J. Bharath
17001-CM-028

Syllabus

1) functional blocks of CPU
2) Data Representation
   * fixed point and floating point number representation
3) Instructions
4) Addressing modes ✓
5) Stored Program Concept ✓
6) Instruction execution
7) memory Hierarchy
8) virtual memory ✓
9) Associative memory
10) cache memory
11) I/O Organization
12) methods of data transfer
    * programmed I/O
    * interrupt initiated I/O
    * DMA Transfer
13) ——→ IOP (Input Output Processing)
14) Pipeline and vector processing
15) Flynns classification

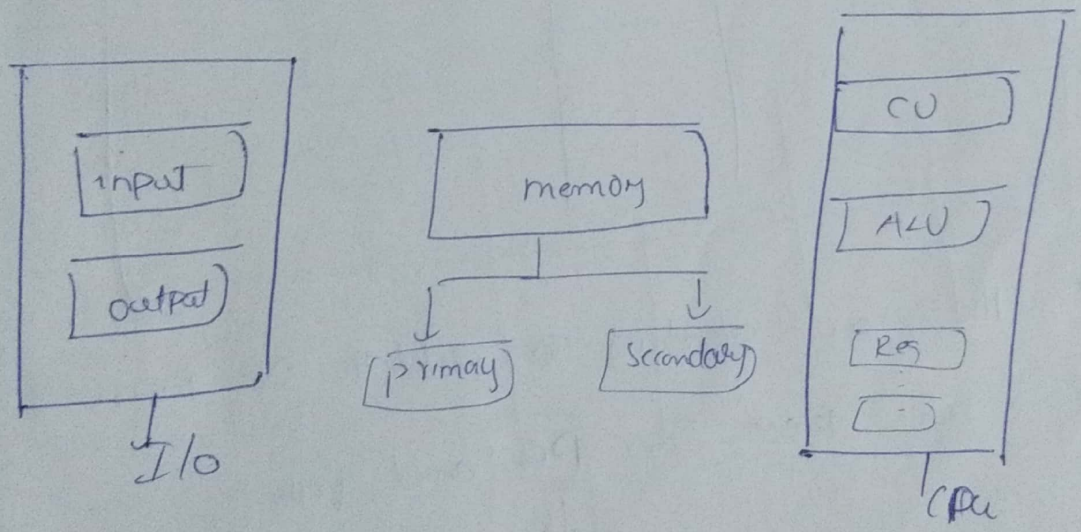
Computer organization:

It is concerned with study of hardware components and how these hardware components were connected together to make a computer system

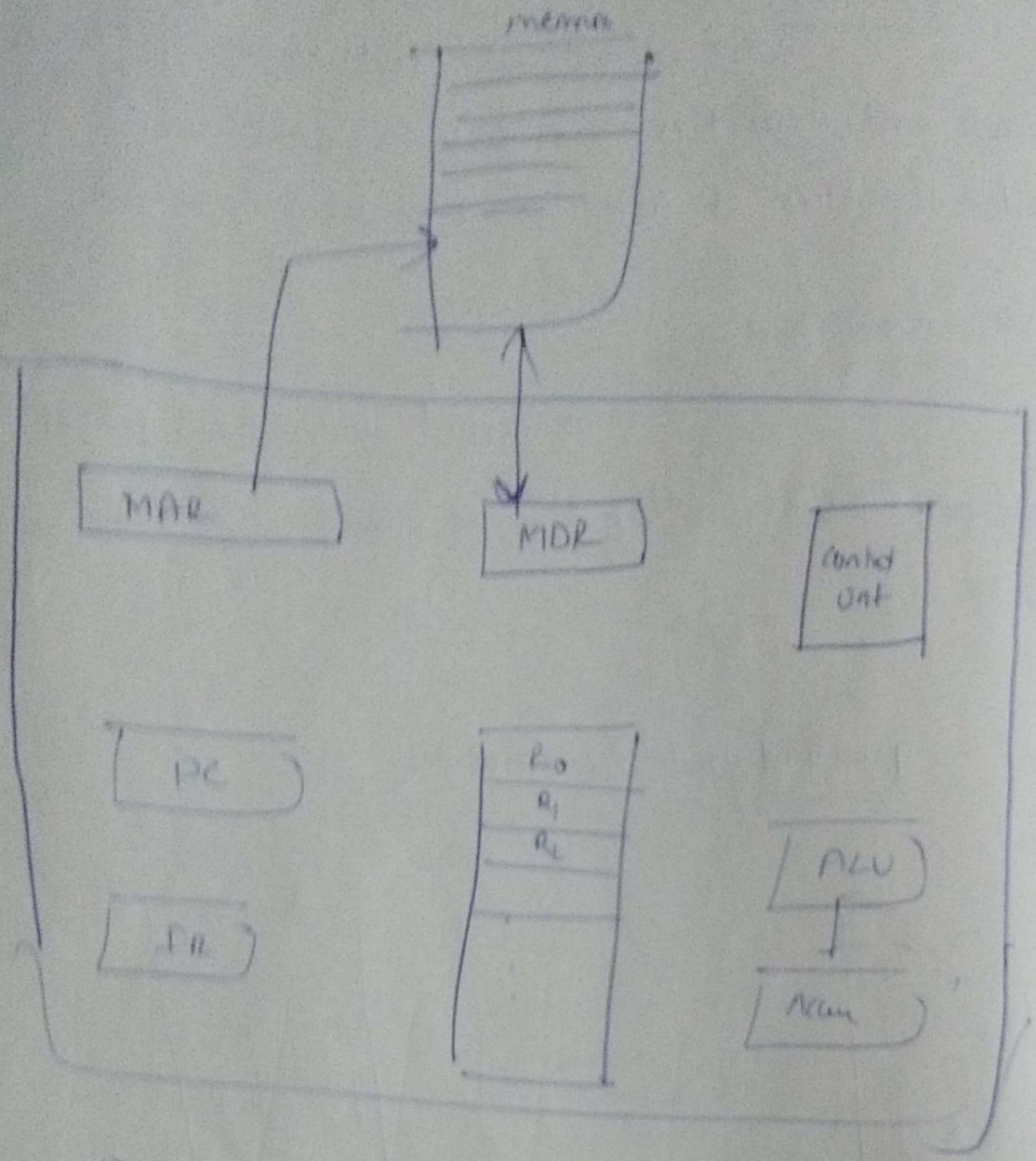
Computer architecture:

It is concerned with the study of structural and behavioural of a computer as ~~been~~ seen by the user

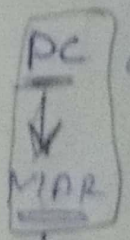
functional units of a computer



Basic operational concepts



→ The starting address of program will kept in PC and from



points to location

↓

Now control unit decides mode read (b) memory read (ii) memory write

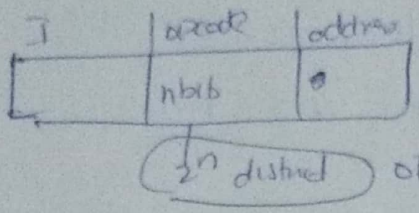
NOTE

PC Navigates the program

Now data will be moved to MDR → IR

In IR The Realisation of instruction will be done

i.e what is opcode, address and what mode



→ Next the instruction will be moved to ALU and calculation will be done and again unit back to memory

Register

Specific Register

1. MAR
2. PC
3. MDR
4. MBR: memory Buffer Register
5. IR
6. Accumulator
7. Flag Register

General purpose registers

1. $R_0 \cdots R_{n-1}$

MAR :

→ It holds the address of current instruction which is to be executed

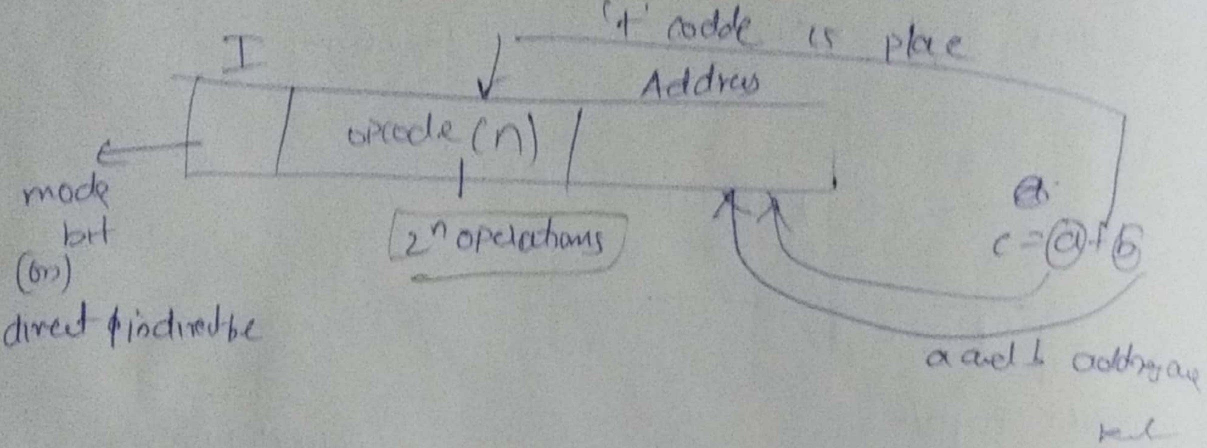
P.C :

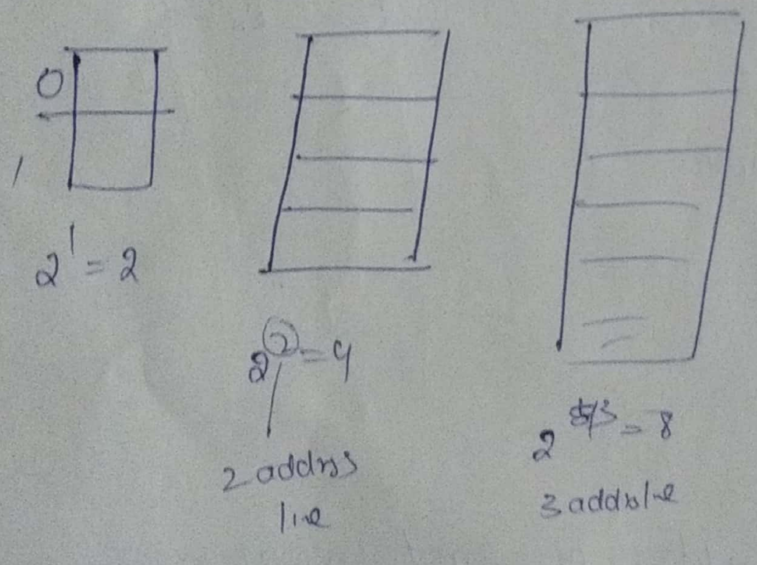
→ It generates the address of next instruction to be get executed.

Ex1: If one instruction is 3 byte and store at 5000h and continue to store to 5003h. The If the current value of PC is 5000h then the ~~address of PC~~ value of PC next cycle is 5003h not 5001h

MDR: It holds the content of memory location which is pointed by MAR

IR: The instruction Realises its formed in IR





$2^{10} =$ 1024 1 kilo

$2^{20} =$ 1024k = 1 Mega

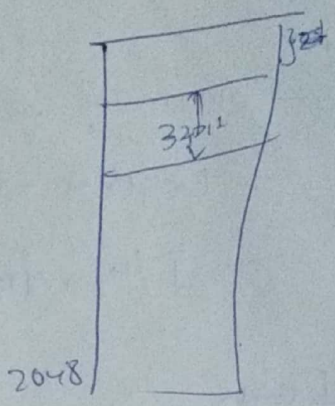
$2^{30} =$ 1024 m = 1 Giga

$2^{40} =$ 1024 G = 1 Tera

Ecet

MM size is 2048 x 32

what is the size of PC and IR



a) $2048 = 2^{11} \times 32$

address = 11 bits

so PC size = 11 bits

and given data/key (or) data size = 32 bits

so the size of MDR = 32

IR = 32 [because data will be loaded in to IR by MDR

Reg = 32 [i. . . . . IR to Reg]

Accumulator = 32 [. . . . . Reg to AC]

Program execution

1. Fetch → fetch Route

T0: MAR ← PC

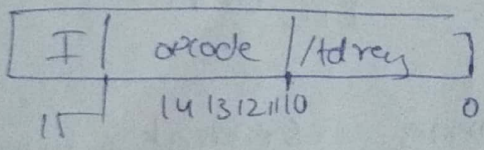
T1: MDR ← m[MAR], PC ← PC + 1

T2: IR ← MDR

2. Decode

IR:

T3:



Execute

Tu- memory -reference [for knowing address of variable
- Register- reference [for getting value of operands]
- I/O reference [Any scanf() used]

Regs

Microoperation and Register ~~control~~ Transfer language (RTL)

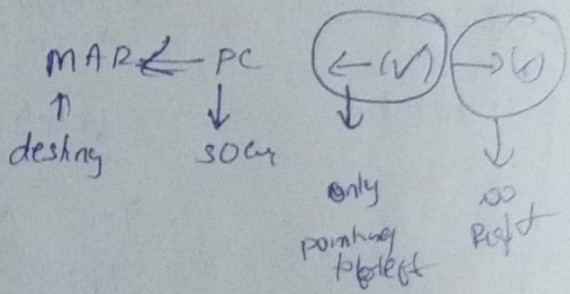
* Moperation:

→ The operations which are formed on the data stored with in register called microoperations

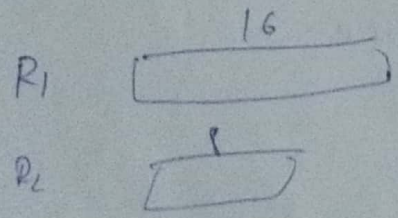
* RTL G

Notations

1. capital letters eg R1, R2, R3 MAR, PC
2. Arrow mark (←)



3. Parenthesis ( )



R2 ← R1(0-7)

4. comma, separate two microoperations in one clock cycle

eg MDR ← M[MAR], PC ← PC+1

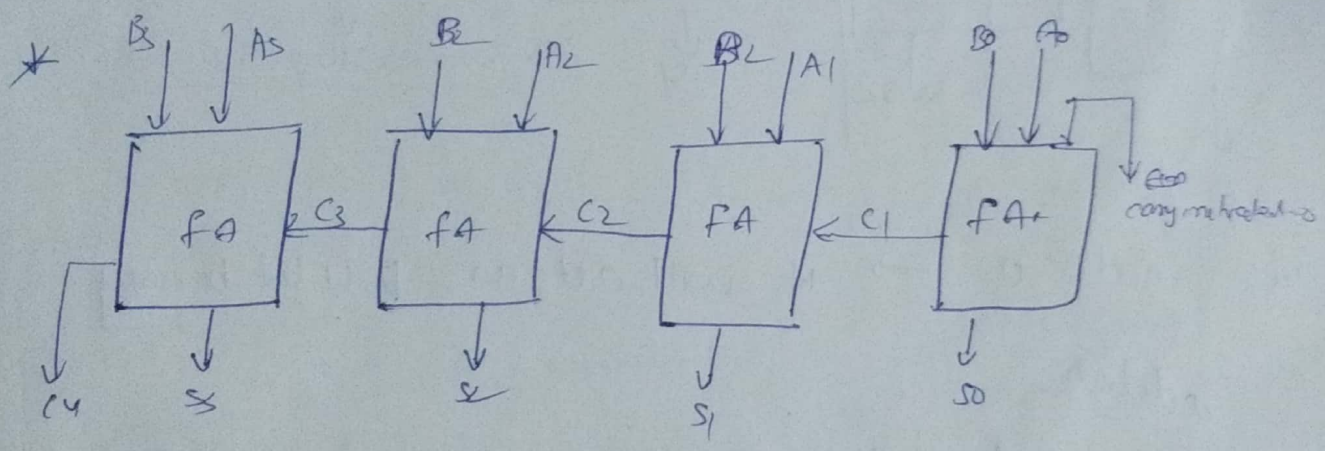
Microoperations (μ)

1. Arithmatic microoperations
2. Logical microoperations
3. shift microoperations

Arithmatic microOperations :

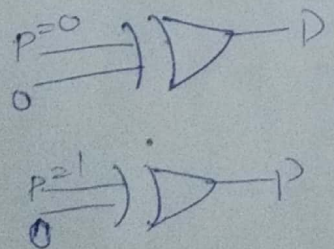
* Basic : 1) Add
2) SUB
3) INC
4) DEC

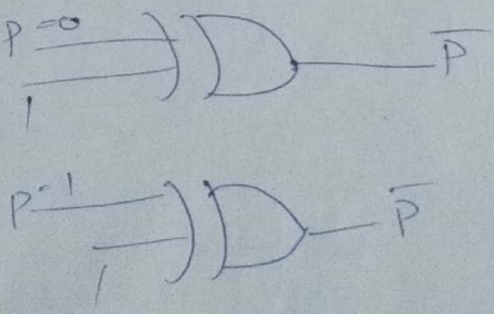
4- bit Binary adder



4-bit Binary subtractor:

A - B = A + B + 1

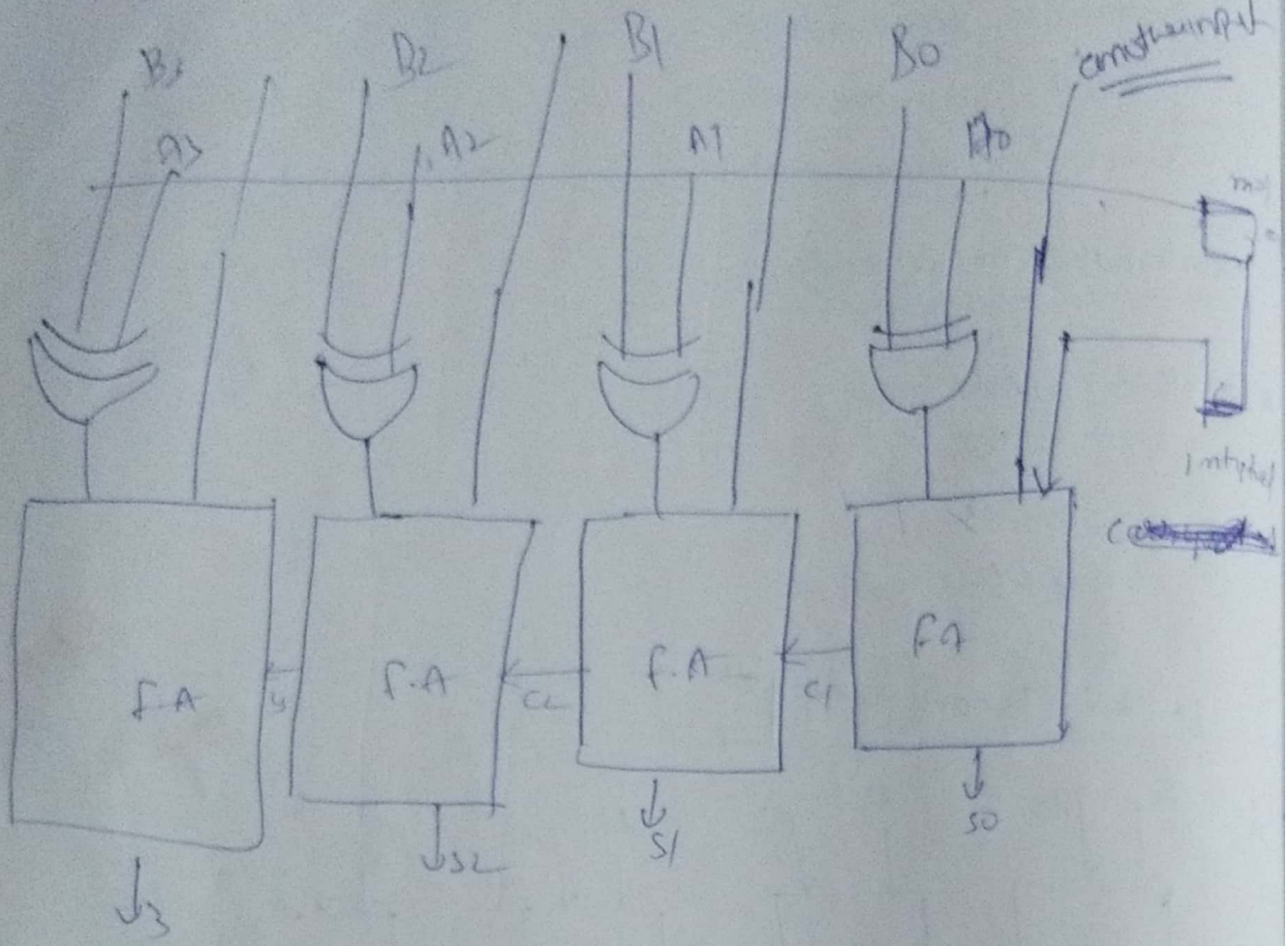




when standard I/P=0 The output is same P

When standard I/P=1 The output complmts

4 bit binary adder - subtractor



if mod = 0 → it will act as 4 bit binary adder

if mod = 1 it will act as 4 bit binary subtractor

4 bit binary subtractor

A = A

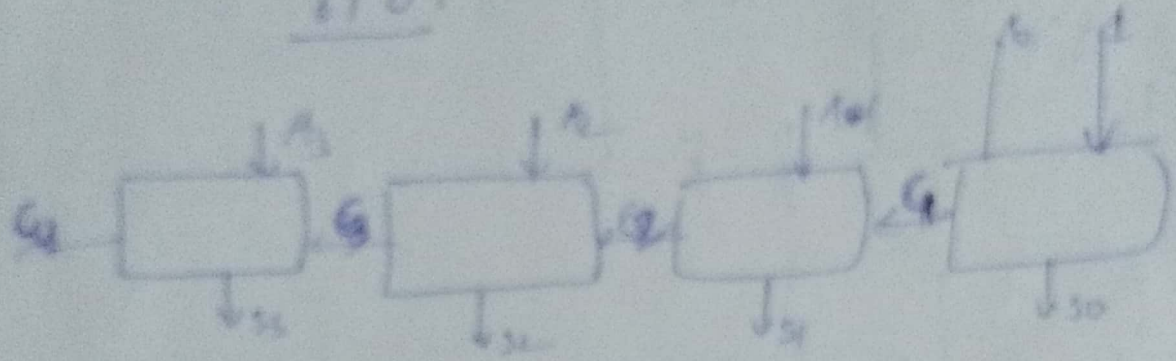
A = A+1

0100

1

0101

Two input so we have to use half adder



4 bit binary decrement

A = A

A = A-1;

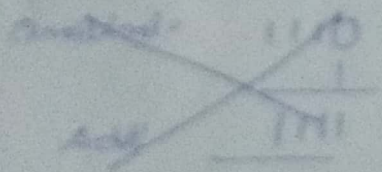
A = A+(-1);

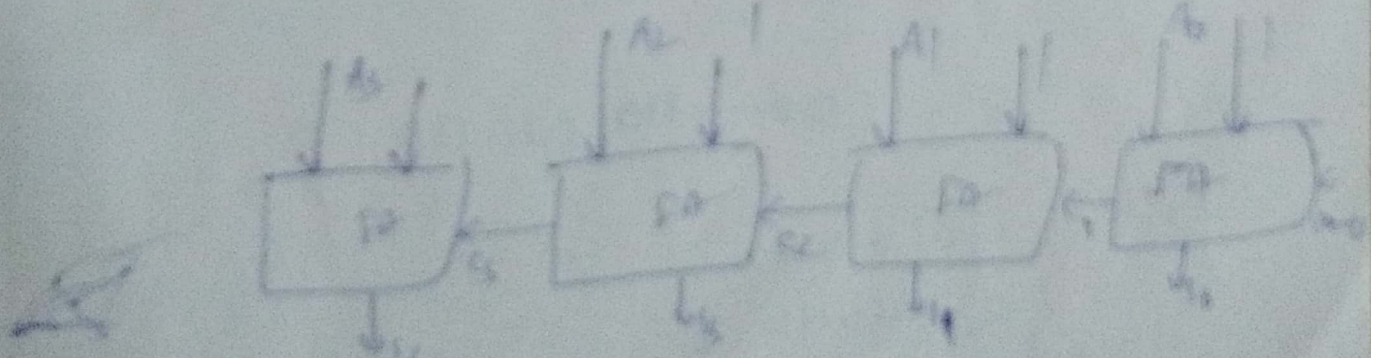
(-1) → 2's complement = 1111

0100

1111

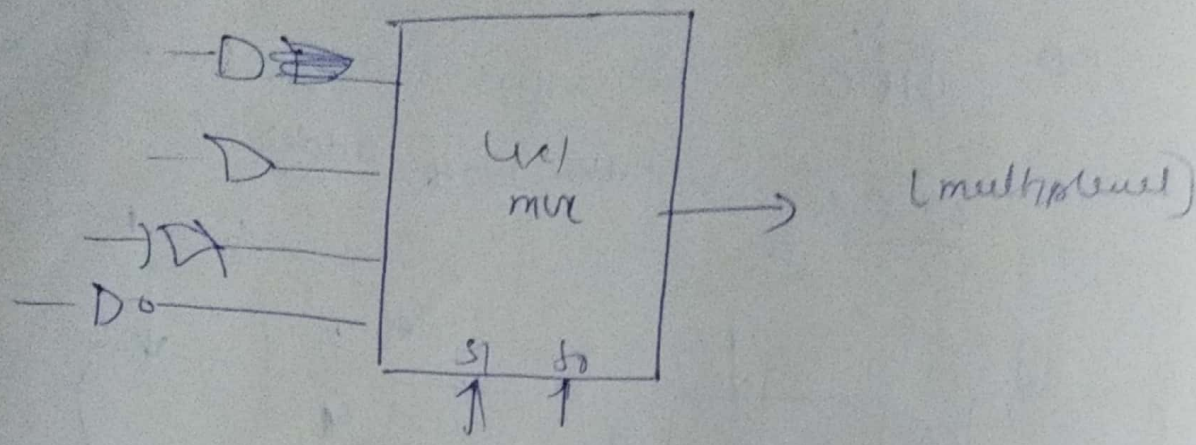
0101





→ for n bit decremeted → n full adders

Logical micro-operation circuits



S1	S0	
0	0	AND
0	1	OR
1	0	XOR
1	1	NOT

Shift micro operations :

1) Logical Shift

Left: The zero will be inserted at weight position

eg: 1010 → 0100 1101 → 1010

0 1 0 1 1 0 1 0

Right: 1010 1101 → 0101 0110

② Arithmetic shift

~~Left~~ Right shift

- MSB treated as sign bit
  - 0 → +ve
  - 1 → -ve

1 010 1101
→ 1 101 0110 (1)

Sign Preserved

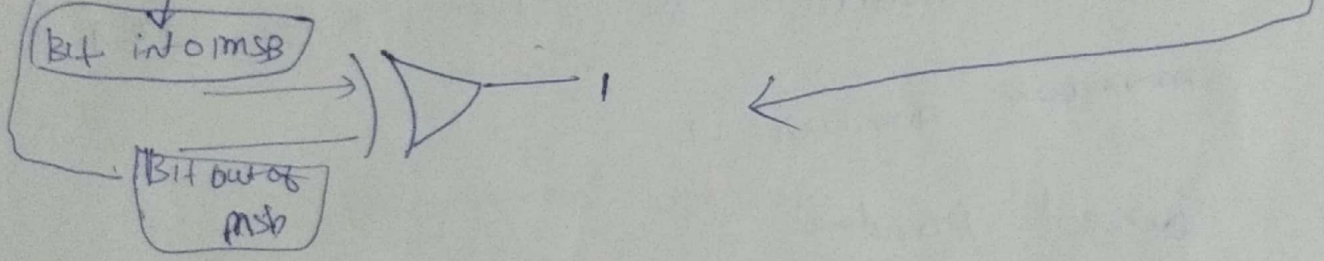
B Left shift [Zero inserted at weight position

1010 1101

1 0 101 1010 → ~~zero reverse~~ Sign reversed

↑ Zero

to know the which sign has changed for cpw



- exor gate → used for sign reversal

C Circular

Left 1 0 1 0 1 1 0 1

(1) 0 1 0 1 1 0 1 1

Right 1 0 1 0 1 1 0 1

(1) 1 0 1 0 1 1 0 (1)

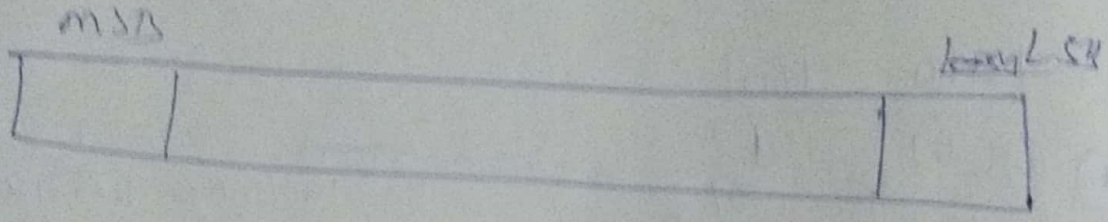
Data Representation:

1. Fixed point
2. floating point

Performance metrics

MIPS → Million instructions per second

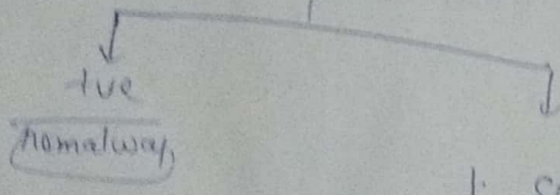
MFlops → Million floating point operations



* the maximum number for 8 bit register 127

minimum number = -1

fixed point



1. signed magnitude
2. signed one's complement
3. signed two's complement

sign magnitude

Take complement of sign bit only of positive number and copy the magnitude

Ex $-14 = +14 = 01110$

$+14 = (11110)$

$-14 = 11110$

Signed 1st compliment.

Take total Compliment of the number

Ex $-14 = +14 = 01110$ includes sign bit

$+14 = 10001$

$-14 = 10001$

signed 2's compliment

$-14 =$ ones complent +1

$-14 \rightarrow$ 1's compliment $= 10001 + 1$

$= (10010)$

ex -89

-89

$1\ 0\ 0\ 1\ 1\ 0\ 0\ 1$
$64\ 32\ 16\ 8\ 4\ 2\ 1$

signed magnitude:

$-89 = \underset{\text{sign}}{1}1001001$

$+89 = (01011001)$

Signed once complmt

$= (10100110) \rightarrow$ Total complmt of +89

signed 2's complmt

$= 10100110$
$+\ 1$
$= 10100111 = -89 = 1011$

Signed 2's complnet

$-89 =$

$+89 = 01001001$

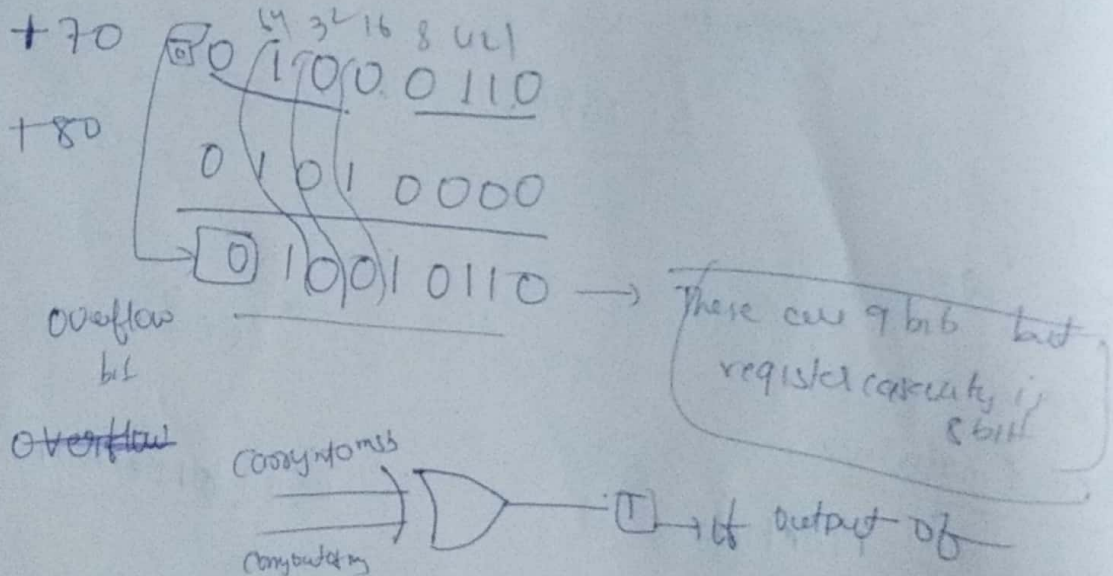
2's com → 10110111

$-89 = 10110111$

Overflow detection circuit

→ The maximum ~~possib~~ +ve value which can store in an 8 bit register. ~~+127~~ +127

Consider a system to perform a 8 bit operation with a register capacity as 8 bit



When ~~and~~ Then o/p=1 Now processor supplies an additional flipflop ~~st~~ now o/p = +150

→ When two n bit values are added and if the result is (n+1) bits, the additional bit cannot be accommodated into the register, these situations need to be intimated to the microprocessor with help of overflow detection circuit

→ An ~~[illegible]~~ EX-OR gate is used to determine the overflow (previous page)

→ When overflow is being detected microprocessor supplies an additional flip-flop to accommodate the additional bit and treat it has sign bit

→ The (n+1) bits is shown as a result to the user

* Floating point representation :

$M \times R^{E}$ — E → exponent

M ↓ Mantissa, R ↓ Radix

R will not occupy any space

0.00005 → memory [0.00 | 5] → 3 no.

= (5×10^{-5}) → [ ]

5 bits, 3 bits, 4 bit for sign

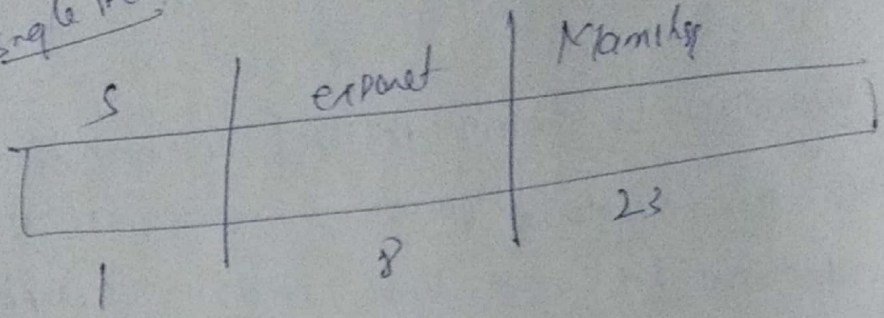
So total → 7 bits

So due to floating point Representation memory size reduced

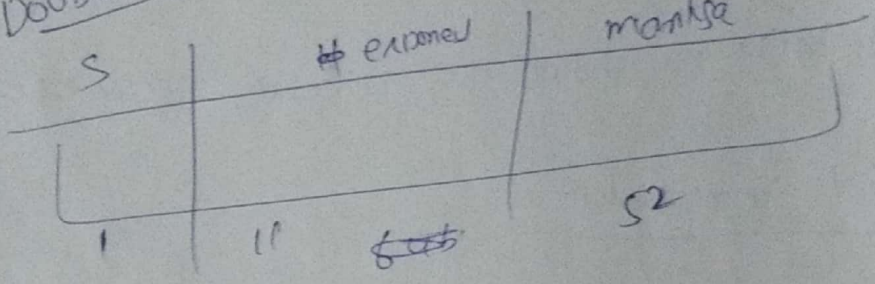
To reduce the ~~IEEE flo~~ memory location they introduced IEEE floating point

① single precision format (32-bit)
② Double point precision format (64bit

Single precision



Double precision



64bit

* $(87.67)_{10}$ convert to single point precision format

steps
① Convert the given decimal in to binary

$(87.67)_{10}$ $87 = 0101\ 0111$

$+87 = 01010111$

2 | 87
2 | 43 - 1
2 | 21 - 1
2 | 10 - 1
2 | 5 - 0
2 | 2 - 1
2 | 1 - 0

$(0.67)_{10}$ = 0.67 x 2 = 1.34 — 1

0.34 x 2 = 0.68 — 0

0.68 x 2 = 1.36 — 1

0.36 x 2 = 0.72 — 0

0.72 x 2 = 1.44 — 1

0.44 x 2 = 0.88 — 0

= 101010

$(0.67)_{10} = 0.101010$

$(87.67)_{10} = (01010111.101010)_2$

(2) Normalize the result

$1010111.101010 \times 2^0$

we have move left(-ve) Right such that point before number is non-ve

$1.0101111010100 \times 2^{+6}$

(3) Calculate the Biased exponent

$E = Exp + 6$

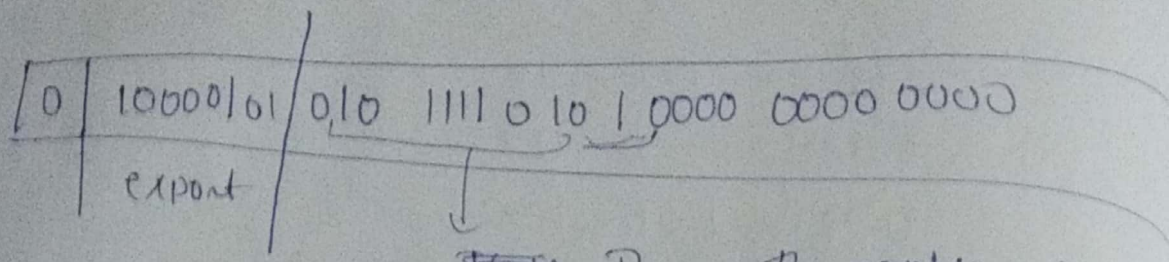
$BE = E + 127$

$= 6 + 127$

$= +133$

$(133)_{10} = (10000101)_2$

(4) Represent them in formal



This is the mantisa part after normalisation in 3rd step.

(4) now Group it into four bits inorder to represent in the hexadecimal

= 42Af5000h

home work :-

1) $(-97.66)_{10}$
2) $(-129.27)_{10}$

1) -97.66

sol $-97 = (1)\ 1100001$

sign

$= 11100001$

$0.66 =$

$0.66 \times 2 = 1.32 - 1$
$0.32 \times 2 = 0.64 = 0$
$0.64 \times 2 = 1.28 = 1$
$0.28 \times 2 = 0.56 = 0$
~~$0.48 \times 2 = 0.96 = 0$~~
~~$0.96 \times 2 = 1.92 = 1$~~
$0.56 \times 2 = 1.12 = 1$
$0.12 \times 2 = 0.24 = 0$

$= 0.101010$

2	97
2	48 – 1
2	24 – 0
2	12 – 0
2	6 – 0
2	3 – 0
2	1 – 1
	0 – 1

$= (1100001)$

(-99.60)$_{10}$ = (1100001.101010)$_2$

② Normalize

$1.100001101010 \times 2^6$

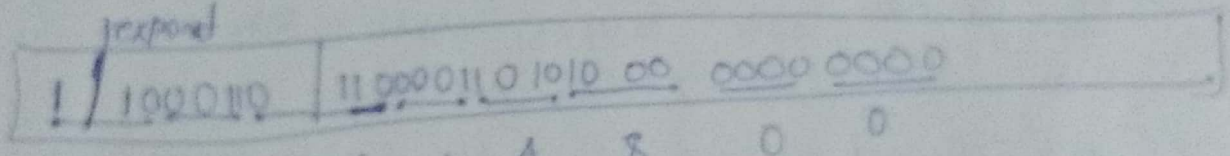
$1.10000110101000 \times 2^6$

④ E = 127

7 + 127 = 134 = 1000 0110

128	64	32	16	8	4	2	1
1	0	0	0	0	1	1	0

③ Decimal form to final



= (6361A800)$_{16}$

a) −129.27

Sol −129 = 11000 0001

0.27 =

0.27 × 2 = 0.54 → 0
0.54 × 2 = 1.08 → 1
0.08 × 2 = 1.6 → 1
0.16 × 2 = 1.2 → 1
0.32 × 2 = 0.64 → 0
0.64 × 2 = 0.8 → 0

1000 0001

Max 6th 6 times

2	129
2	64 - 1
2	32 - 0
2	16 - 0
2	8 - 0
2	4 - 0
2	2 - 0
	1

Instructions :-

I/O organisation :

I/O commands :

→ Input command, output Command, control command and status command.

Peripheral Devices :

1) The devices which are connected externally to the computer are known as peripheral device

2) CPU use the following set of I/O commands to interact with I/O device

1) Data input - The I/O device supply data to the processor

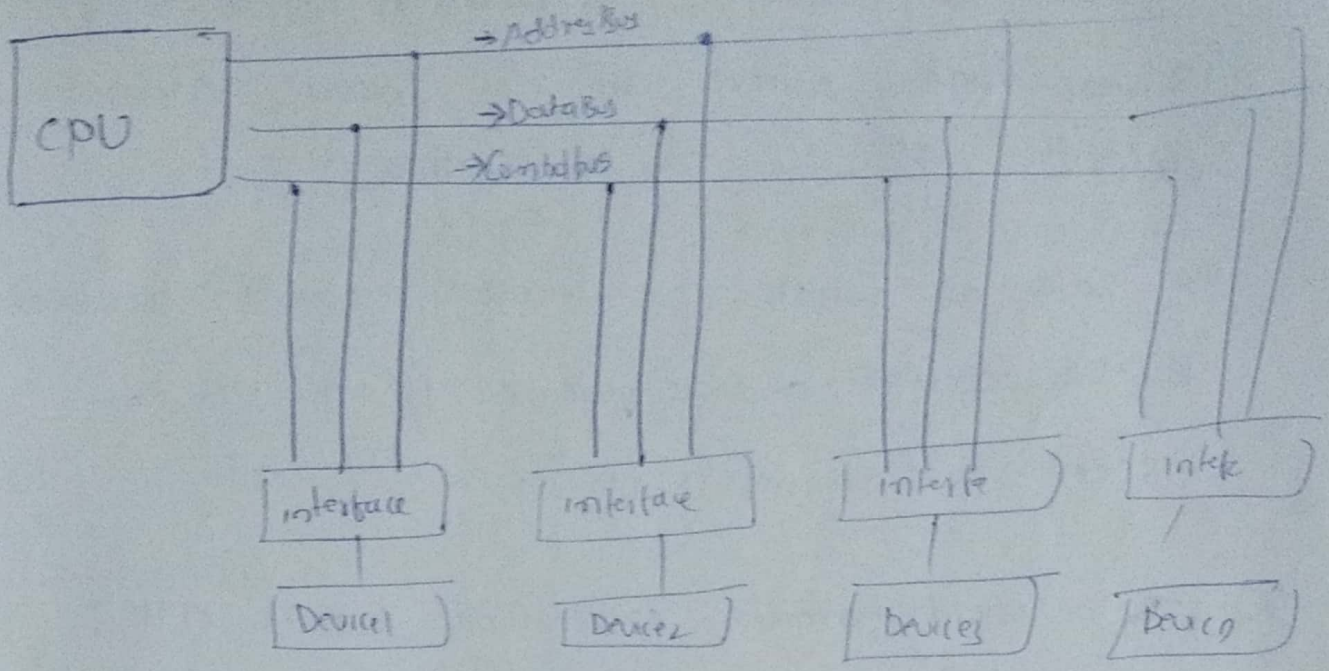
2) Data output - CPU supplies data to an I/O device

3) Control command :- The human readable format is converted into to stream of 0's and 1's for the machine to understand and Perform the necessary operation)

* Once the control word is loaded into to device interface then only the device starts functioning

4) Status Command :-

The amount of work done will be intimated to user.



Necessity of an I/O interface

1) Synchronisation mechanism is required
2) Transfer rates
3) word formats of CPU and I/O devices are differe
4) difficult to remember all I/O device operations

explanation for above

→ CPU is an electronic device which work with speed of electrons where as I/O devices are electromechanical where laptops work to transfer (or) receive data from I/O device. Therefore Synchronization mechanism is required

→ The rate at which CPU transfers the data is very high where an I/O device cannot accept at the rate (or) time

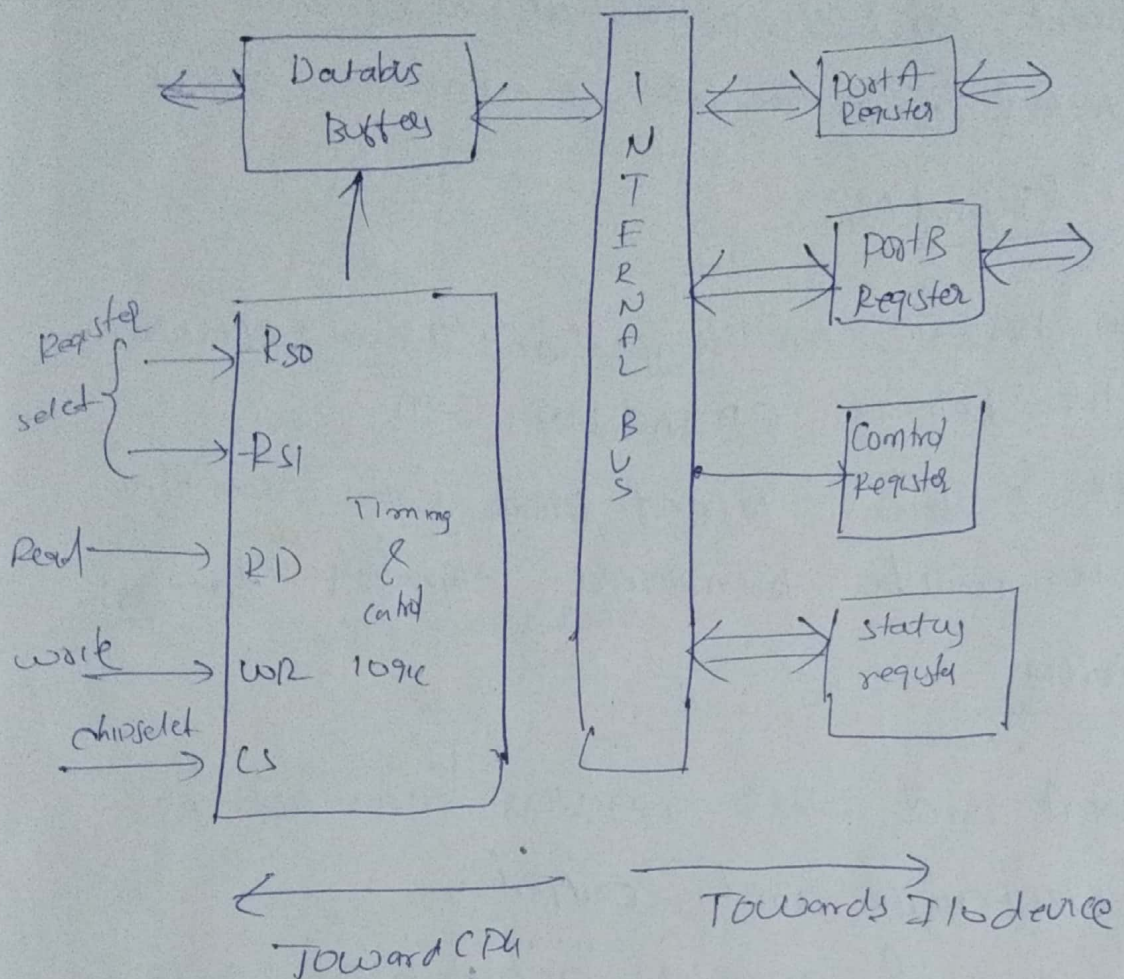
The Rate at which CPU transfers can be accepted by the interface and stepdown to an I/O device speed and transfers to it.

→ The word formats understood by the CPU and I/O devices are different. The interface manages the block transfers (256 bytes) in to word transfers

→ · As the no of I/O devices increases it will be difficult for the CPU to remember and operate the no of I/O devices.

The interface maintains the device address and reduce the burden of the CPU interms of remembering I/O addresses

example I/O of Interface



CS	RS1	RS0	
0	X	X	no Operation
1	0	0	port A Register
1	0	1	Port B
1	1	0	Control register
1	1	1	status register

→ The interface will be selected with help of chip select input. after that the required register is selected with the help of Register Select input (RS_0 and RS_1)

→ It can perform an "I/O read" and "I/O write operations with the help of RD and WR pin

→ So the data to and fro from the CPU will be transferred through databus buffers

→ portA and portB registers can act as both input and output

→ The control register accomodates a control word based on which The I/o device starts performing I/o operation

→ These status register will make the CPU to know the status of the work completed

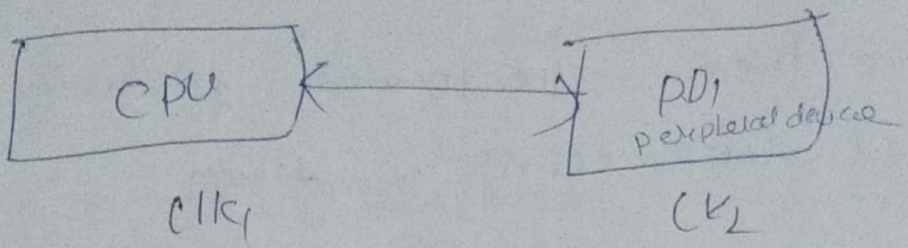
Asynchronous Data transfer Method :-

Synchronous:

The communication is transfer between 2 devices having a common clock

Asynchronous

The The interaction between 2 devices under the different clock pulse

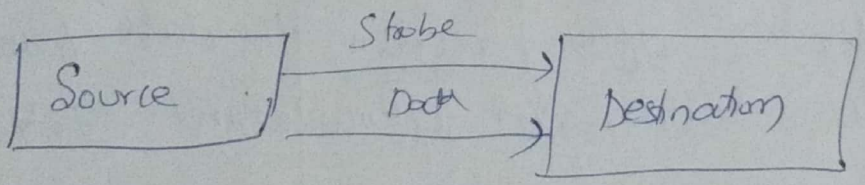


In asynchronous Data transfer

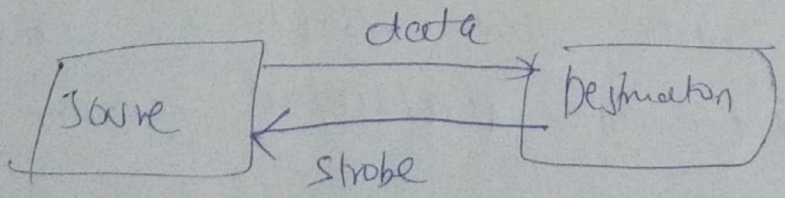
1. Using strobe (signal)
2. Using handshake

1. Using strobe :-

(a) Source-initiated strobe :-



(b) Destination initiated Strobe

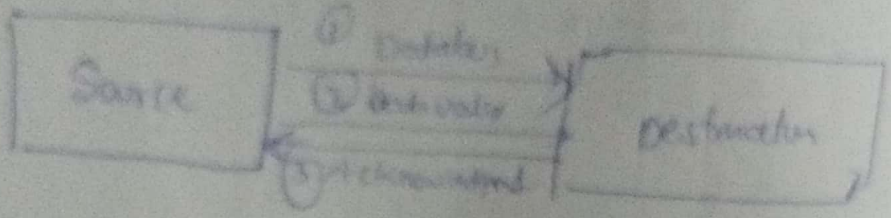


The disadvantage with these method the source never knows whether the data is received by the destination or not.

To overcome these disadvantage 2nd method is proposed using Handshake

② Using handshake :

① Source initiated handshake



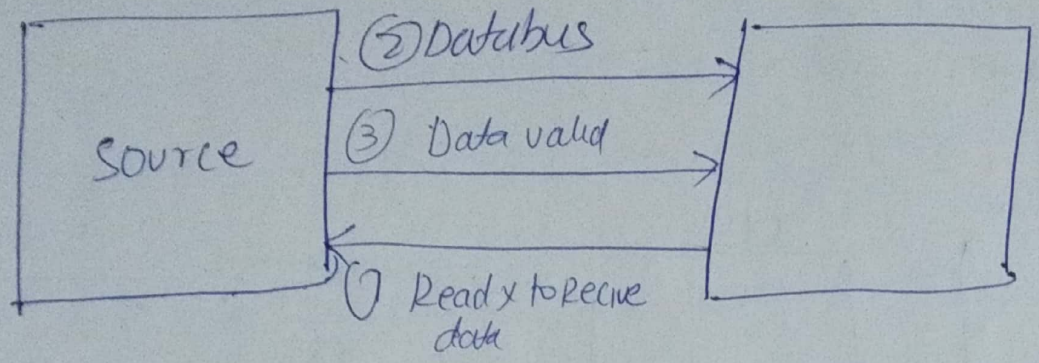
→ The data is placed on databus and enables datavalid signal

→ The destination after receiving the data from databus sends an acknowledgment to the source.

→ After receiving the acknowledgement the source disables the datavalid signals and put another set of data onto the databus and again enables datavalid signal

→ These process will be continued until destination stops sending the acknowledgment signal

(b) Deshnation-initiated handshake



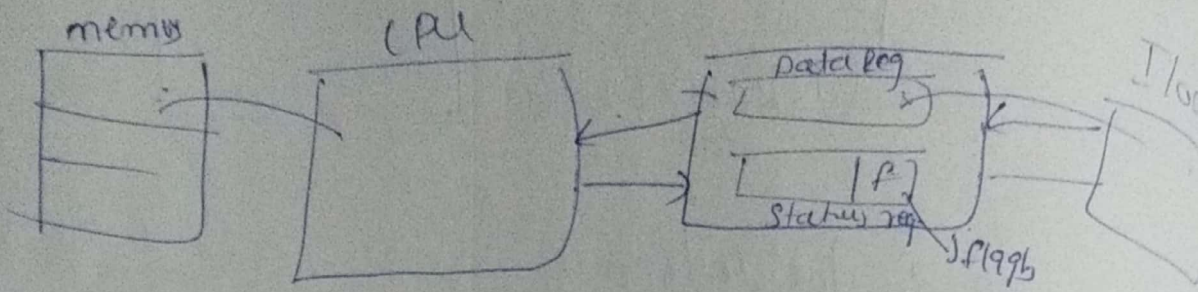
In these method the deshnation sends a ready to recive data signal to source. Upon source receiving the ~~reading~~ "ready to receive" data signal. It put the data on to the databus, and enables data valid signal.

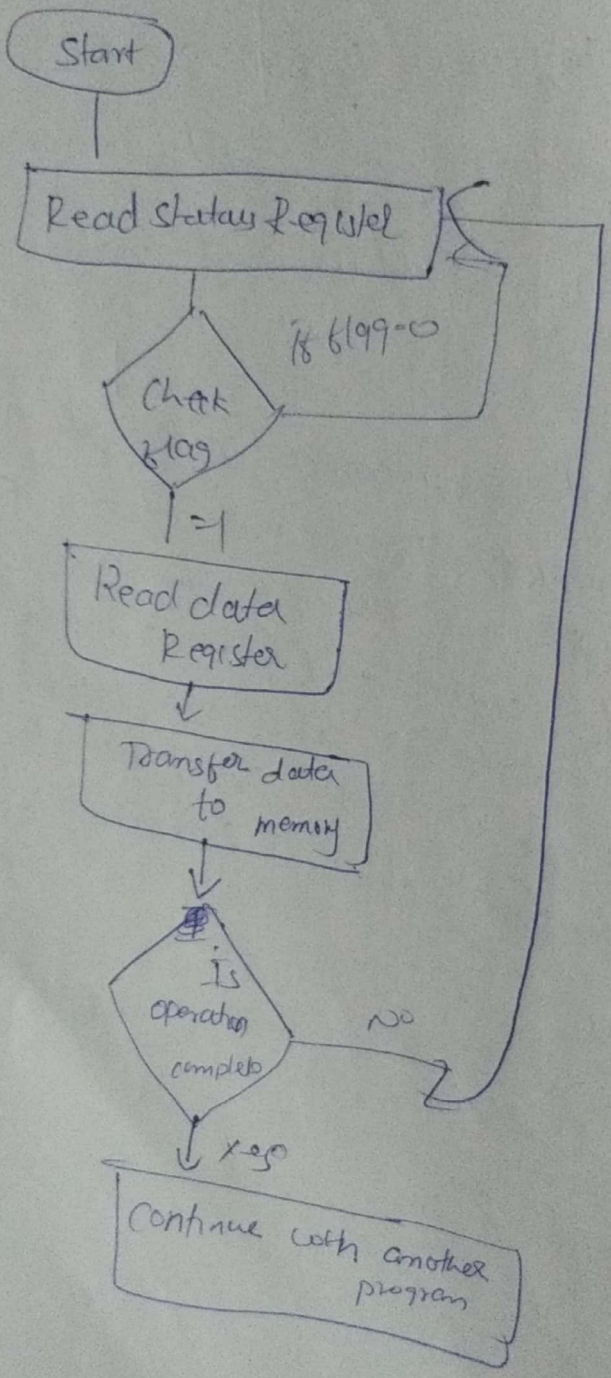
→ After the data is received by deshnation, ~~it~~ it send ready to receive data ~~sig~~ signal again. Upon receiving it source disables the data valid signal and put another set of data on to the databus and validates the datavalid signal. These process will be continued until deshnation stops sending ready to receive data signal

Modes of Data transfer :

1) Programed I/O

→ Here set of ... already defined. CPU follows that instru...





CPU:

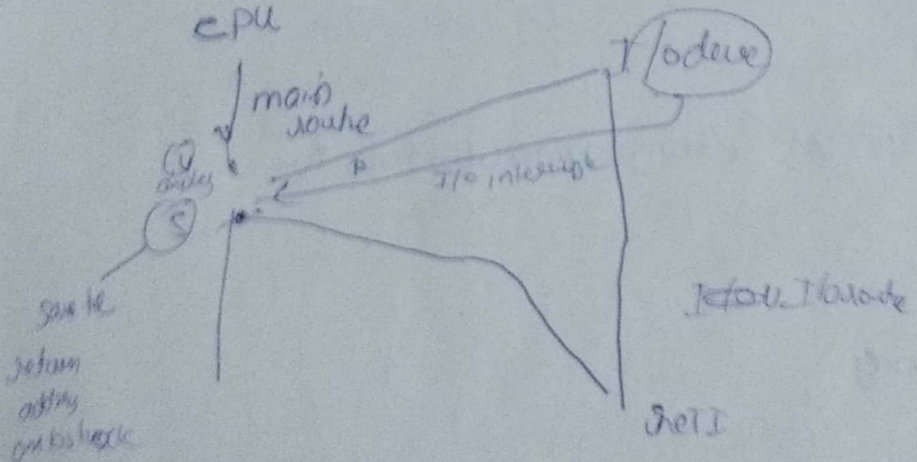
Bus x read-write

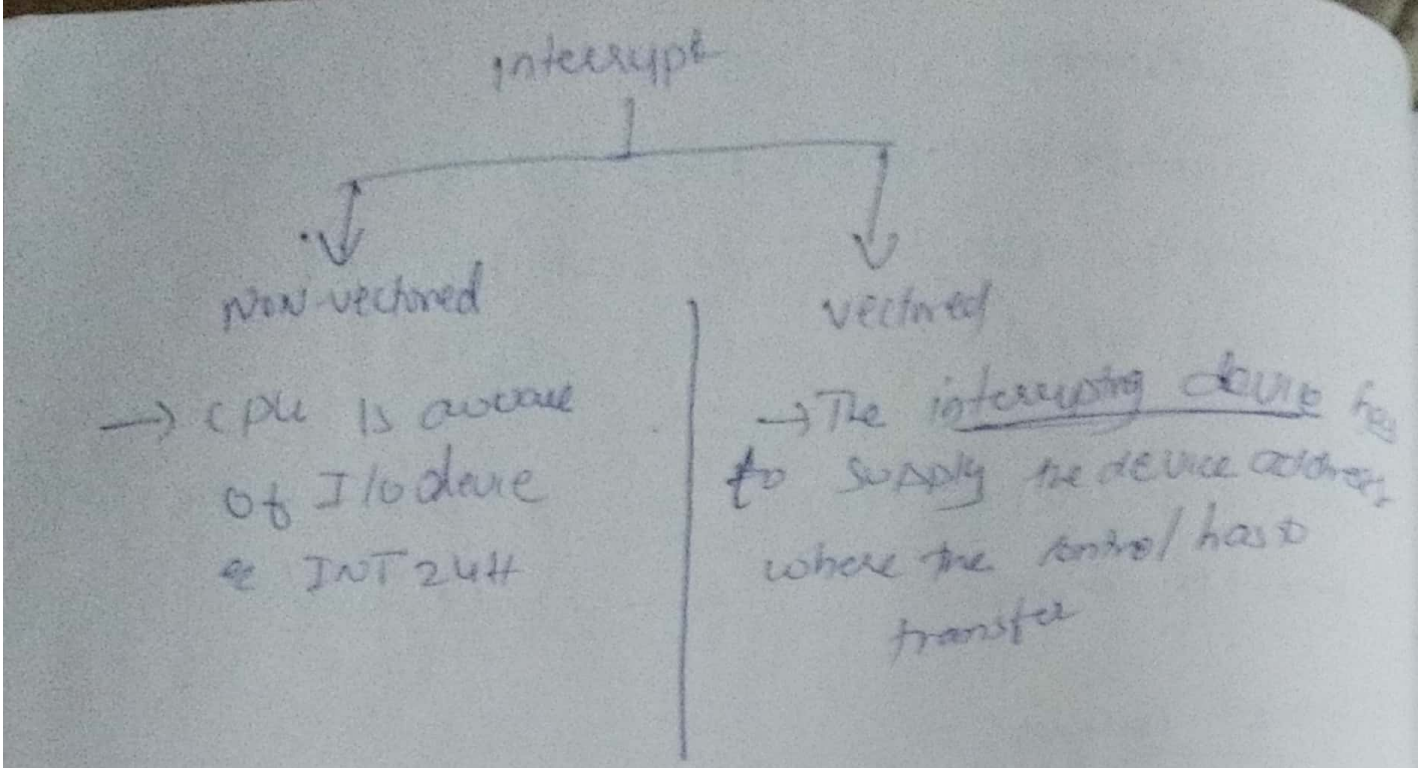
→ The disadvantage with these method is cpu wait for an I/o device to transfer data, Making the speed of the system come down to the speed of an I/o device.

→ When an os finds cpu ~~busy~~ to give some task it appears to be "busy needles state"

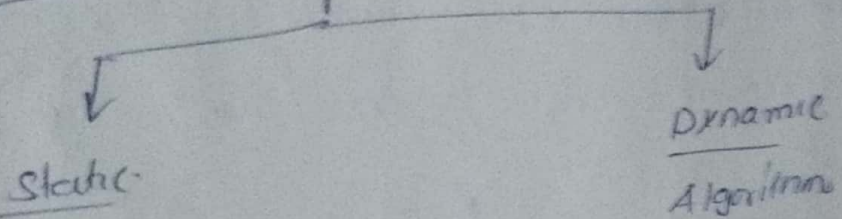
Interrupt initiated I/O :-

The disadvantage of cpu waiting for an I/o device will endup using these method i.e whenever I/o device is ready it sends an interrupt to cpu, ~~when~~ then cpu completes the current instruction execution and process the I/o device request,





* Priority interrupt :



1) Daisy Chaining

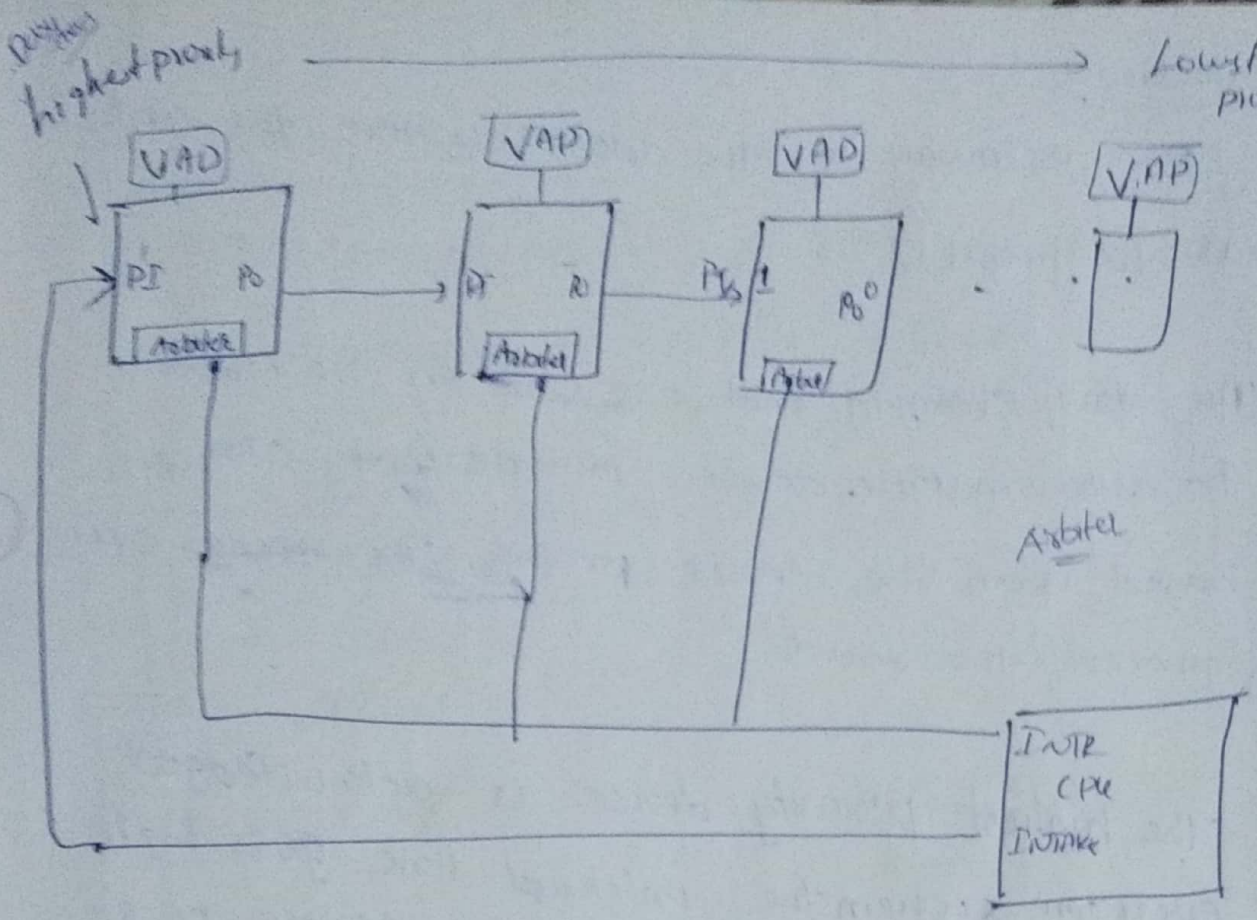
2)

Static priority :-

The priority is set during manufacturing the

Dynamic priority

→ priority varies



VAD = Vector address

PI = priority In

PO: priority out

If PI=1, PO=0

* It means that current devices has given interrupt and processing by CPU (device

~~If PI=1~~

even though device 3 has made request

Then highest priority is given to device. first it will check PI

if

→ Interrupts of of 2 category

1) vectored:

The interrupting source has to supply the address where the control has to transfer

Non vectored:

CPU is aware of the address where the control has to transfer

* The daisy chaining method is used in the computer to allow multiple devices to interrupt CPU, based upon the device priority. The ~~process~~ CPU process the request

* The highest priority device is acknowledged every time when the interrupt line goes high if he doesn't want to use the source of the CPU, it he passes the priority to the next device

* CPU process the device whose ~~priority~~ PI=1 and P0=0

* The device can disable priorities of devices which are having lesser than the current devices and the devices interrupt request which are having more than the current device priority cannot be ~~disabled~~ disabled.

Arbitration logic:
conflict resolving logic

→ whenever more than one device process request, the processor has to judge to which device it has to process

DMA

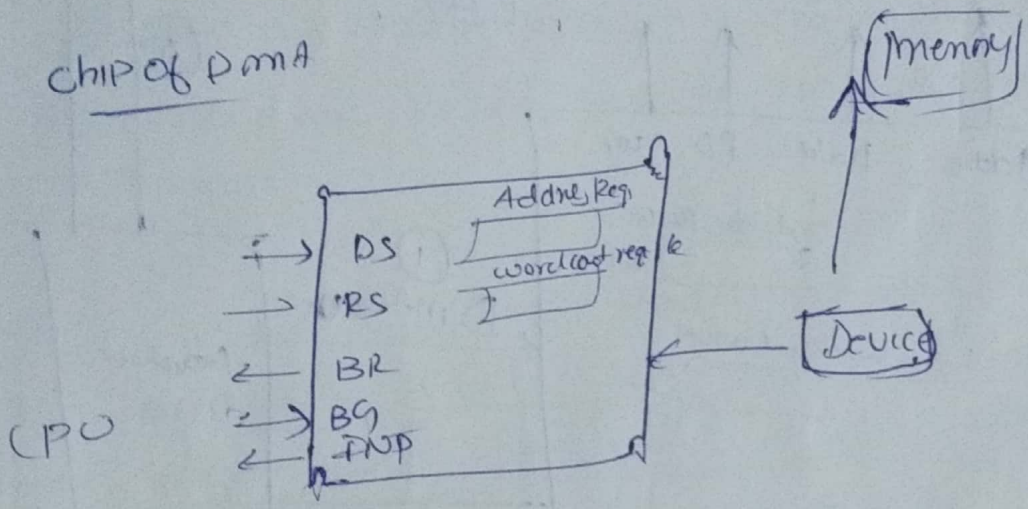
(Direct memory access)

→ Removing CPU from picture and letting the peripheral device to transfer data directly to memory

Combol Signals of DMA

- BR: Bus Request
  BG: Bus GRant
  INTR: Interrupt

Chip of DMA



Address Register

It store

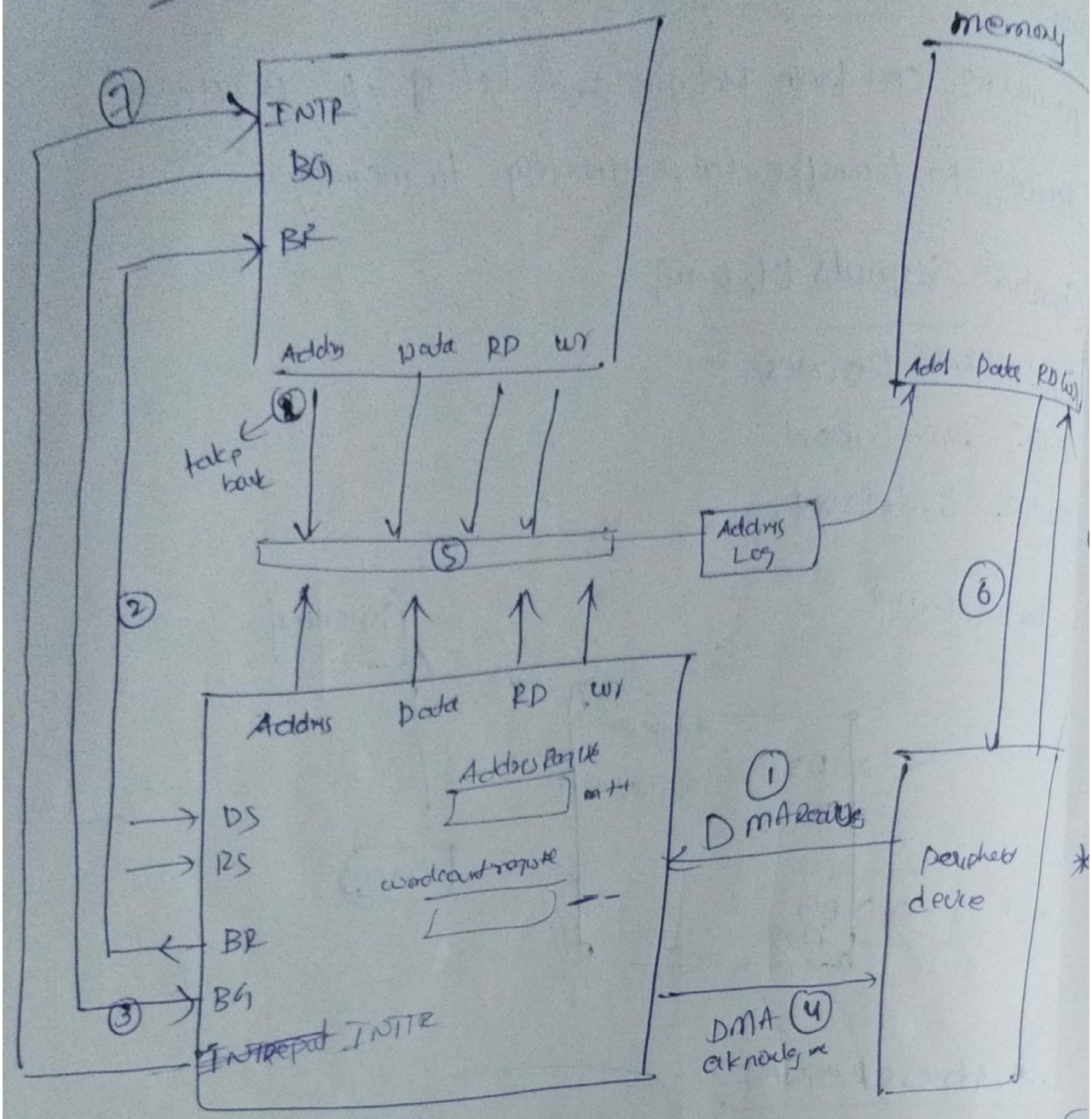
→ The address of the ~~memory~~ wh location (@) of the memory where data needed to be transfer

→ It will be incremented

→ DS → device-select (chip select)

RS → Register Select (either Address Register (@) word Counter)

Dma transfer



(1) peripheral device sends a DMA request to the DMA controller

(2) Upon receiving it DMA issues Bus request signal to the CPU.

(3) After receiving the Bus request CPU gives

the Address register and word count register of DMA controller and then grants the bus by issuing busgrant signal

(4)

→ DMA controller Acknowledges the peripheral device that the request is going to be processed

(5) & (6)

The DMA takes the bus and visits memory location and then transfers data from memory to peripheral device (or) from peripheral device to memory

* for each one word of data transfer the content of address register will be incremented by 1 and word count register will be decremented by 1

(7)

The DMA controller ~~is sues~~ issues INTR signal to the CPU once the wordcount register value becomes "0"

(8)

After receiving INTR signal CPU examines the wordcount register value if it really becomes '0' then the CPU takes the busback

DMA performs 2 types of data transfer operations

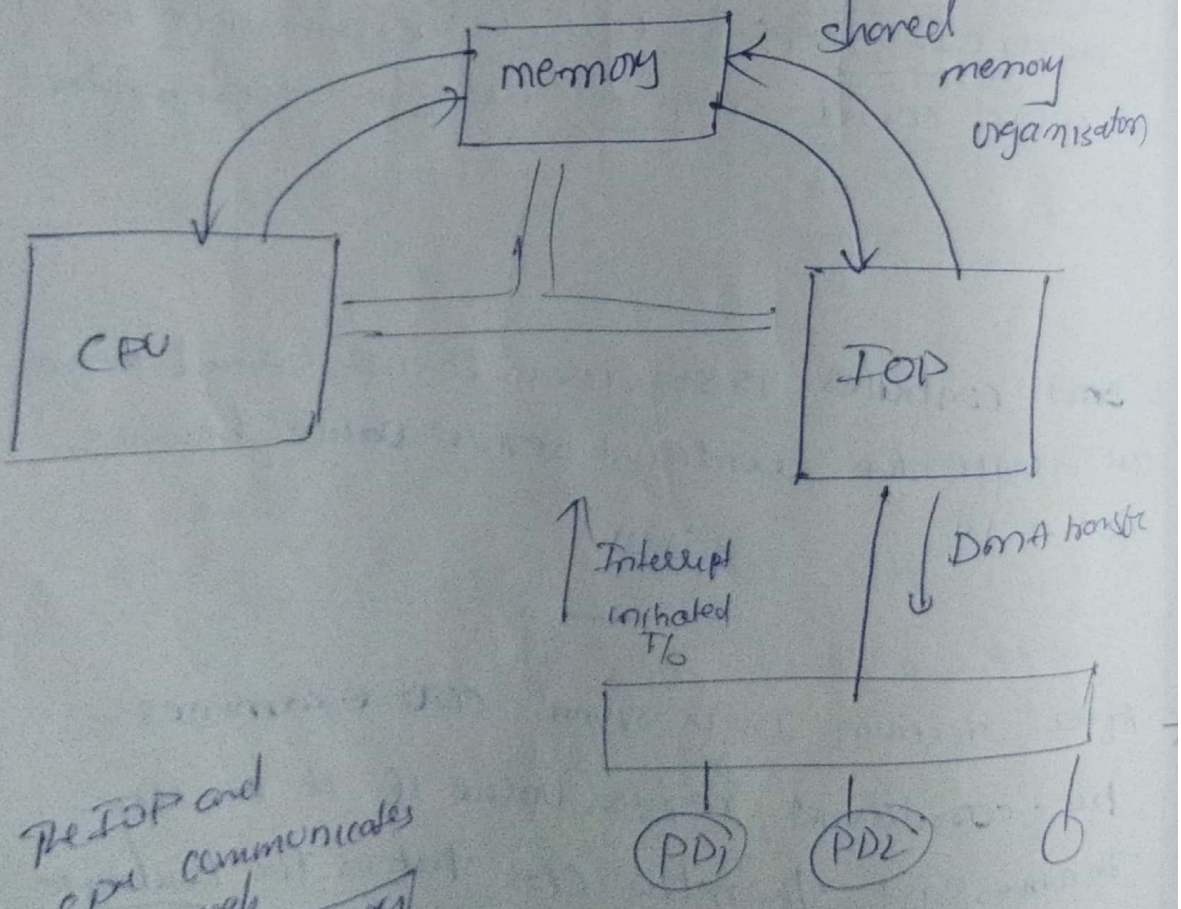
1) cycle stealing
2) Burst transfer

* cycle stealing:
It performs one word data transfer at a time
* It is preferred for transferring less amount of data

* Burst transfer:
Bulk amount of data is transfered

The DMA almost performs Burst transfer

IOP (Input output processes):



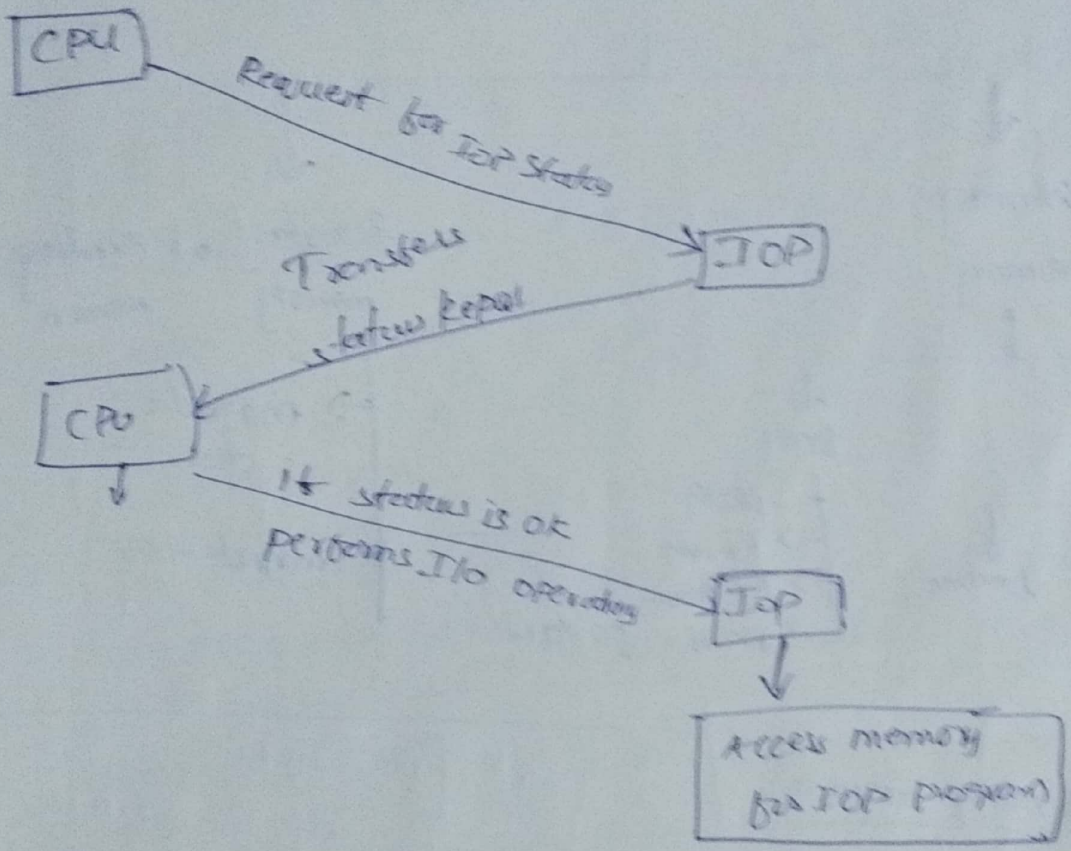
The IOP and CPU communicates through shared memory

The CPU will request IOP to send the status

→ IOP put in memory and CPU reads from memory and when status is ok

→ Also when CPU grants permission IOP will access to IOP routines and perform the I/O operations

CPU - IOP Communication



NOTE: example for Intel

for 8086 . . 8089 is IOP and IBM 360/370 is IOP

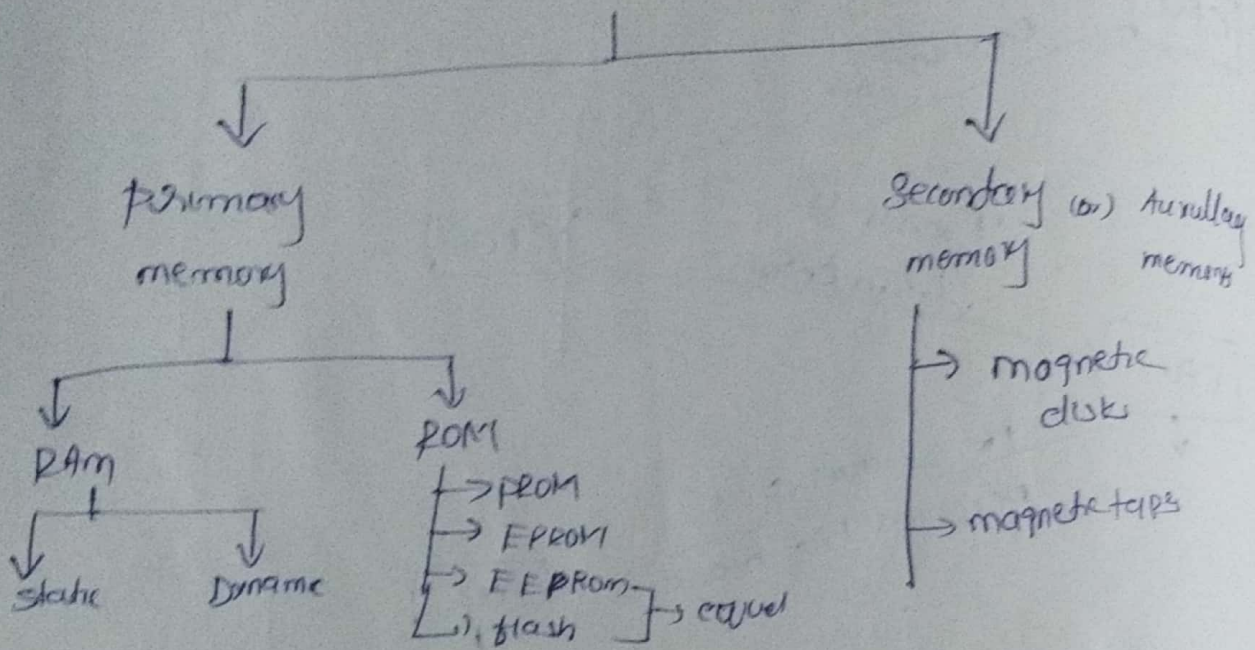
NOTE

→ Peripheral devices uses interrupt initiated I/O method to interrupt input output processors (IOP)

→ IOP uses the DMA method for transfering data between memory and I/O devices

→ CPU and IOP communicate with each other by using "shared memory organization" method

* Memory Organization *



Memory hierarchy

Note: [flash drives are also secondary memory as we know that secondary memory are slower that speed of transfer we use flash drive but flash drive are not fast]

(1) ROM

16 ~~RAM~~ = 512×8

Then no of ~~chip~~ 128×8 chip required to build memory

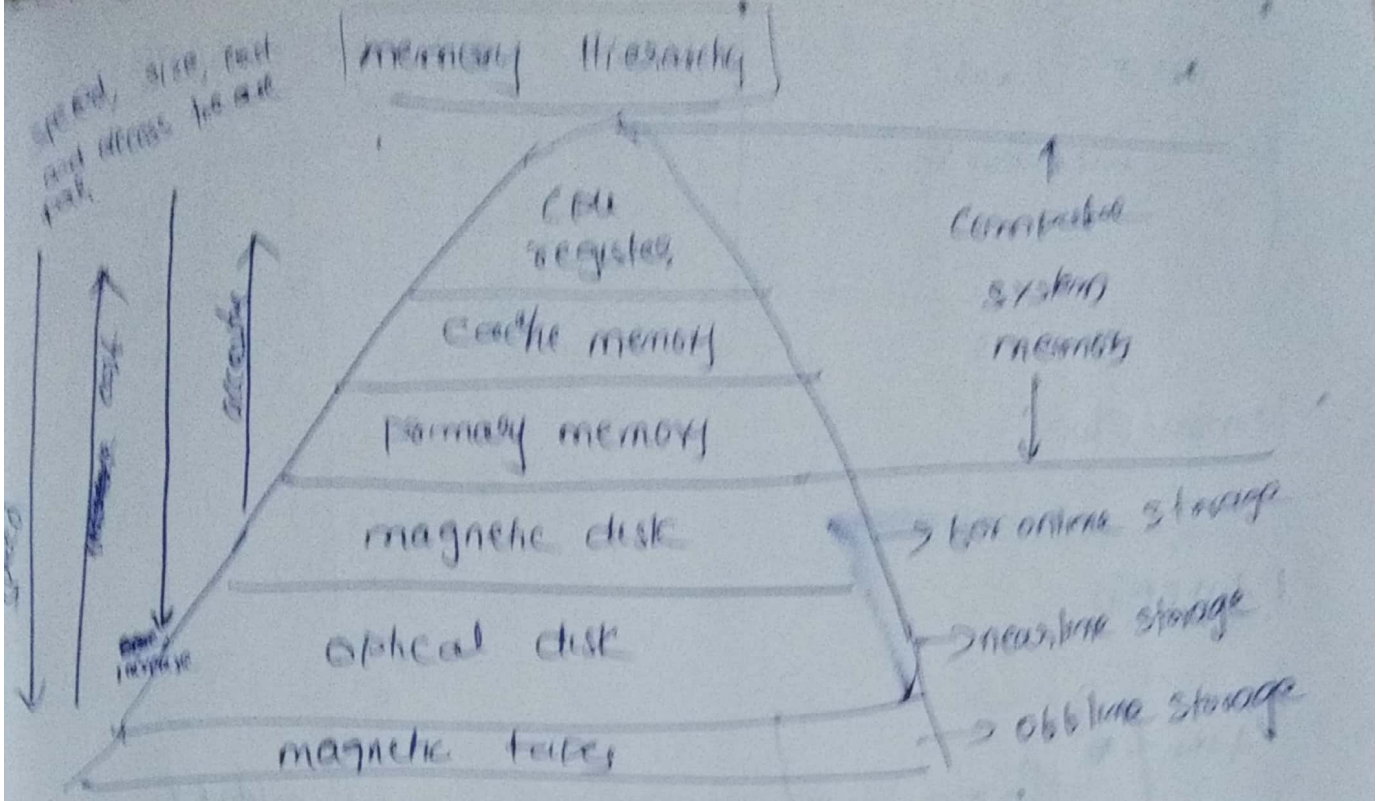
Storage of RAM - 128×8 — (1)

128×8 — (2)

128×8 — (3)

128×8 — (4)

So we require 4 128×8 chips for RAM



Main memory: [Composed of both RAM and ROM chips]

→ 2 types of RAMs

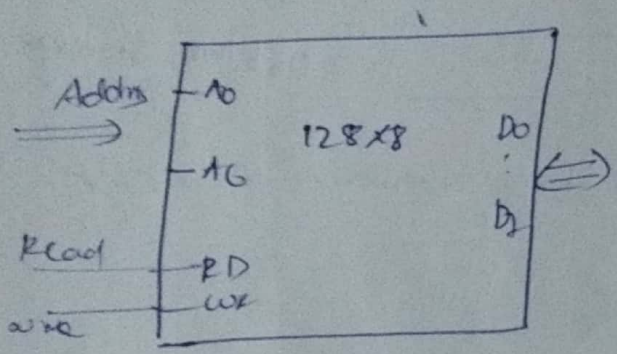
Static RAM	Dynamic RAM
→ It is made up of flipflops (most pbllit	→ It is made up of capacitors 1 capacitor + 1 Transistor = 1 flipflop
→ Density of storage is less	Density of storage is more
→ No periodical refreshment is required	Has capacitor tends to leak so periodicall refreshment is required
→ Costlier	cheapeal
→ Retains data for longer period of time	Retains data for short duration
more faster (high speed)	Comparatively slows than SRAM

used cache memories

used for main memories

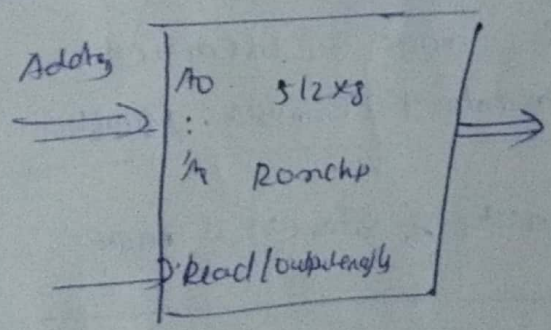
Memory Chip:

RAM chip



no of addrs lig
= 7

Rom chip



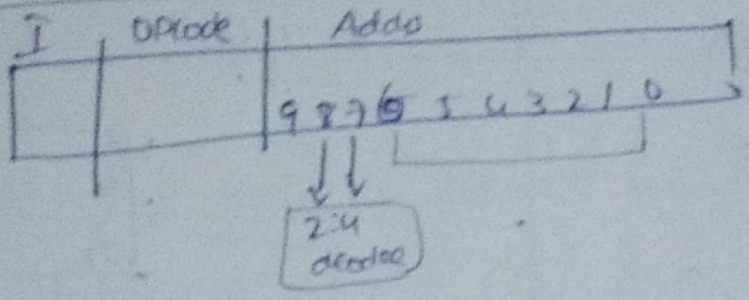
no of addrs lig = 9

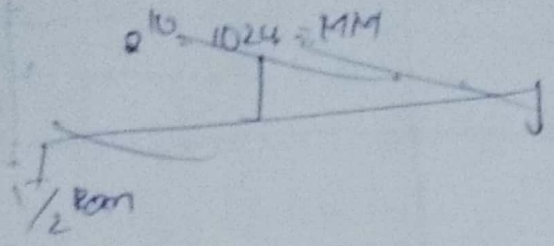
OE = output enable

NOTE

ROM occupies more storage than the RAM size

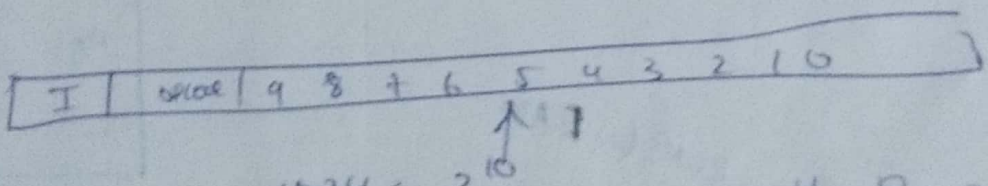
CPU - Connection to main memory





Consider a Computer system consist of 1024 x 8 has main memory. out of which exactly half is RAM and half memory is Rom. The Ram chip available 128x8 and ROM chip available is 512x8. how many chips are reaured to form main memory and specify the size of decoders used.

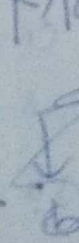
Sol



$1024 = 2^{10}$

Note

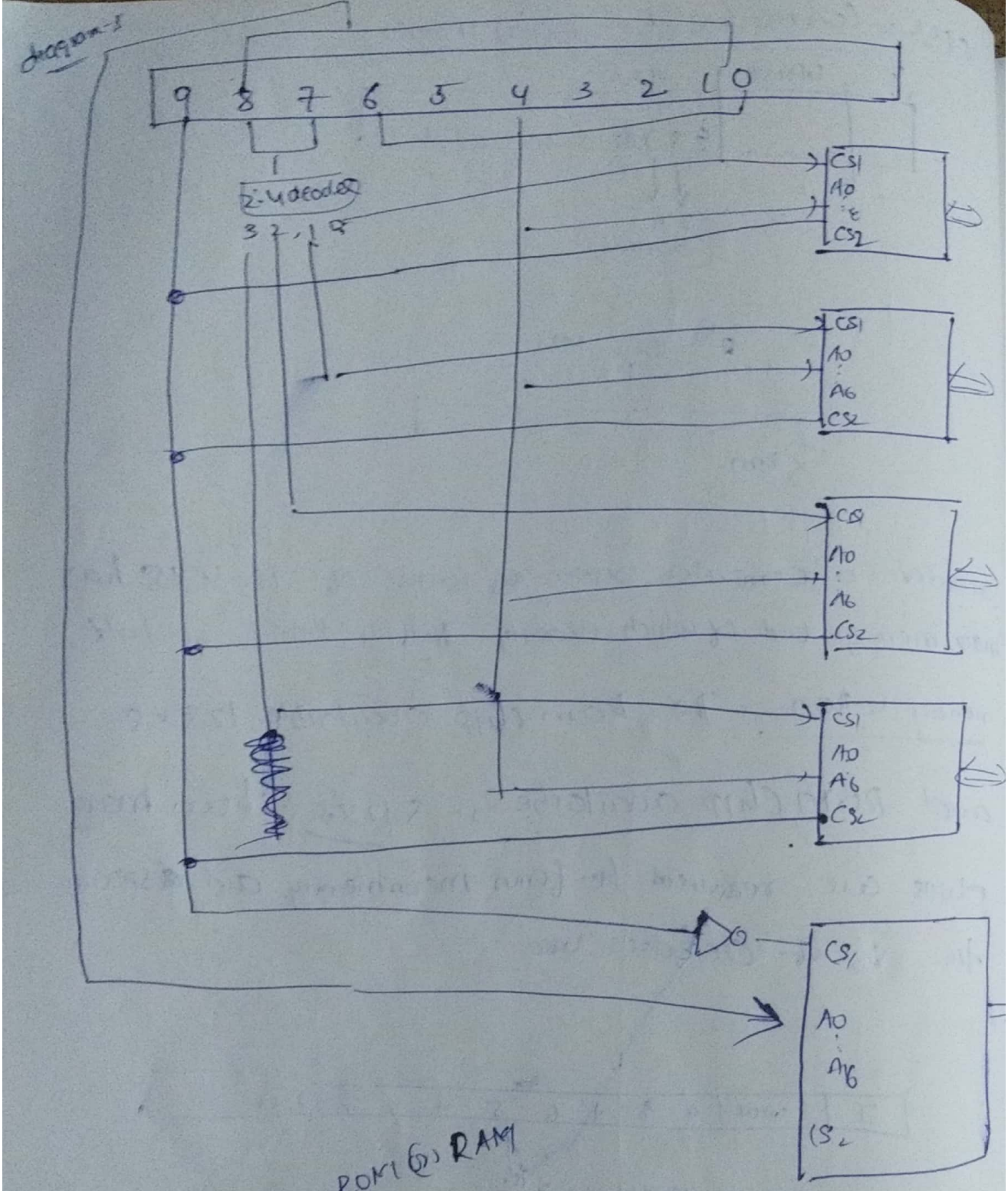
Main memory [both Ram and Rom]



If there are multiple ROM chips then will be address

→ The decoder size is dependend upon the no of chips

→ The dec



Q → Used to select ROM & RAM

Memory Address map

Ram1	0 – 127
Ram2	128 → 255
Ram3	256 – 383
Ram4	384 – 511
Ram5	512 – 1023

1A) How many 128x8 RAM chips are needed to provide memory capacity of 2048 bytes?

1B) How many lines of address bus must be used to access 2048 bytes of memory how many of these lines will be common to all chips?

1C) How many lines must be decoded for chip select? Specify the size of decoder

Answers

~~x~~ 8

1A) $128 \times 8 \longrightarrow 2^{11} \times 8$

$2^7 \times x \longrightarrow 2^{11}$

$x = 2^4 = 16$ chips

$x = 16$ chips

How much it should be multiplied to get 2048 bytes

1B) + 11 → address lines
+ 8 → data lines

7 are common to all

1C) 4:16

Extend the memory ...
RAM and 4096 bytes of ROM. Lead to
Load the memory address map and indicate what
size decoders are needed

8 RAM chip size : (4096)

Rom chip size : (4096)

** Calculation for RAM **

→ $2^7 \times x = 2^{12}$

$(x = 25)$

$x = 32$ 128×8 chips required

→ decoder size

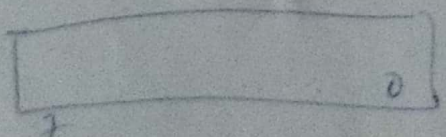
5:32

** calculation for the Rom:

- $2^{12} x = 2^9 \times 3$

$x = 8$

8 512 Rom chips reqd



8 3:8 decoder.

Rom chips of 1024 x 8. The computer system needs 2^K bytes of RAM, 4K Bytes of ROM, and 4 interface units, each with 4 Registers. A memory mapped I/O configuration is used

a) How many Ram and Rom chips are needs

b) Draw a memory address for the system.

RAMs chips

- $256 \times 8 = 2K (2^{11})$
- $= 2^8 \times 1K = 2^{11}$
- $= 2^8 \times 2^3 = 2^{11}$
- 8 RAM chips

3:8

Rom chips

- $512 = 2^{10} \times 1K$

4 Rom chips

2:4 decoder

Total main memory size is 6K

for this type of questions Given information

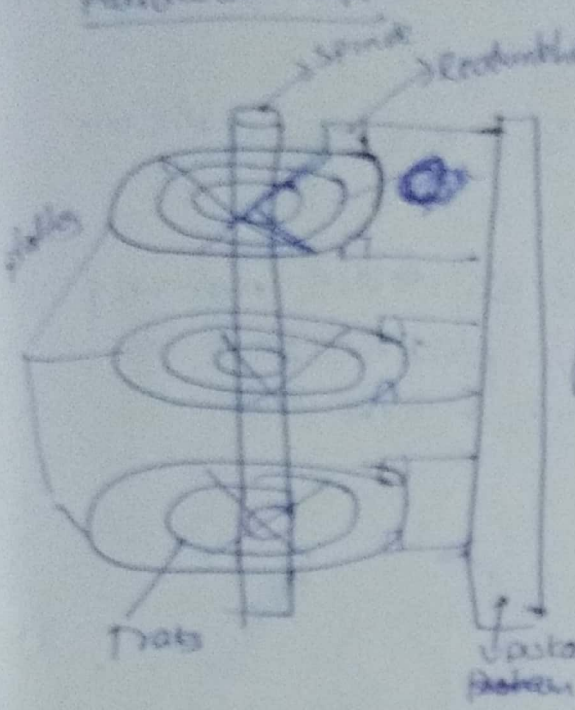
1) RAM, Rom space (or) directly give total memory size and memory conditions
2) available RAM and Rom chips
3) finds No of RAM chips and Rom chips
4) find size of decoder for RAM and Rom

Auxiliary memory:

(Helping)

- Magnetic disk
- Magnetic tapes

Hardware support



Rotational delay: delay caused by magnetic disk moving disk

Seek delay: delay caused by R/w header

Disk consists of

* platters:

These platters consists of concentric circles known as tracks. These tracks are further divided in to sectors and sectors consists of blocks

→ To access data from the disk 2 types of delays are encountered

(i) Rotational delay: It is the delay caused moving disk

(ii) Seek delay: It is delay caused by R/w header

→ R/w header restricts its movement towards north and south direction directions only

→ where as disk can rotate 360°

→ It is an electromechanical device where mechanical operations are involved in accessing data

→ The rate at which the data is saved should be equal to rate at which it is retrieved

→ If there is a mismatch then loss of data will occur

→ It is a semi-random access device meaning that the R/W header can be placed randomly at any sector, but processing will be done sequentially

Magnetic tapes :

→ These devices are obselete (no not using)

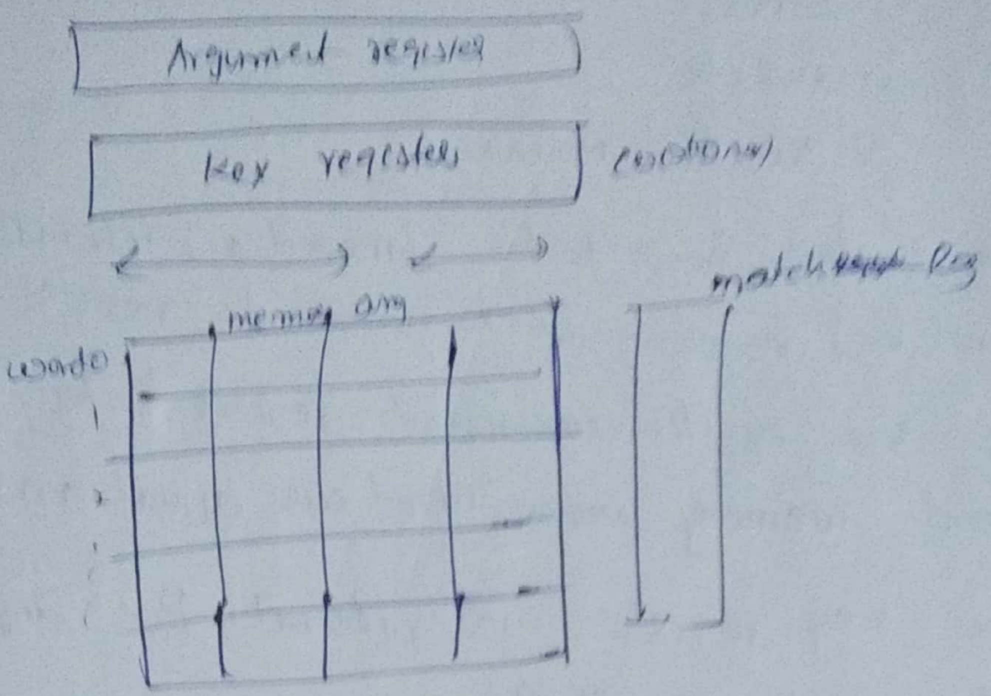
→ These are sequentially allesed device

→ These are used for Bulk storage of data at a cheaper price.

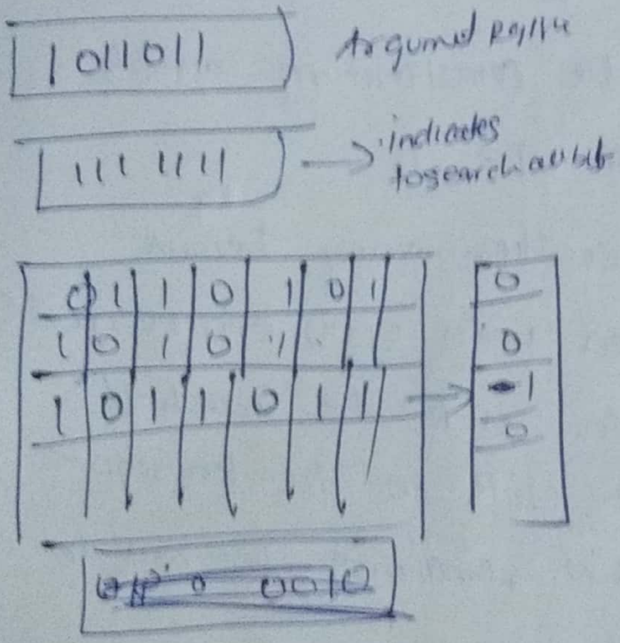
* Associative Memory *

→ It is also known as content addressed memory:

Leusc pm

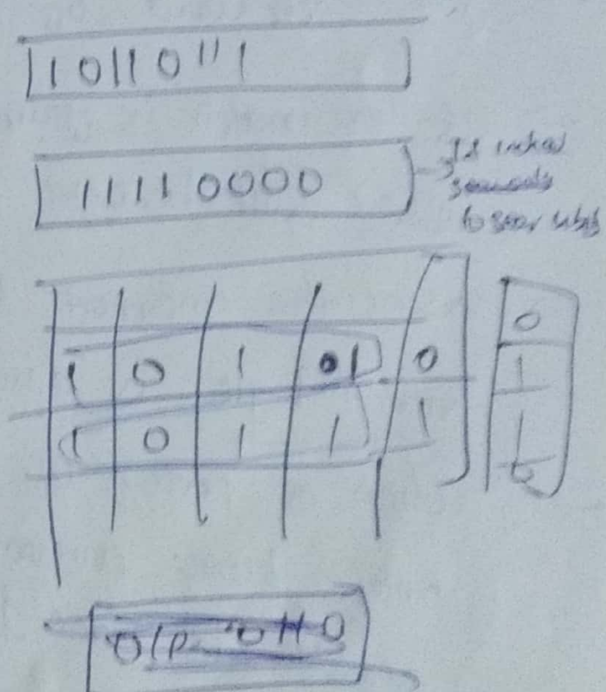


ex1: working / significance of key register



O/P 1011011

ex2 significance of key reg



OP 101010
1011 1

→ Here we are ignoring remaining 4 bits.

→ Associative memory is used

1) insert
2) Delete
3) search operation

→ The content need to be inserted, or deleted or search will be inserted in to argument register

→ The size of the argument register, key register and memory word are equal in size

→ The key register is optional, Bx default it consists of all 1's

→ match register consists of single bit of storage for each word of memory of memory array

→ If a match is found the corresponding match bit will be "one" ~~here~~ else '0'

→ associative memory is costlier memory because match logic is associated with each and every cell

→ when a portion of data is to be searched only those corresponding bits in the key register should be made 1 and remaining bits will be '0's

→ The content of argument register will be searched parallely among all the memory locations in the memory array.

Locality of reference

Temporal reference

→ loop execution

```
for (i=0; i<100; i++)
{
.
.
}
```

Spatial reference

→ program execution

→ If ith Instruction is executed by CPU, soon it requires (i+1)th Instruction

100 times of execution

1 time for main memory

99 time for the cache memory

Hit ratio = $\frac{\text{No of hits}}{\text{Total no of references}}$

Avg Memory Access time = H (Access cache) + (1−H) [Access time + main memory access time].

H = Hit ratio

(1−H) → miss ration

C → Cache

→ CPU generates the Mainmemory address with that address it visits the cache and founds that instruction then it will return to CPU, if same instruction is not found in cache, it visits mainmemory, while it is taking that instruction to CPU it drops a copy in to the cache memory for future reference.

Mapping

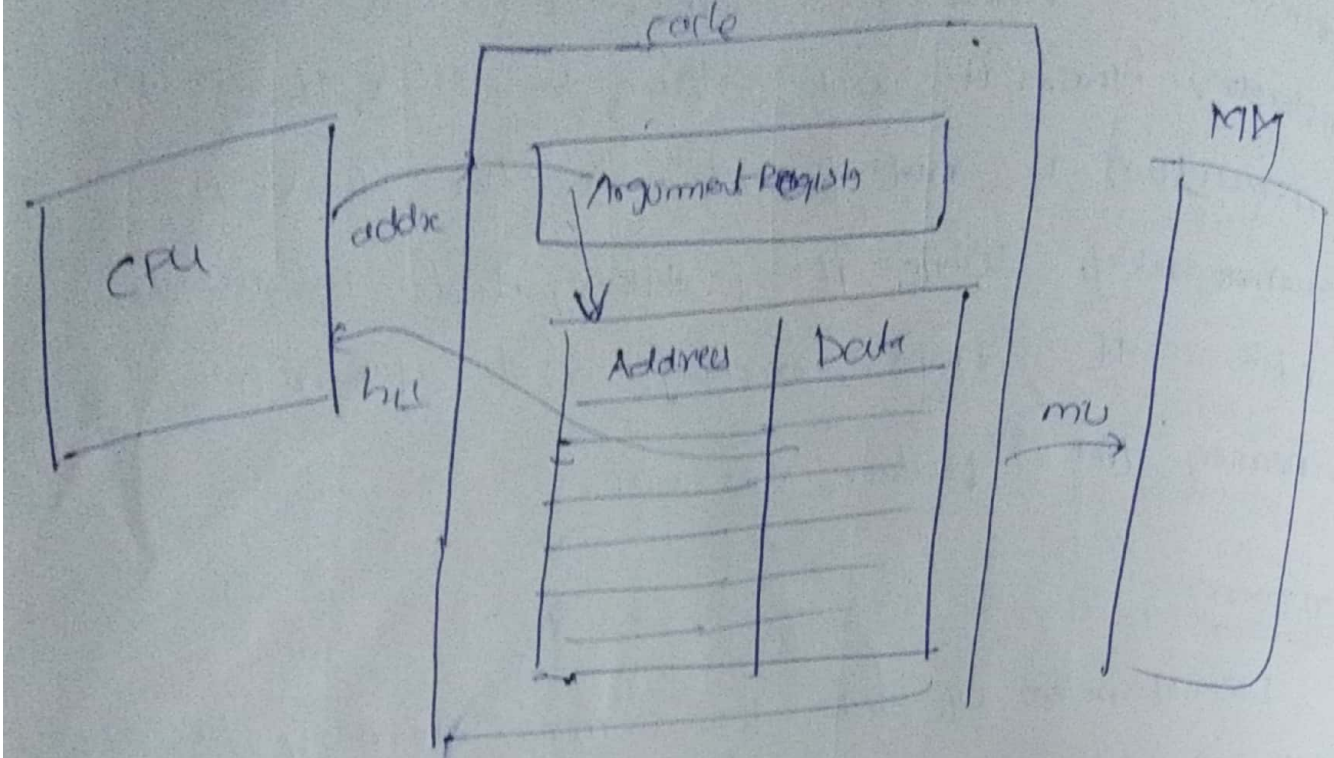
Transformation of data from main memory to cache is known as mapping

Types of mapping

1) Associative mapping
2. Direct mapping
3. Set-associative mapping

Associative mapping

In these mapping the cache memory is made up of associative memory.



The size of the argument Register is equal to size of the address part.

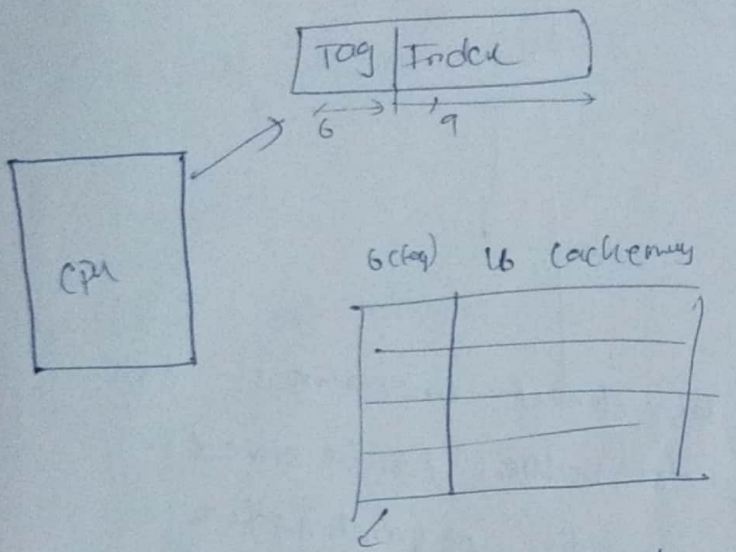
→ The associative memory array is made with address part and Data part.

→ CPU Generated address will be kept in to the argument Register, that argument Register content will be searched in the address part of memory array, If a match is found its corresponding data is return to to CPU, It termed as hit.

→ If no match is found it visits main memory and drops a copy of address and data in to the cache memory for future reference.

Direct mapping:

As the Associative memory is costlier (because match logic is associated with each and every cell), the designers though of implementing cache memory with same semiconductor material like main Memory

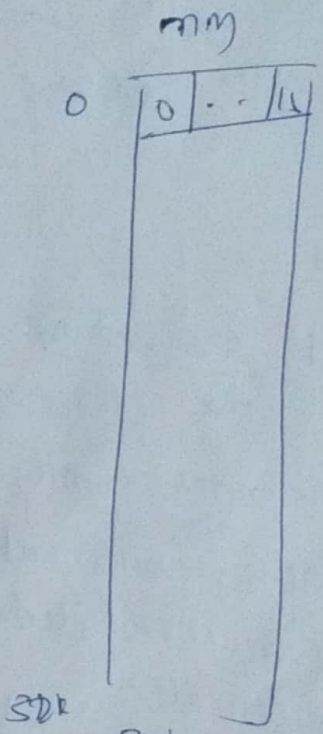


after extendy

cache

Cache Size = 512 x 22

Tag + 16



Address lng = 15

data = 16

<Tag, Data> = set

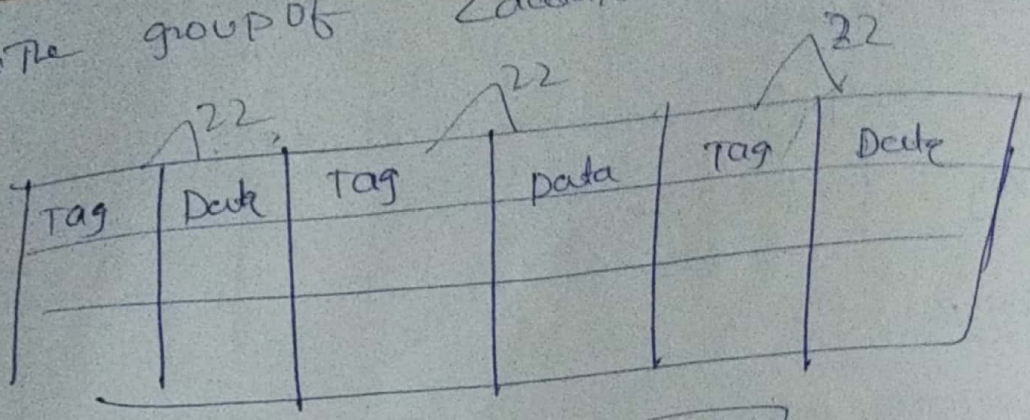
→ Constraint for direct mapping

for one index only one tag

Set Associative mapping

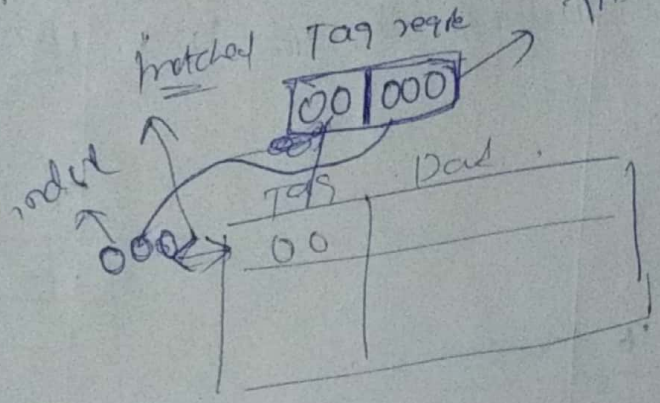
for one index we can hap many tag values

@ The group of <data, set>



Size of a cache = 512 × 66

eg:



Set Associative:

⇒ The disadvantage of direct mapping is for one index only one tag values can be stored in cache, costn that the probability of finding the reoccurred data in the cache memory will be difficult.

→ for that reason a set associate mapping technique came in to exutence. In these technique more than one set (<Tag, Data>) is accomodated in the cache.

Here for one index value → more than one tag value will be

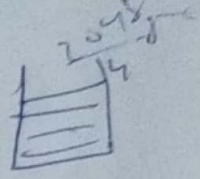
→ A Two way set associative cache memory uses block size of 4 words. The cache can accomodate a total of 2048 words from main memory. The main memory size '128K x 32"

a) How many bits are there in tag, index, block and word fields of the address format.

b) How many bits are there in each word of cache and how are their divided in to function

c) how many blocks can be cache accomodate

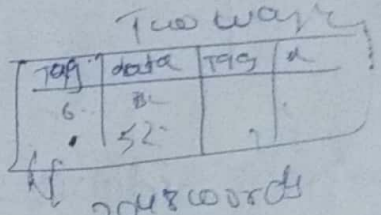
d) What is the size of cache memory



Ans 128K x 32

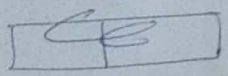
* 17 → ~~dat~~ Addresslines
* 32 → datalines

CPU generate 17 bit



= 2^{11}

= 11 words



~~In 17 words~~

3

In 17 bits 11 bits will be index

6 bits will be tag

a) The size of cache

(6+32+8)

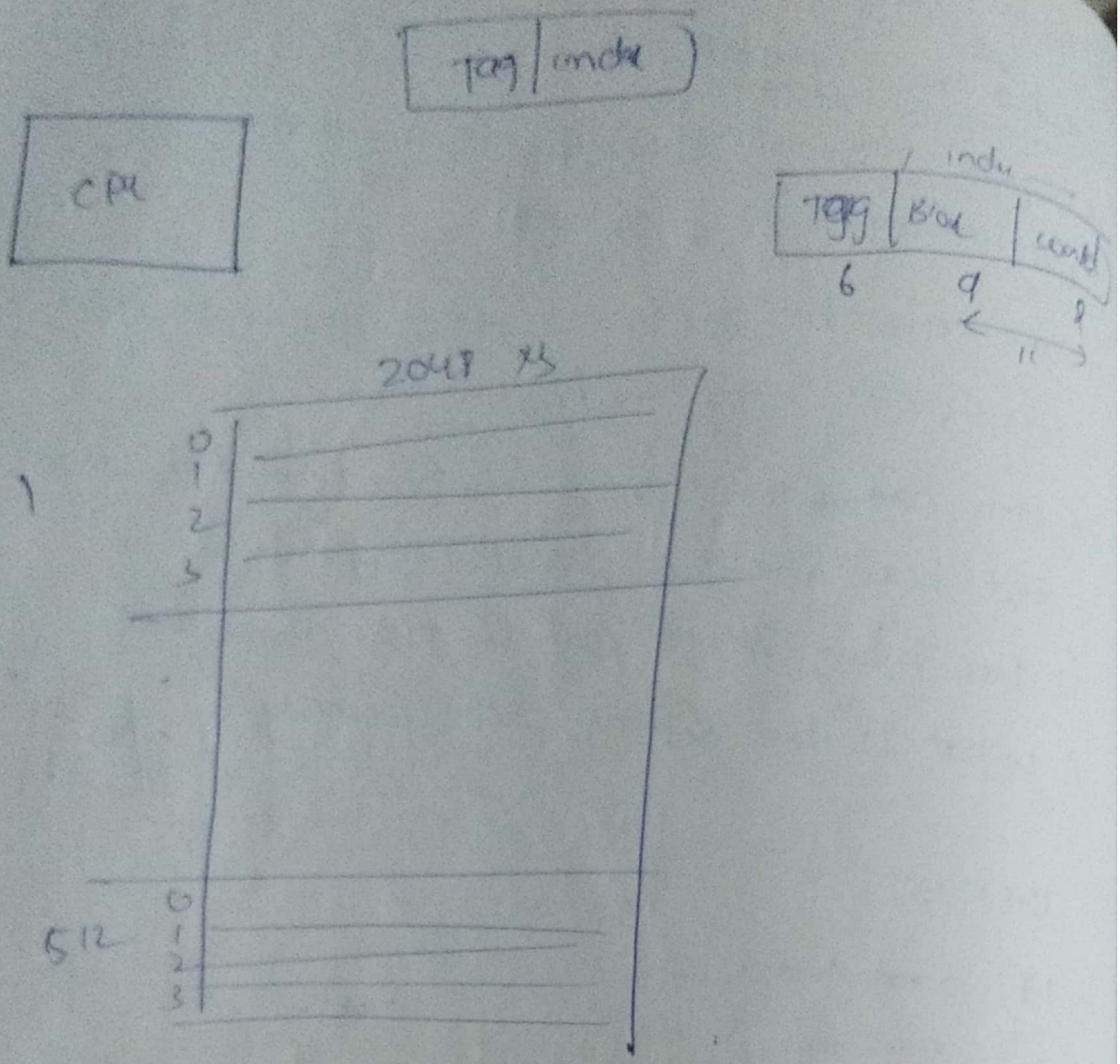
$\frac{1}{3}\frac{8}{2}$

$\frac{}{2\,6}$

= 2048 x 76

a) In tag → 6 bits, index = 11 bits

word field → 2



→ TO address ~~block~~ block = 512 = 2^9 = 9 bits

→ TO address word in block $4 < 2^2$ = 2 bits

~~No fact No~~

Total block $\frac{2^{11}}{2^2} = 2^9 = 512$

given size of block (4 words)

a) Tag = 6
Index = 11
Block = 9
word = 2

b) 76 bits

Tag	Data
6	32

c) $\frac{2^{11}}{2^2} = 2^9 = 512$ blocks

a) 2048×76

Q) A digital computer has a memory unit of 64K×16 and a cache memory of "1k words". The cache uses direct mapping with a block size of 4 words

a) How many bits are there

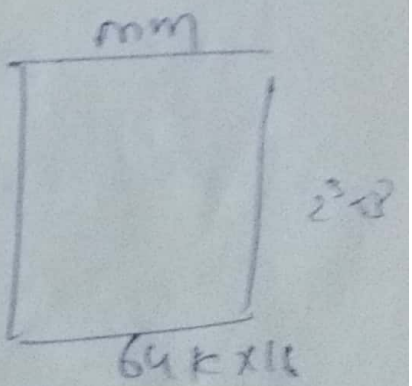
(i) Tag = 6
(ii) index = 10
(iii) block = 256
(iv) word fields of address format

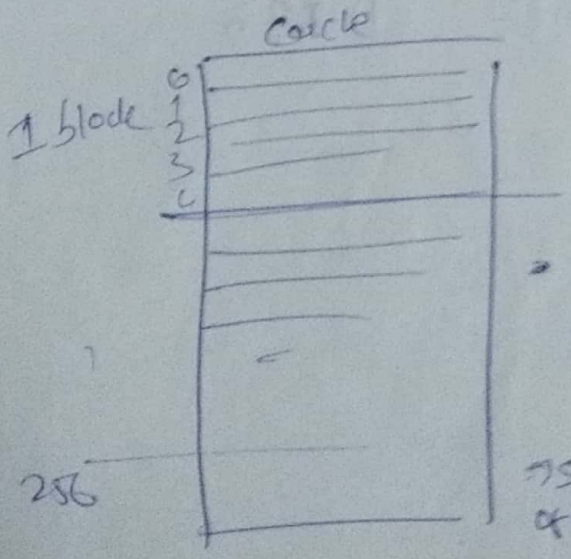
b) how many bits are there in each word of cache and how are they divided into fields? Include a valid bit.

c) how many block can be cache accommodate

d) What is the size of cache memory

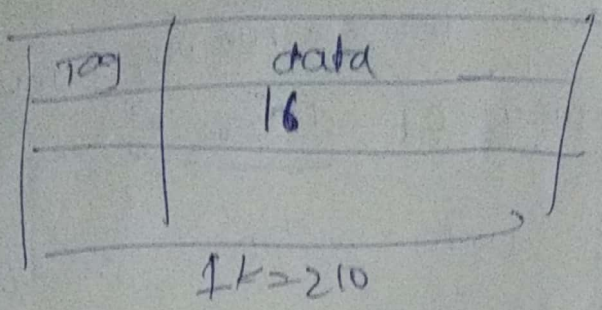
Q





- Size of cache = 1k words
  → $\frac{2^{10}}{2^2} = 2^8$ = 256 blocks
- 17 address
- 16 → data
- 7 size of block

direct mapping



CPU generated address = 16 bits

Internal Index = 10 bits

Tag = 6 bits

a) Tag = 6 bits

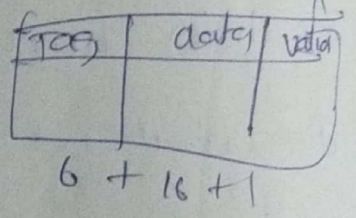
Index = 10 bits

block = 8 (256 = 2^8)

word = 2

b) bits = ~~Tag~~ Tag = 6

data = 16.



= 23 bits

c) No of block = 256

d) size of cache = 1K x 23

Virtual memory :

→ An extension to main memory which uses portions of secondary memory is known as

Address space	memory space
→ logical address	→ physical address
→ Secondary memory	→ Main Memory
→ pages	→ blocks, frames

→ Total secondary memory can be virtual but all of should be in DUDGED

→ Address space → The address generated by programmer is known as logical address. Collection of such logical addresses is known as address space

→ memory space.

CPU generates physical address collection of those addressed is known as memory space.

Question

An address space is specified by 24 bits and corresponding memory space by 16 bits.

a) No many words are there in address space

b) No how many words are there in memory space

c) If a page of 2K words, how many pages and blocks are there in system

a) $2^{24} = 16MB$

b) $2^{11} = 64KB$

c) No of pages = $\frac{16MB}{2K}$

No of pages = No frames = No of blocks

$= 8K$

$= 2^{13}$

b) No of blocks = $\frac{64KB}{2K}$ <-- 32 blocks

Page Replacement Algorithms

* The Two most commonly use page replacement Algorithms
  1) fifo
  2) LRU

fifo

→ The fifo algorithm selects for replacement the page that has been in the memory the longest time

→ LRU : If based on the assumption that Least Recently Used page is a better choice for removal than the least recently Loaded pages as in fifo

6) A virtual memory system has an address space of 8K words, a memory space of 4K words and page and block size of 1K words. The following page reference changes occur during a given time interval

4, 2, 0, 1, 2, 6, 1, 4, 0, 1, 0, 2, 3, 5, 7

Determine the no of page faults and the pages that were resident in mm after each page reference change if the replacement algorithm used is ~~fifo~~ a) fifo

b) LRU

[pagefault: if a reffered page is found in cache it is termed as a page hit
→ if a required page is not found in the cache then it is termed as pagefault]

A) 4 2 0 1 2 6 1 4 0 1 0 2 3 5 7
• • • • hit • hit • hit hit hit • • • •

fifo:

4	6	5
2	4	7
0	2	
1	3	

pagefaults = 10
pagehits = 5

= given memory space = 4K words ✓

Size of each frame = 1K word

No of frames = 4K word / 1K word = 4 frames

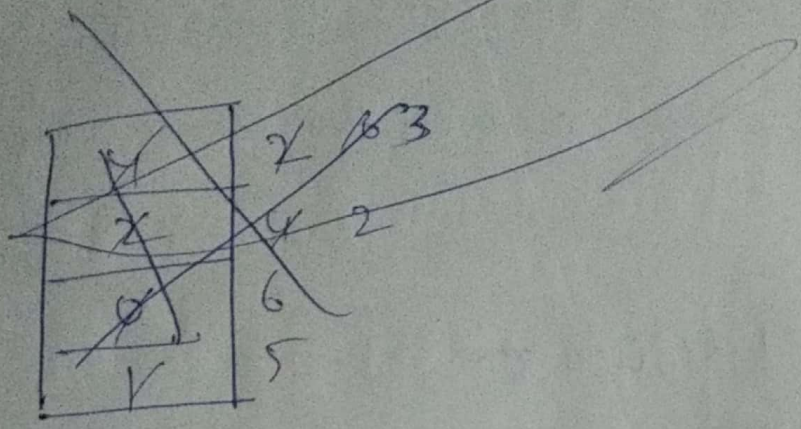
some of book starting 4, 2, 0, 1 will not considered has pagefaults

so

we can say pagefaults = 6

e/o by depending on options

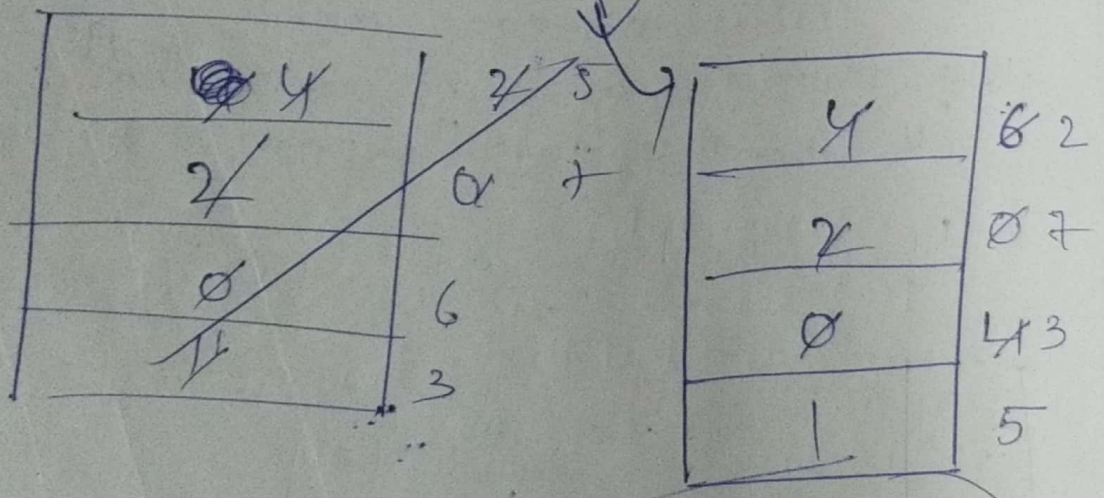
4 2 0 1 2 6 1 4 0 1 0 2 3 5 7



LRU

4 2 0 1 2 6 1 4 0 1 0 2

3 5 7



LRU:
It will replace which is not used for longest period of time

MRU
It will replace which is used just now

4 2 0 1 2 6 1 4 0 102

4	
2	6
0	5
1	2

optimal

4 2 0 1 2 6 140 102 357

4	7
2	6
0	3
1	5

9 → pagefaults

6 → pagehits

for these we can't say ~~exact~~ how many will recide in memory finally

Flynn's classification

→ M.J flynn classified computers on the earth into four categories based upon their instruction fetch and instruction execution.

(i) SISD [Single instruction stream, Single Data stream] → uniprocessor

(ii) SIMD (Single instruction stream multiple data stream]

(iii) MISD (Multiple instruction Stream Single data Stream) (does not exist)

(iv) MIMD (multiple instruction stream multiple data stream) → multiprocessor

Pipelining :

→ Decomposing a sequential process into sub operations where each sub operation is given dedicated segment such that all segments work concurrently.

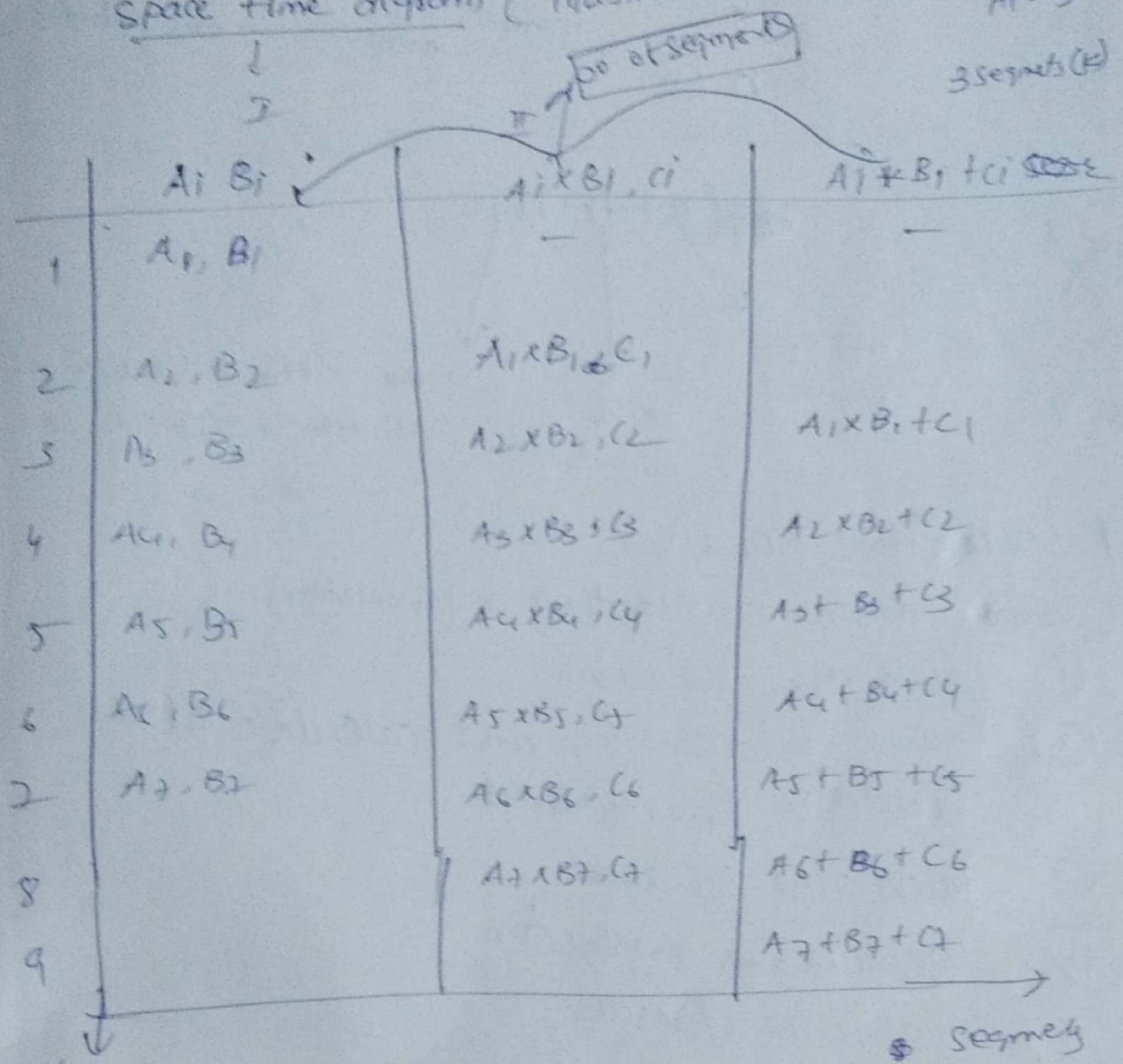
→ pipelining can be applied on 2 concepts

1) ALU operations (data streams)
2) Instruction execution (Instruction Stream)

Arithmetic Pipelining :

$(A_i + B_i) + C_i$, $i = 1$ to 7 (execute 7 times)

Space time diagram (illustrates behaviour of Arithmetic pipeline)



without pipelining

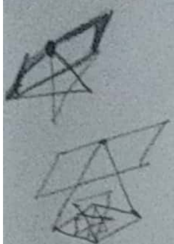
1. A_i, B_i
2. $A_i \times B_i, C_i$
3. $A_i \times B_i + C_i$

$7 \times 3 = 21$ → clock pulse

speed up ratio = [to comp[...] pipelined system]

$$\frac{n\,t_n}{(k+n-1)\,t_p}$$

- n → no of instructions (i = 0(7))
- t_n → Time Taken to execute an instruction in non-piplined sys[...]
- k → no of segments
- n → no of instruction
- t_p → Time taken to execute an instruction in pipeline system



INSTRUCTION PIPELINE :-

steps involved in executing a instruction

→

1. fetch the instruction
2. Decode the instruction and calculate effective address if required
3. fetch the operand
4. execute the instruction
5. store the Result.

~~Pipline Com~~	FI_i	$D\&EA_i$	FO_i	$E\&SR_i$
1	FI_1	—	—	—
2	FI_2	D&EA	—	—
3	FI_3	$D\&EA_2$	FO_1	
4	FI_4	$D\&EA_3$	FO_2	E&SR

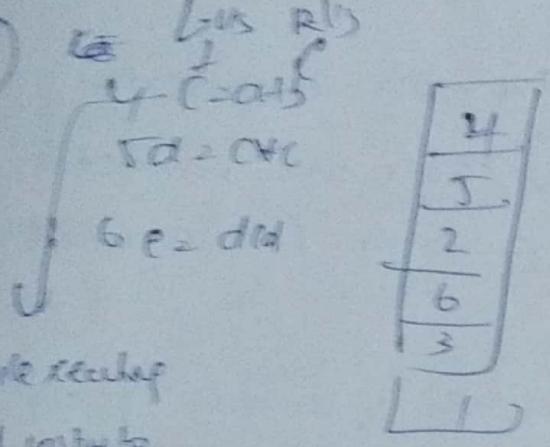
[illegible] pipelining starts at 4th clock cycle = no of segments

∴ optimisation of pipelining ate no of segments [illegible]

pipeline Conflicts / Difficulties in implementing instruction pipeline

Units for instruction pipelining

1. Resource Conflicts (BUS)
2. Data Dependencies ( )
3. Branching Difficulties ( ) ——— addr



while reading 4 instructs

The value of C will not be 9 it will be 6 (because already 5 instructions)

Resource Conflict

(1) when the bus is been used to perform instruction fetch and operand fetch in the same clock cycle, a conflict arises in allocating Bus to perform which operation

Data dependencies

(1) When a compiler finds the symbol which appering on LHS also appears immediately ~~B~~ on RHS a conflict arises such is known as data dependency

Solutions to overcome data dependency

1) Hardware interlocks ✓
2) Operand forwarding ✓
3) Delayed load ✓

3) Branching difficulties :-

→ As the pipelining concept execute the instruction sequentially, there are instructions which will make the program to run out of sequence

```
e.g. 4. add();
      5
      6
     100 add()
         {
         }
     }
```

basically if we apply pipeline it will execute 5 instructions but we need 100 to 105 instructions

* RISC Pipelining (Reduced instruction set computer)

→ These processor will implement the instruction pipeline in 3 segments

① I : instruction fetch
A : ALU operation
E : execute the instruction

Vector processing :

Solutions for branching difficulty :

1) prefetch target instruction
2) Branch target Buffer
3) Branch prediction
4) delayed Branch.

Vector Processing :

→ A High dimension vector calculation will be done on high end machine in very less amount of time. that are beyond the capability of a conventional computer

→ These problems are characterized by the fact that these required a vast number of computations that will take a conventional computer days or weeks to complete

→ In many and engineering applications the problems can be formulated in terms of vectors and matrices that lend themselves to vector processing

applications

1) long Range whether forecasting
2) petroleum explorations
3) Seismic data analysis
4) medical diagnosis
5) Aerodynamics & spaceflight Simulations
6) Artificial intelligence & expert System.
7) Image processing

Vector Operations

1) Loop operations
2) matrix multiplication
3) memory interleaving
4) Superscalar processors
5) Super Computers

Addressing modes

ways to locate an operand

types

1) Immediate mode:

The operand is stored along with the instruction itself.

Ex: MOV R0, #45H

2) direct mode.

It is an address of an operand is specified with in the instruction

MOV R0, 45H

3) indirect mode:

MOV R0, @R1

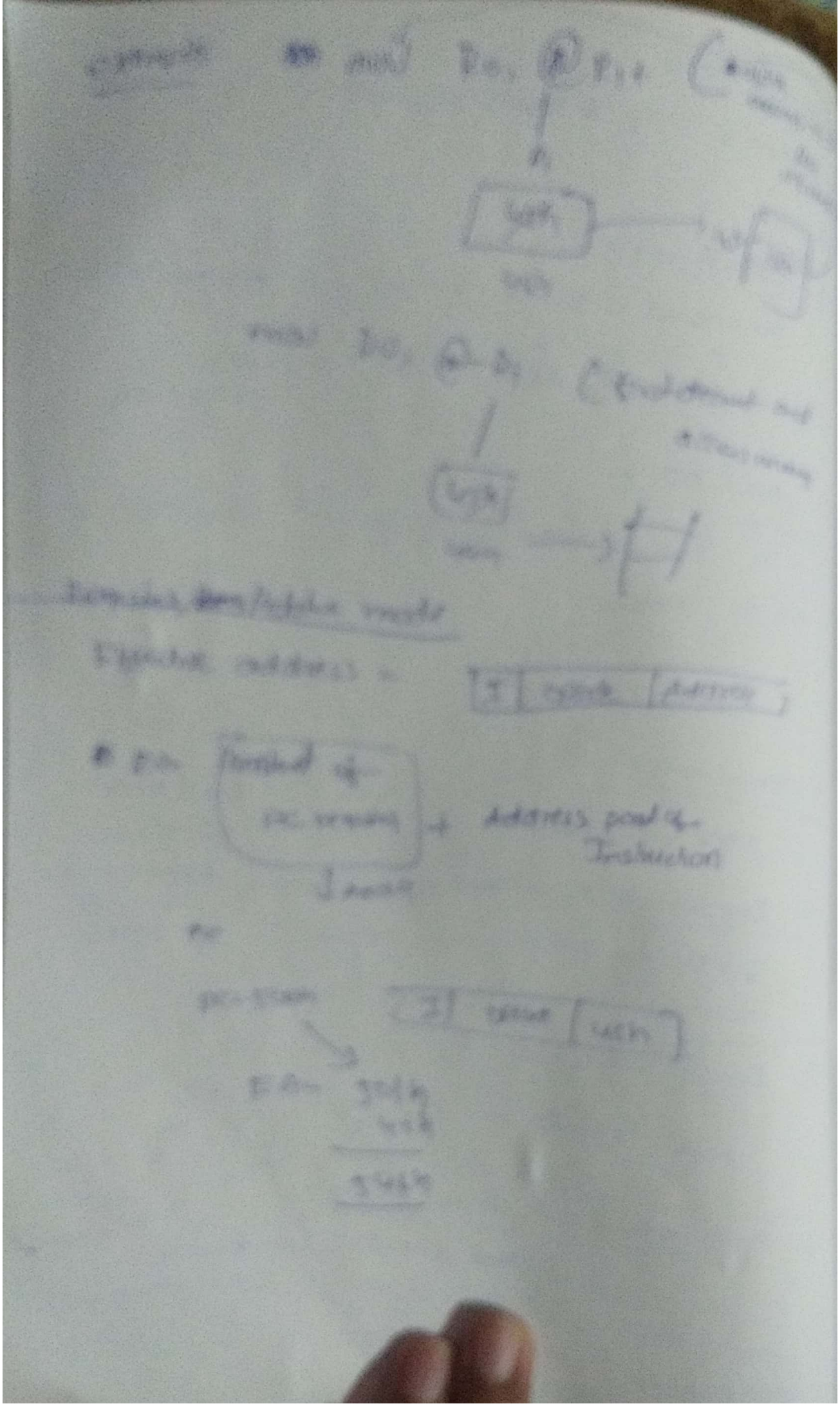
4) Register:

5) Register indirect mode:

MOV R0, @R1

6) Auto indirect / Auto decrement mode:

It is similar register indirect mode where the content of register is auto incremented/ decremented to the next address



Indexed mode :

EA = Content of Index Register + Address part of instruction

→ Instead of PC ~~base~~ register an index register is used in calculation of Effective address

Eg:

Index Register = 500h [I | opcode | 45h]

EA = 500h
   + 45h

   545h

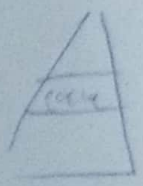
Some of the arrays

5) The memory which is used to store the copy of data (or) instructions stored in Larger memory (outside the CPU) called 'Level1 cache"

6) The larger memory placed between the primary cache and memory is called ____

a) Level1 cache (primary cache)
b) Level2 cache (secondary cache (or) external cache)
c) EEPROM
d) TLB

7) The next level of memory hierarchy after L2 cache is ____



a) SM
b) TLB
c) MM
d) Register

(Next Level means above Level)

Level1: It is a memory which is is inside processor (or) CPU
→ It is implemented with the use of SRAM
→ SRAM

Level2: It is outside processor and in between the primary memory and CPU

level3 cache manages the both L1 and L2

fastness of of

L1 > L2 > L3

on cache memory

The Reason for the implementation of the cache memory is ____

a) To increase the internal memory of the system
b) The difference in speeds of operation of the processor and memory
c) To reduce the memory access and execute
d) All of the mentioned

The effectiveness of the cache memory is based on the property of ____

a) Locality of reference
b) Memory Localisation
c) memory size
d) None of the mentioned

Locality of reference when there are more operands

The temporal aspect of the locality of reference means ____

a) The recently executed instructions will be executed soon again

4. The spatial aspect of the locality of reference means ——

Ans That the instruction in close proximity [closeness] of the instruction executed will be executed in future

explanation.

→ nearby instruction is more likely to be executed in future

5) The correspondance between the main memory blocks and those in the cache is given by mapping function

7) The write-through procedure is used

TO write directly on the memory and the cache simultaneously.

8) Dirty bit is used to signify that the cache location is updated.

9) copy-back protocol is used

a) TO copy the contents of the memory on to the cache

b) TO update the contents of the memory from cache

4. The spatial aspect of the locality of reference means ——

Ans That the instruction in close proximity [closeness] of the instruction executed will be executed in future

explanation,

→ nearby instruction is more likely to be executed in future

5) The correspondance between the main memory blocks and those in the cache is given by mapping function

7) The write through procedure is used

TO write directly on the memory and the cache simultaneously.

8) Dirty bit is used to signify that the cache location is updated

9) copy-back protocol is used

a) TO copy the contents of the memory on to the cache

b) TO update the contents of the memory from cache

C) To remove the content of cache and push it on to memory

d) none

explanation:

This is another way of performing the write operation, where in cache is updated first and then the memory.

b) The approach where the memory contents are transfered directly to the processor from memory is called Early-start

---

which memory is difficult to interface with processor?

A) static memory

B) Dynamic memory

C) ROM

D) none

explanation:

Because it requires periodical refreshing

The minimum time delay required between initiation of "two successive memory operation is called memory cycle time

for a memory system the cycle time is is same as

Longer than access time

Cycle time = Latency time + transfer time

Latency time = Latency time is overhead of finding the right memory location and preparing to access it

Transmission time : Time required to transfer the data

8) The Refreshing rate of dynamic RAMs is approximately once in 2 millisecond and dynamic Ram has higher bit density

9)

11) The access time of magnetic bubble memory is approximately

A) 30 nanosecond
B) 30 micro seconds
C) 30 PICO second
D) 0.5second

12) The use of "hardware in Memory management through "segment relocation" and protection is

Sol → To perform address translation to reduce size of memory

→ To " " " " execution

memory refreshing may be done

A. By the CPU that contains a special refresher counter, only

B. self (by external refresher)

C. CPU registers, cache, main memory are external memory of the system

D. Any advantage of memory interfacing is that

A) Large memory is obtained

B) Effective speed of the memory is increased

C) The cost of the memory is reduced

D) usable memory is obtained

21) which of the following need extra hardware for DRAM refreshing?

A) PCI5

B) motorola - 68000

C)

NOTE

→ property of Locality of reference may fail.

It a program has

A. Many conditional jumps

B. many unconditional jump

C. many operands

D. many operators

24. what is the correct sequence of the delays that happen during a data transfer from a disk to memory

Sol: Seek time → latency time → transfer time

If is the time required for read/write head to move from one track to other

Access time

The time a program or device takes to locate a single piece of information and make it available to computer for processing

the time it takes to position the proper sector [In secchor tracks will be there] under read/write head

25) How many RAM chips of size (256K x 1bit) are required to build 1M Bxte memory?

A. 8
B. 32
C. 24
D. 32

256K = 2^{18}

$2^{18} \times (1) \times x = 2^{10} \times 8$

↓

In order to equal to Bite

$x = (8) \times (4) \rightarrow$ 32 chip

↓

It is equal to 8 bit (i.e data line)

So 32 chips are require

Latency time

the time it takes to position the proper sector [In sector tracks will be there] under read/write head

25) How many RAM chips of size (256K x 1bit) are required to build 1M Byte memory?

A. 8
B. 32
C. 24
D. 32

256K = 2^{18}

$2^{18} \times (1) \times x = 2^{10} \times 8$

↓

In order to equal to Byte

$x = (8) \times (4)$ → 32 chip

If u equal to 8 bit (i.e data line)

So 32 chips are required

Four memory chips of 16K size have their address buses connected together. The system will be of size 16K×16

How many address are required for 25×40 order

Ans 1000 [25×40=1000]

A flip flop can be converted to T flip flop by additional logic circuit

D = T⊕Q

Ans D = T·Q̄ + T̄·Q

D = T⊕Q

→ If main memory of 4K bytes and the cache is 2K words. If uses associative mapping then each word of cache shall be

Ans 2*8 = 16 bit

... register & multiple register of a hardware implementing booth's algorithm have (111011) & 110...

... Slalt he

(1C) 10

---

memory unit is accessed by contents called

Associative memory

Aging register are counter which indicate how long these associate page have been refered

48, 52, 53, 58, 59, 64

The address of page table is pointed by

· page table base register "

The performance of cache measured in terms of hit ratio

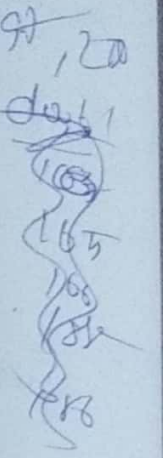
→ A k bit field can specify 2^k register

Translation from symbolic to Binary is done in —

2 phase

→ 159, 160, 173, 176, 181, 184, 186, 191, 198, 199, 200

→ fragmentation dividing the MM is to equal size fragments



→ dedicated computer means which assigns the one and only one task

→ interrupts initiated through instruction called software interrupts

→ Memory access in RISC architecture is limited to instructions

STA and LDA

→ CPU doesn't perform the operation

A) Data transfer
B) Logic operation
C) Arithmetic operation
D) All of above

| As CPU contains ALU, it will perform arithmetic and logic