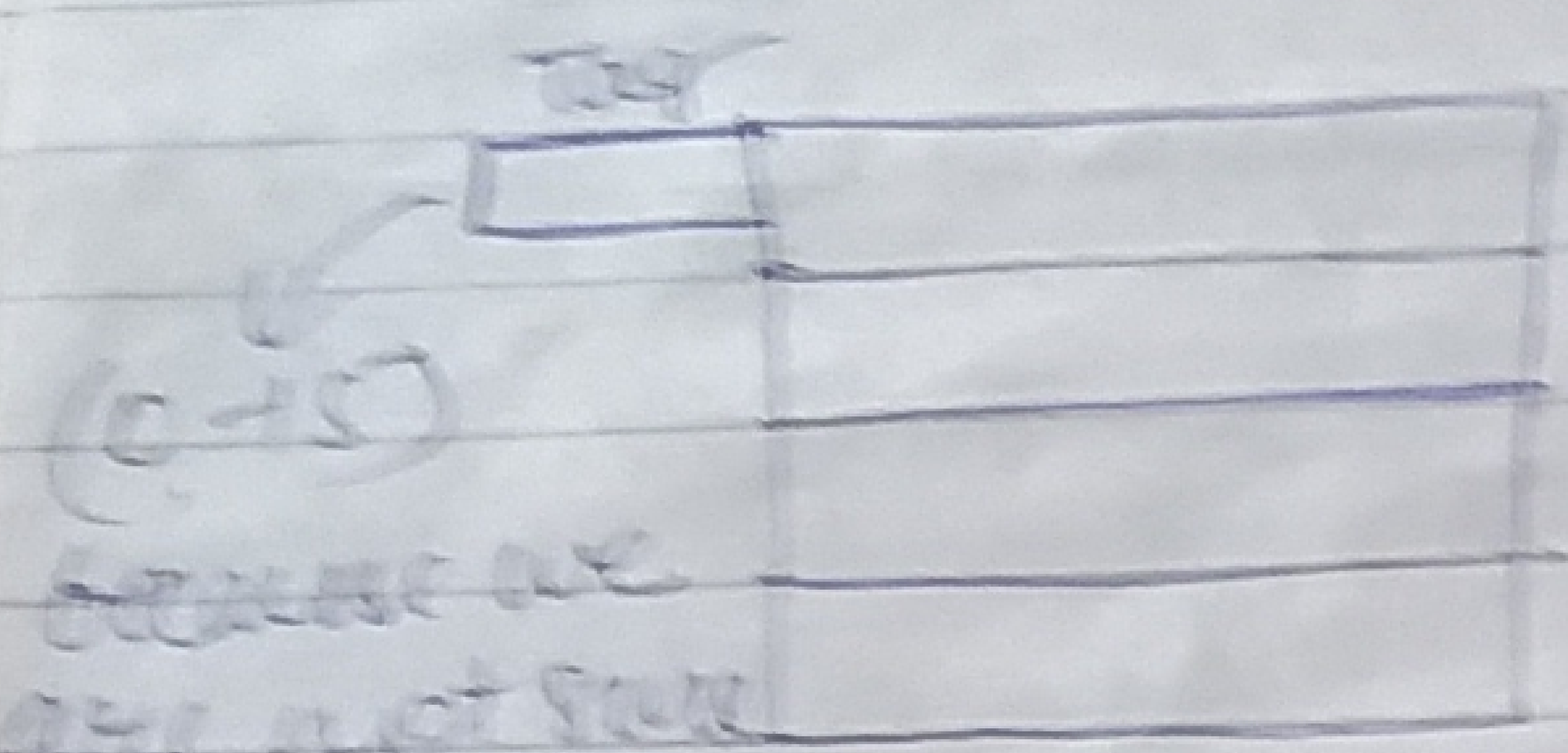
— file

~~[illegible] directly accessible to CPU~~



Associative Mapping

Any block can occupy any cache line
~~It is one to one function~~
It is many to many function



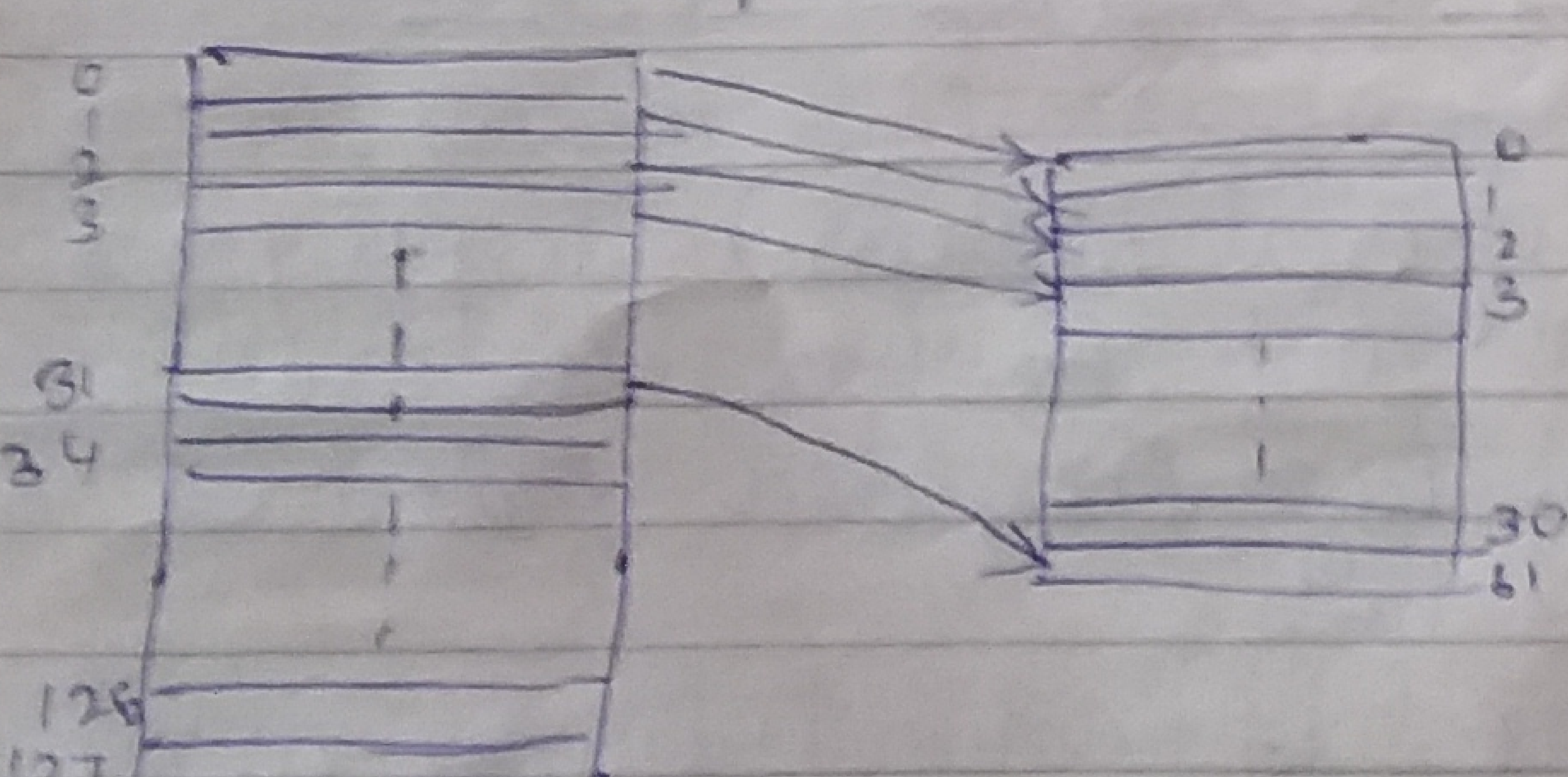
← Carry of 0-15 blocks

(0-15) because we are not sure which Tags are there

→ searching gets difficult
→ Access time increases
Thus for using it we use n-comparator which compare B.No with Tag simultaneously.

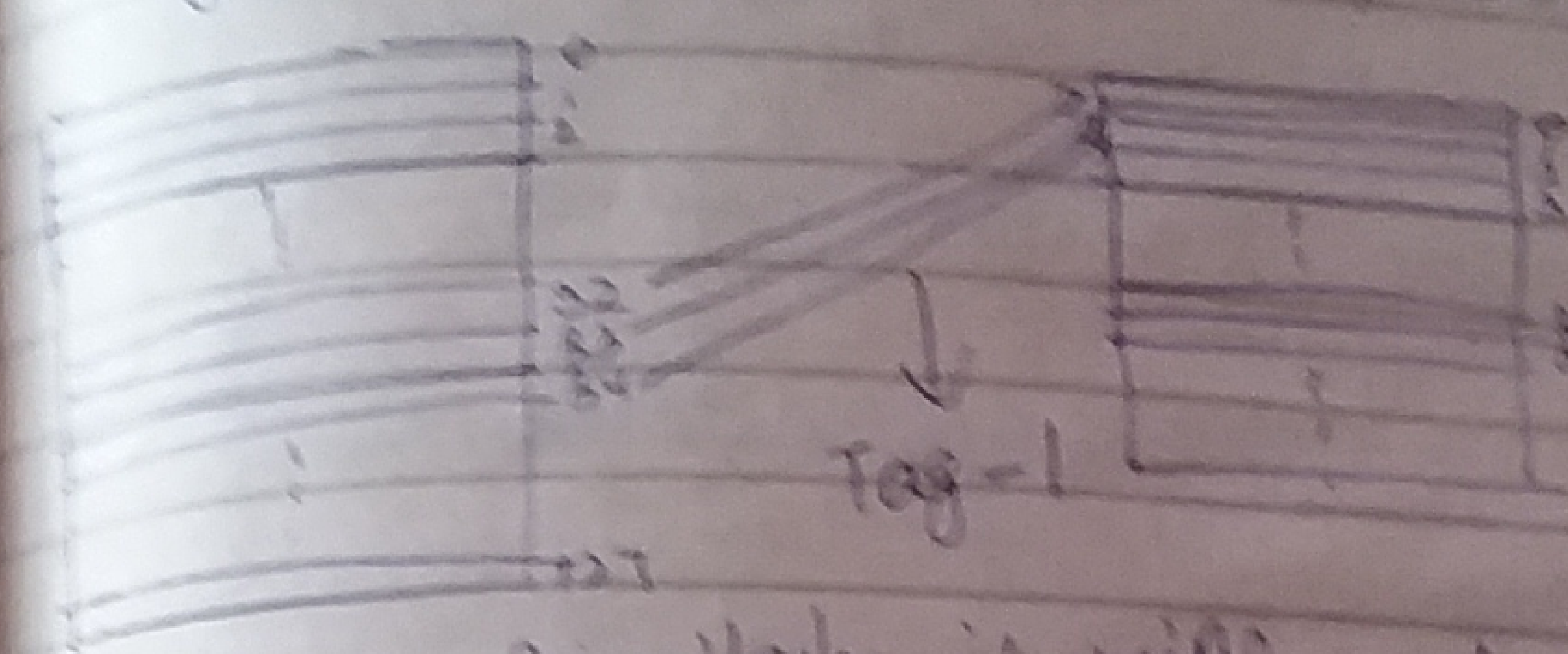
SET-ASSOCIATIVE MAPPING

Modulo concepts come.



all corresponding are maped till 31
They all are stored a

Tag 0. Then the next 32nd block goes to 32 % 32 = 0, 33 % 32 = 1.



Thus similarly it will contain 4 tag of 32 blocks each.

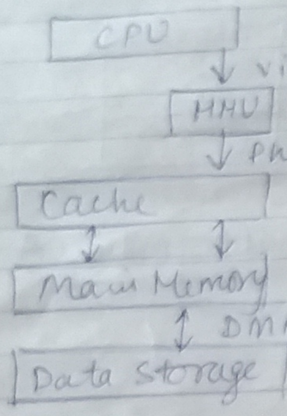
VIRTUAL MEMORY

It is the Seperation of Logical Memory from Physical Memory.

It is used in case where Main Memory is of small size.
Our program doesn't get fit in Main Memory it needs more space.
Thus the part of program which is not currently executed is stored in secondary memory.
Thus, once first part is executed, then the other part is executed by replacing first program by other part.

It is basically a technique the automatically swaps program data block b/w Main & Secondary Memory are called Virtual Memory.

The address that CPU issues to access is Virtual Data Address.



DATA TRANSFER
(from Sender to Receiver)

Synchronous Data Transfer
All connected device derive the inf. using common clock

Asynchronous Data Transfer
No common clock

Asynchronous Data Transfer

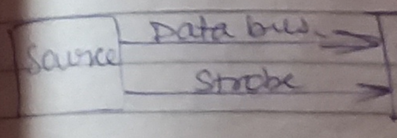
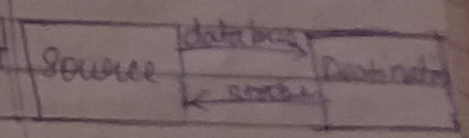
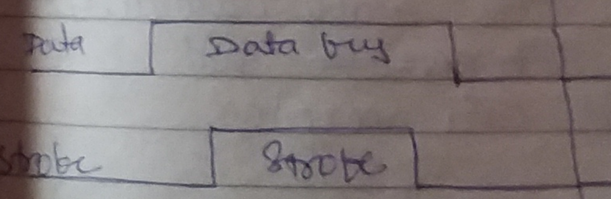
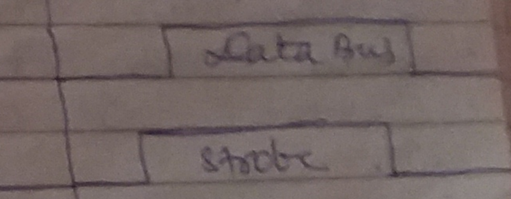
It transfers the data between 2 independent units. requires that control signal be transmitted to indicate the time at which data is being transmitted.

Strobe Pulse Method	Handshaking Method

...l is supplied by one ...it to indicate other ... when transfer has ...ces	control signal is accompanied with each data being transmitted to indicate transfer of data.
No receiving signal	Receiving signal corresponds with other to indicate receipt of data

STROBE PULSE

The strobe may be activated by either Source or destination
Strobe = Signal
can be source Initiated or Destination Initiated.

Source Initiated	Destination Initiated
	
Str. Source sends signal that im sending data & place data on data bus	Destination sends signal that m ready to receive & Source place on data bus
	

Disadvantages

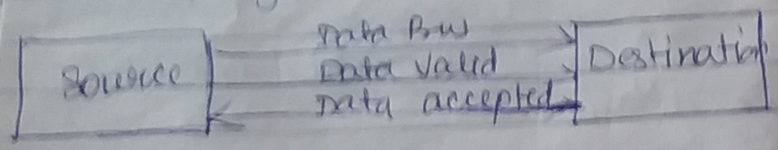
The source unit doesn't get any information regarding completion of transfer	The destination unit doesn't get any info. whether the source has actually placed data or not.

Thus use of HANDSHAKE METHOD which also contains a Reply Signal

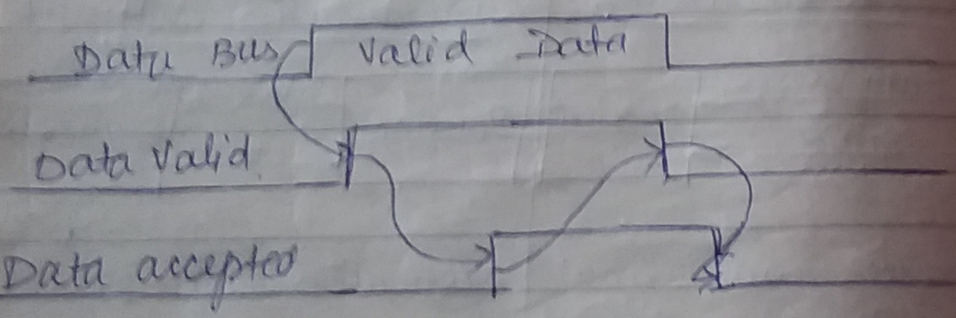
HANDSHAKE METHOD

Source Initiated

Block Diagram



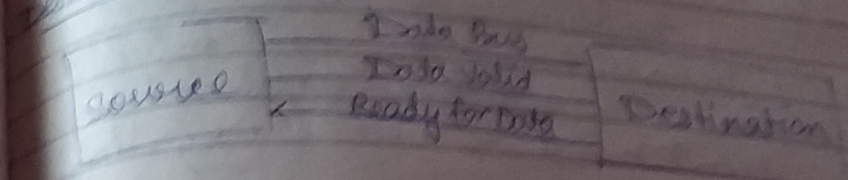
Timing Diagram

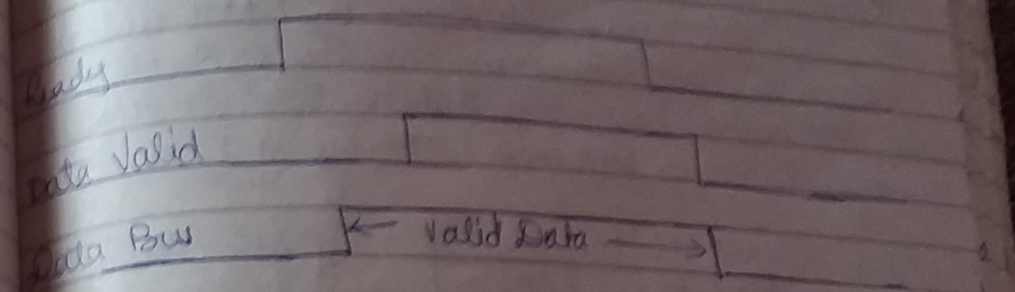


- Source place data on Bus, Enable data valid
- Destination accepts data, enable data accepted
- Source disable data ~~accepted~~ valid
- Destination disable data accepted

Rate of transfer determined by slower unit

Destination Initiated





- Destination Requests and says Ready
- Source Place Data, Data Valid Enable
- Destination Accepts data, disable Ready Signal
- Source disable data valid

Disadvantage

- Needs active participation from both. Thus flexible ~~but~~ reliable
- One unit is faulty. Thus Incomplete

SERIAL COMMUNICATION

It transfers in binary pulse by bit using a two wires that is sender and receiver.

It is a technique used to send data bit

e.g. - we want to send an 8 bit binary data (11001110)

Thus we can send it either by Little Endian [LSB at lowest memory address & MSB at highest] or Big Endian [LSB at highest Memory address & ~~LSB~~ MSB at lowest].

i-e,

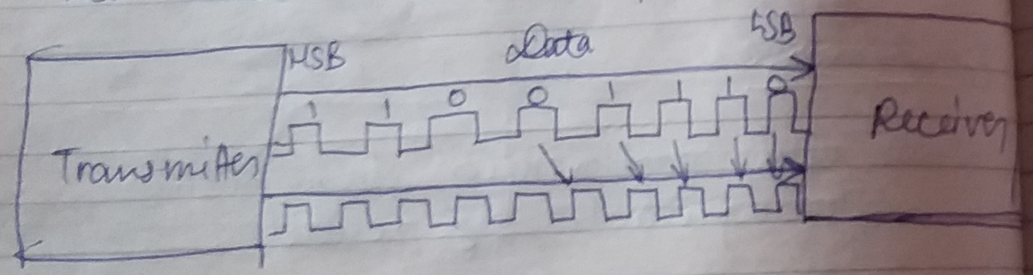
Add	100	101	102	103
Value	10	11	00	11

} Little Endian ——— (1)

Add	100	101	102	103
Value	11	00	11	10

} Big Endian

Let choose Little Endian for this (2)



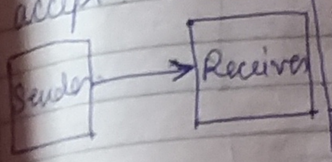
For every clock pulse, one bit is transfer

widely used approach to transfer inf. b/w data processing equipment & peripherals.

Serial Transmission Modes

Simplex

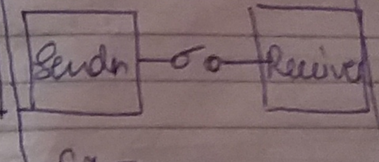
- one-way communication
- only one either sender or receiver active at a time
- Sender transmits, Receiver can only accept



Ex- Radio & TV. transmission

Half Duplex

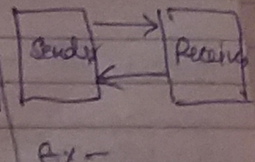
- also one-way
- both are active but not at a time
- Sender transmit, Reciever accept can't send



Ex- Internet,

Full Duplex

- 2 way
- both are active at same time
- Both can transmit & receive at same time



Ex- Smartphones

DMA [DIRECT MEMORY ACCESS]

Removing the CPU., direct access of Memory by peripheral devices are known as DMA

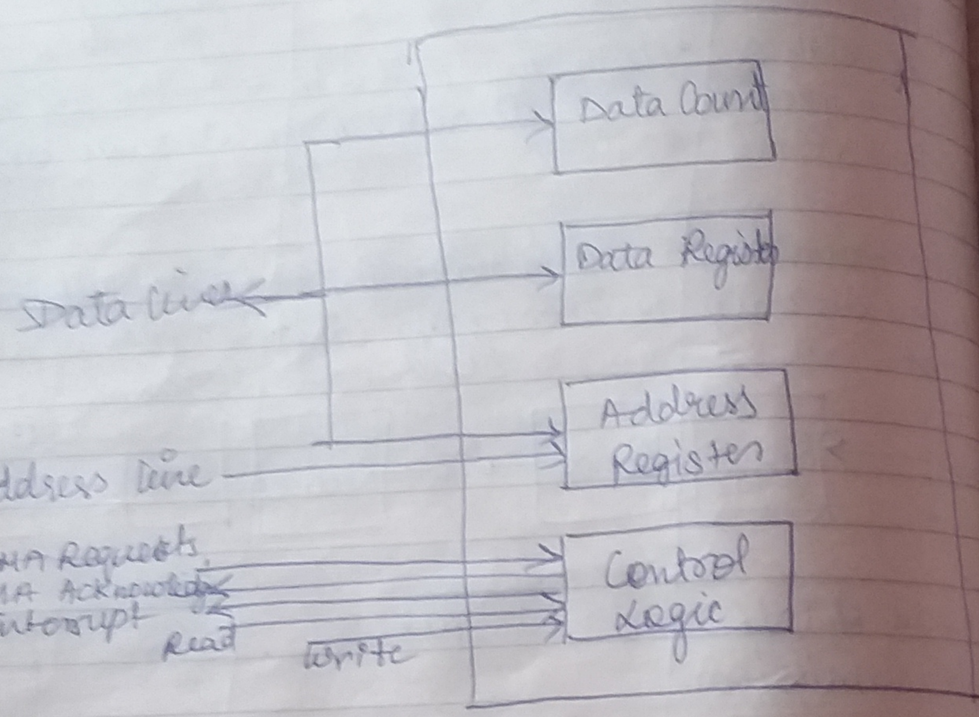
DMA controller manages to transfer data b/w peripheral & memory units.

&

interface transfer data to & from Memory through Memory Bus.

Ex - Disk Drive controller, graphic cards, sound cards, I/O Multifunc processor

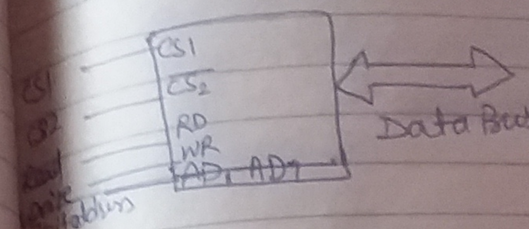
CPU can only initiate the transfer, receive interrupts, etc



Block Diagram of DMA

RAM CHIPS	ROM CHIPS
• have bidirectional data bus to perform data operations like read, write.	• can't put anything on Data Bus, unidirectional Data Bus
• have 2 chip select which enable the	• ROM have more bits on a chip than RAM

• chip select when called by Processor
• capacity of memory is 128 words (8 bit each)
• have 7 bit-address bus & 8 bit data bus



• 8 bit address bus, 8-bit data bus