COA

4/4/18

Set Associative :- = 2 tag set associative

	Tag	Data	Tag	Data
000				
	00		02	
FFF				

Suppose addressing = 00 001 Matched / Hit
set associative acting. 02 001 Hit
00 001 Hit.
01 001 Miss condition and replacement is done in Tag 02.

No. of replacement decreases

Block wise search

searching is like set associative but mapping is direct mapping

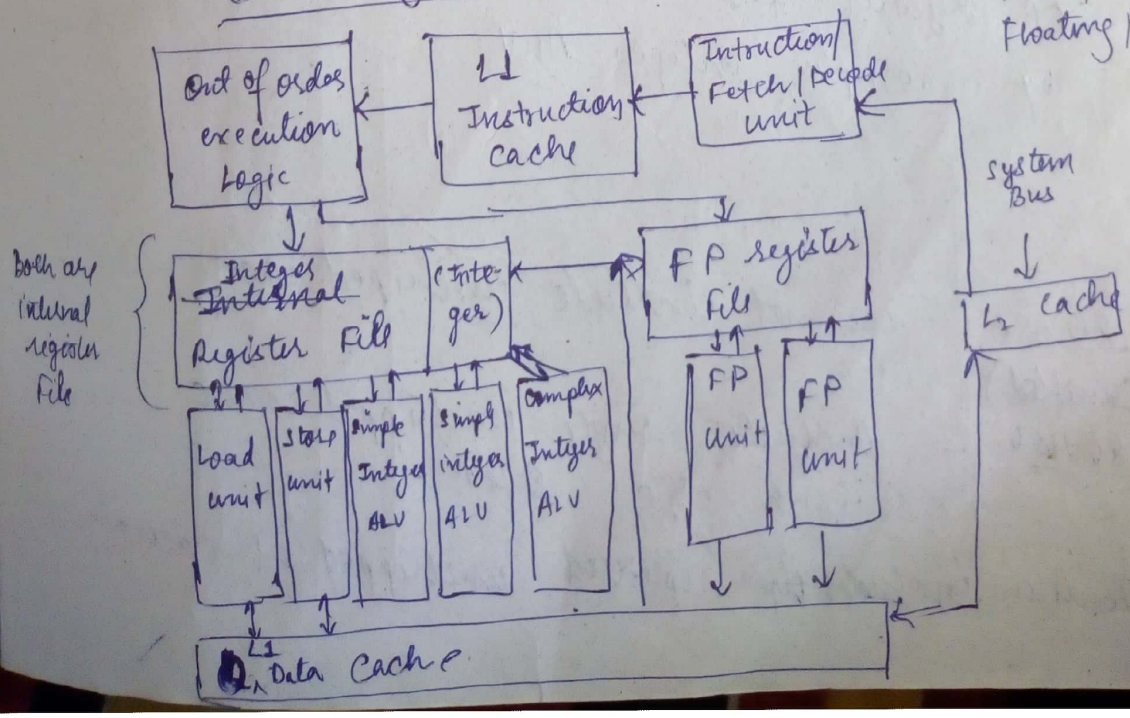
Hit/Miss case or bits of tag / block etc

Exams.

5/04/18

Cache organisation for pentium-4

FP = Floating point



System Bus fetches word from the main memory.

L1 = splitted cache
L2 = unified cache

Out of order Execution logic :- some inst.

when dependency b/w inst/data comes then the related inst is executed → sequencial inst.

The execution sometimes cannot enter to its execution unit. during reading of which inst will enter is decided by this. will be executed first to L2 cache or,

Integer file stores intermediate value calculated by integer + complex integer ALU

FP Register file stores intermediate value calculated by complex integer ALU.

80386 - does not include unchiped cache.

(unified)
80486 - include single unchipped cache with size 8-12 K

Pentium include splitted unchipped L1 cache.

pentium 4 includes L2 (unified) and splited L1 cache.

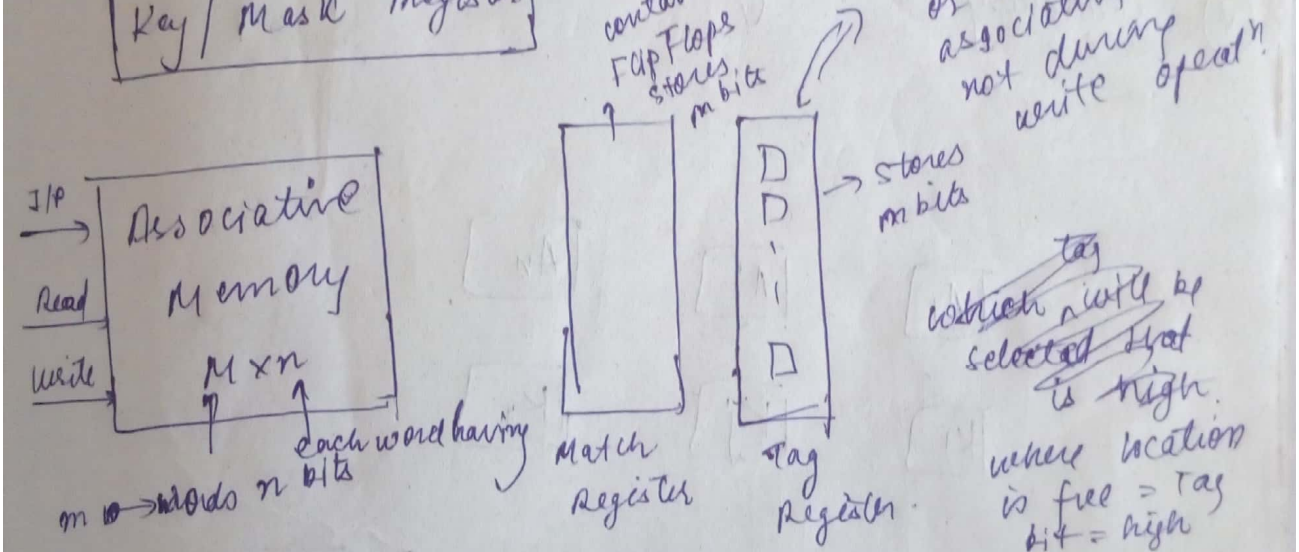
L2 = 8 way set associative memory and size is 256KB

Associative memory: also called → parallel search memory, content addressable memory, multiple access memory.

Addressed by the content itself rather than

Argument Register

Key / Mask Register



To check if is free location or not in associative or not during write operation.

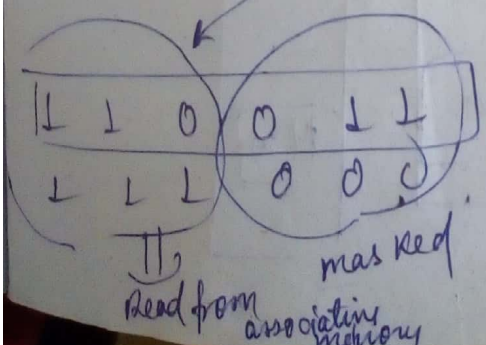
tag which will be selected that is high.

where location is free = tag bit = high

Data placed in Argument Register

Read Operation :-

sometimes not the whole word is required.



if all is read i.e

Mask Register assigns

1 1 0 0 1 1
1 1 1 1 1 1

all are read from associative memory

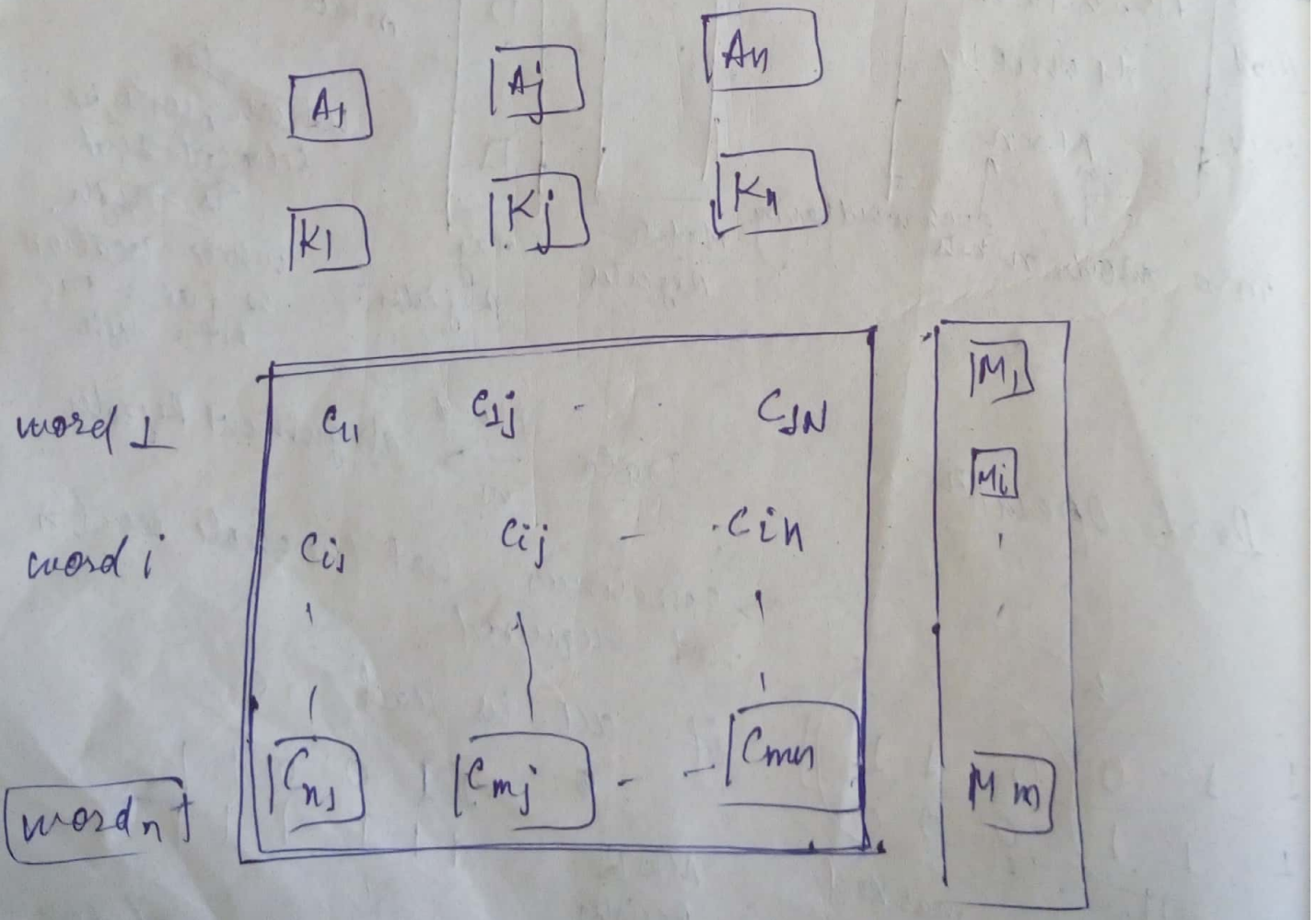
If bit wise there is a match b/w argument register data & associative memory ∘ found
else not found / no read operation.

b/w argument & associating data

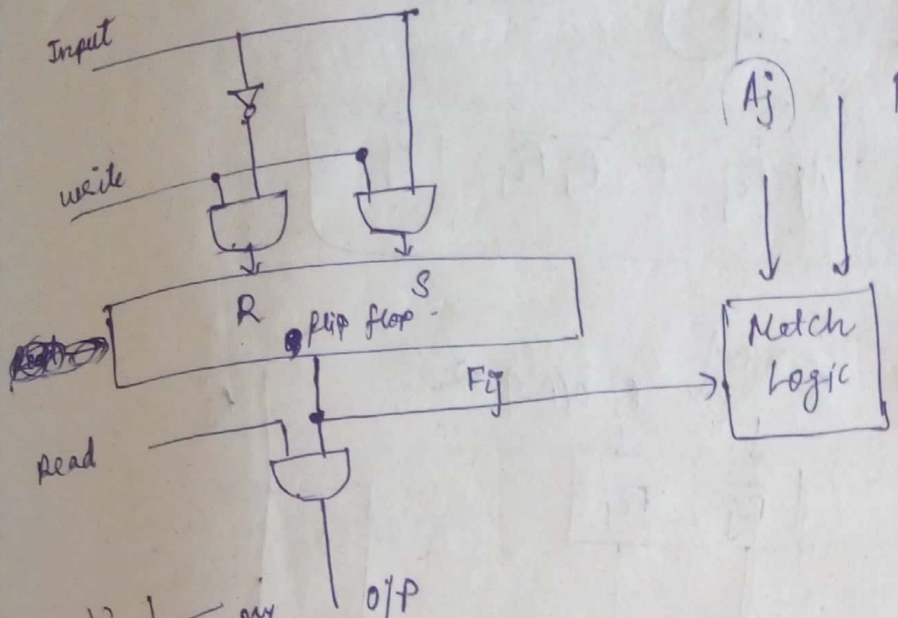
if a matching is 1 or more then its corresponding match bit is high

If multiple match = parallel access / all read. multiple location is access

(*) searching is very fast



Internal organisation of a cell



Mi = 1 then Match

Mi = 0 no match.

S	R	Memory
0	0	1
0	1	1
1	0	
1	1	Flip

Read = 1 = O/P is recieved

Read = 0 — O/P is not recieved

Input Taken written in S ∴ stored in FF

write = 1

~~0~~	~~0~~	~~1~~
~~0~~	~~1~~	~~0~~
~~1~~	~~0~~	~~1~~
1	1	0

Not including Key bit

(*) XNOR is used for matching

$x_i = A_j F_{ij} + \overline{A_j}\ \overline{F_{ij}}$ (For 1 bit)

$M_i = x_1$ and x_2 and x_3 - - and x_i

⇓

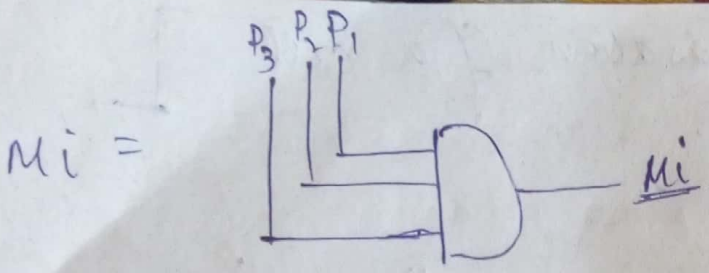
For all bit matching.

Including Key bit:- $K_j = 1$ then comparision is done. x_j = obtained $x_j = A_j F_{ij} + \overline{A_j}\ \overline{F_{ij}}$

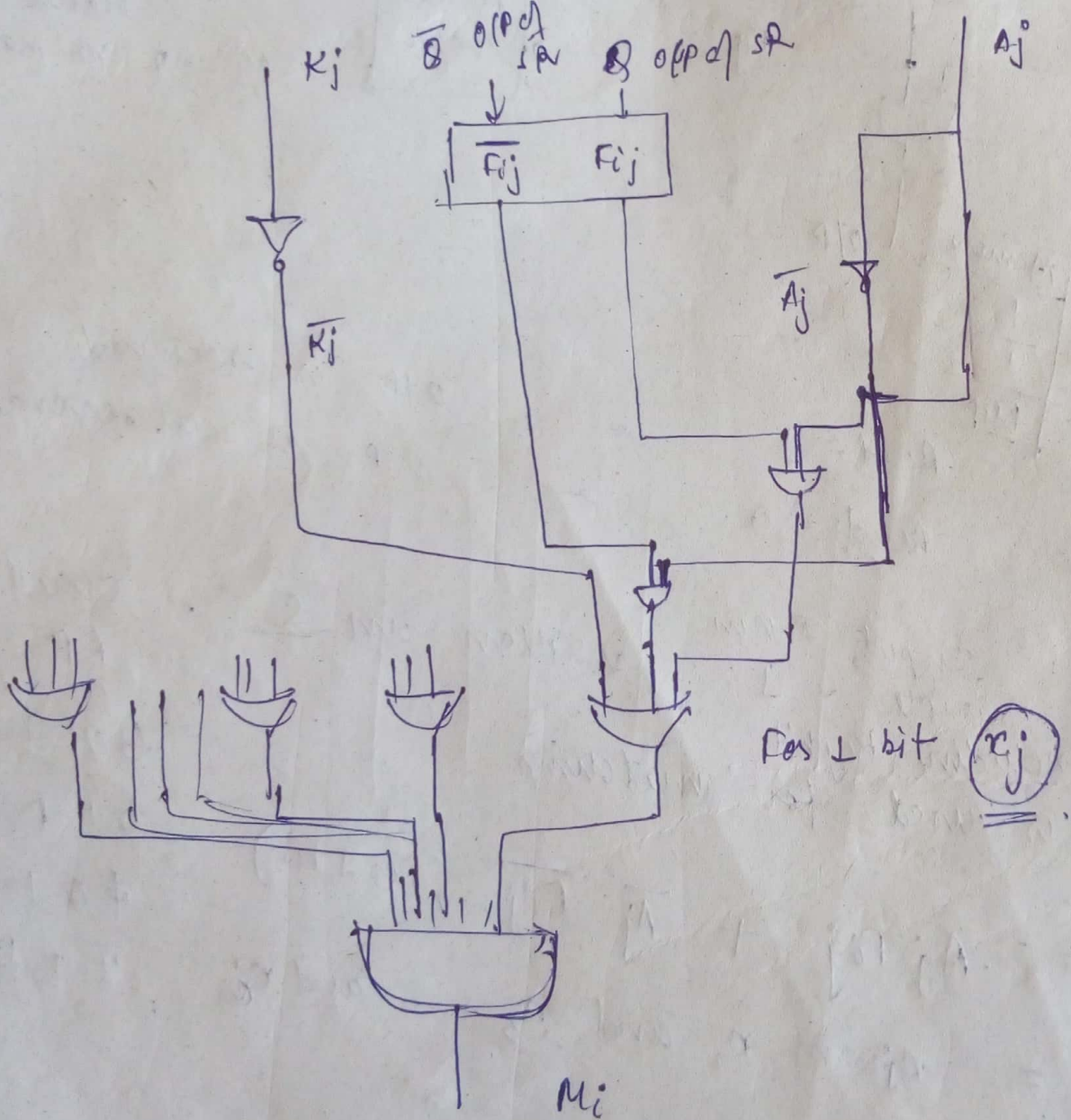
$K_j = 0$ then expression

$x_j = x_j + \overline{K_j}$

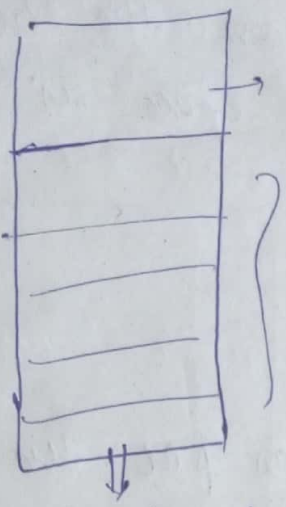
(go through Morris Mano)



(f) $\overline{x_j} = A_j F_{ij} + \overline{A_j}\,\overline{F_{ij}} + \overline{K_j}$



6/4/18



OS stored in secondary storage

(i) This part is divided into diff parts called partition during multiple

(ii) OS secondary storage to main memory

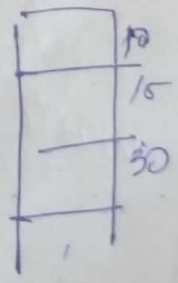
swap out (main memory to secondary storage)

swap in (secondary to main)

2 types of partitioning :-

(i) Fixed :- may be equal size partition
unequal size partition.

(ii) Variable :-
⇓
different size of memory storage

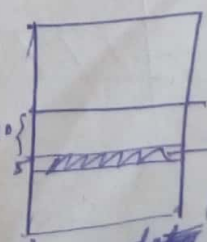


Disadvantage

~~more~~ more than 10K

less than 10K

equal 10K, :- If a process of size 10K and is present it OS do not fits it to the next higher memory location



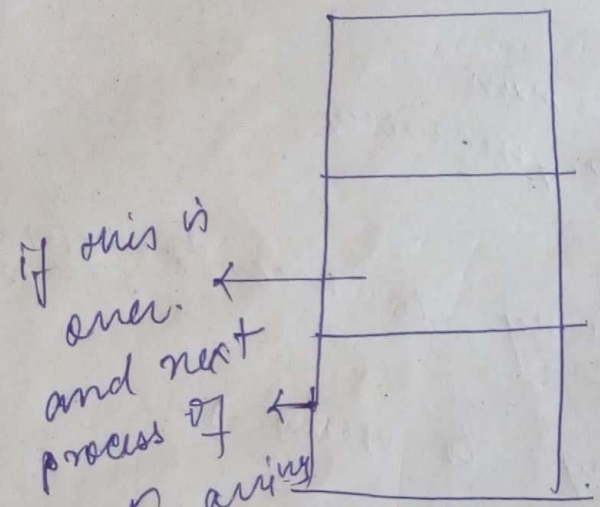
disadvantage of memory

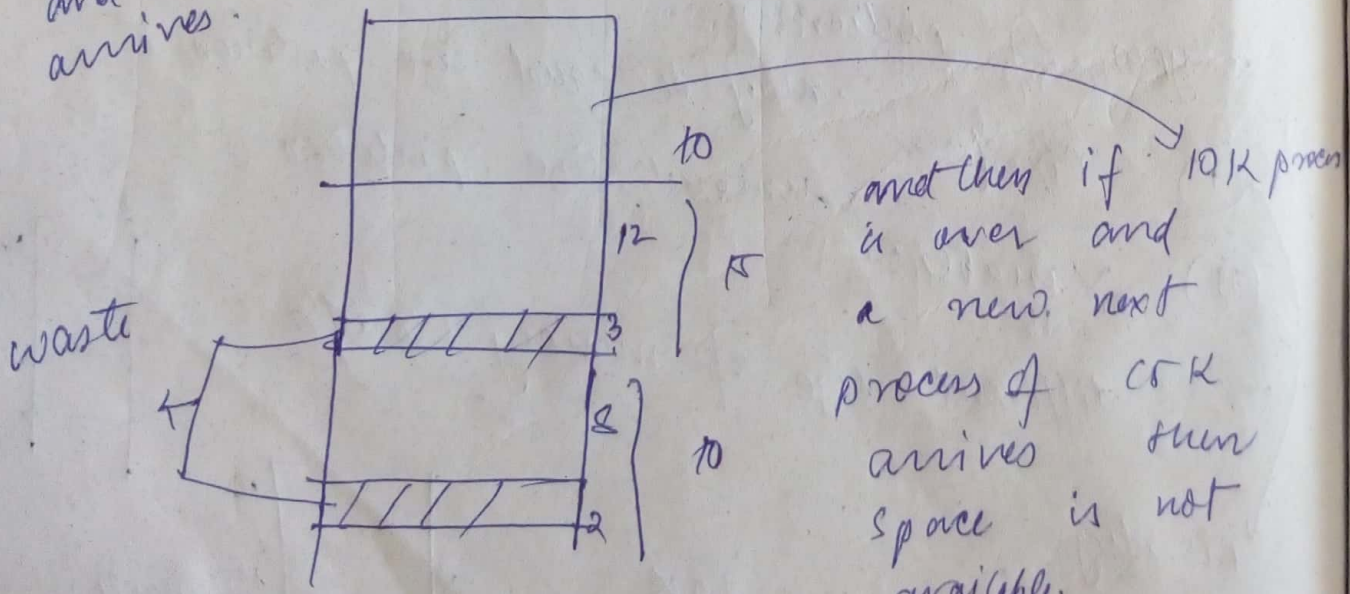
Fixed equal

In equal size partitioning all are equal, and if all processes are more than the size of individual size of partition though space is available but process cannot fit.

variable:-

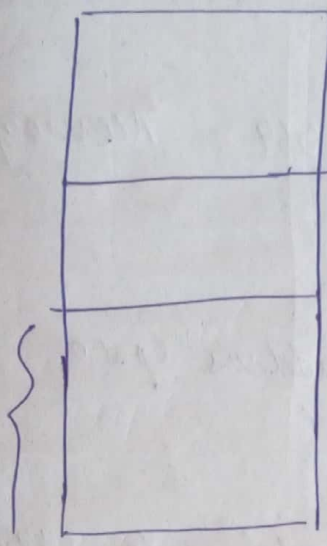
partitioning is done as per the size of the process





Compaction :- If shifts the present process and try to make space in compact manner.

∴ It manages space.

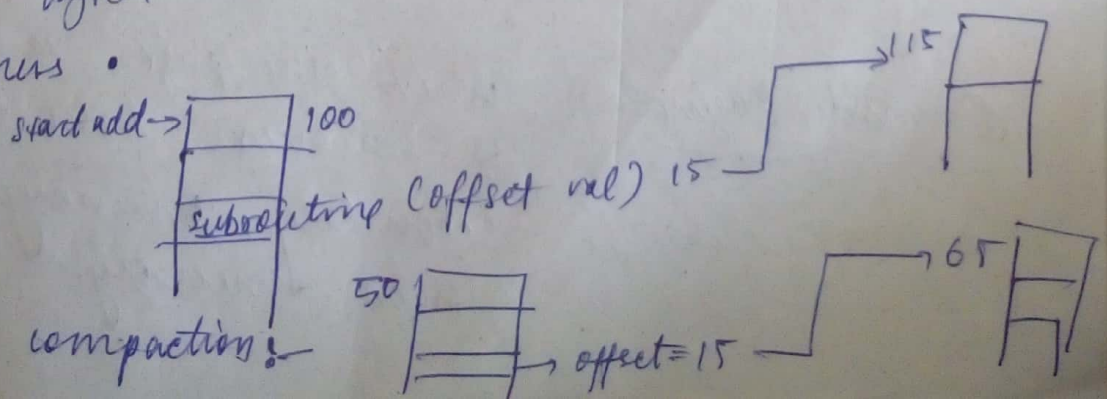


Relocation of the process is done and it happens during compaction to make spaces.

Disadvantages :-

① performance reduce. because OS takes times for shifting and this may reduce performance.

② During the shifting, if within the prog a subroutine comes, the subroutine will also get shifted, this may create mismatch of address. But OS takes care of it by logical address. not in physical address two same add in logical address.



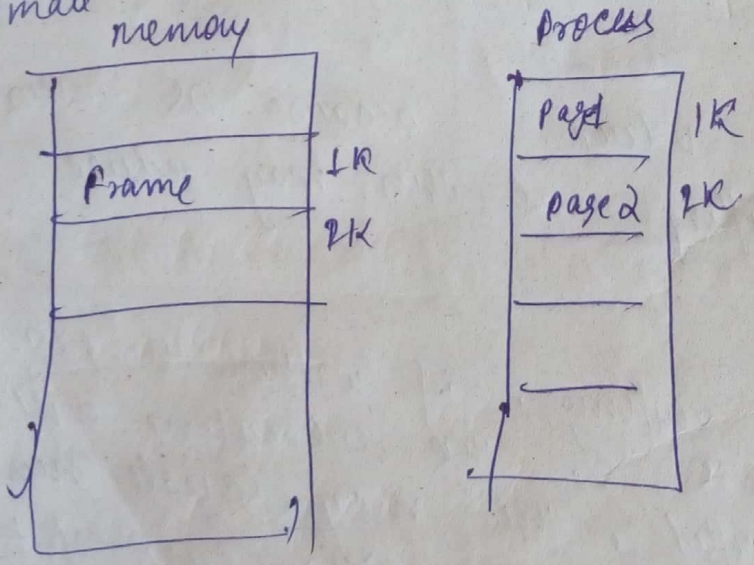
(iii) physical address is actual memory space available to the user (actual add to main memory)

(iv) logical address is stored with the help of secondary storage = start add + offset val.

(v) Total physical address available = memory space

Tot. Logic address available: address space

The available memory is divided into small chunks and the available process is also divided into small chunks same as memory.



Division of RAM

OS decides the size of chunks of process first ~~memory~~ and then it decides the size of memory

(vi) program is present in secondary memory

Demand Paging:- when the page is demanded then it moves from the secondary memory to the main memory.

All the disadvantages is avoided by virtual memory.

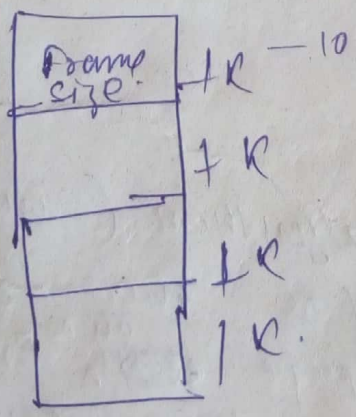
Virtual memory
① Takes care of secondary and ~~main~~ primary memory.
② ~~physical memory~~ address space is very small and of

∴ It uses logical address.

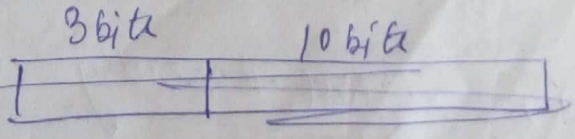
virtual memory / logical address.
let virtual address = 8K = 13 bits required to address
- for 1K = 10 bits $2^3 x = 3+10$ bits.
let Real memory / Physical add = 4K. (actual add. avail. for execution).
↳ 12 bits required to address.
let each page = 1K. ∴ each Frames = 1K.
Real memory space.

13 bit address is there but actually required = 12 bit
From 13 bit ⟶ 12 bit (transformation required by mapping).

1 word = 10 bits
2^{10} = Frame
2 = Frame bit



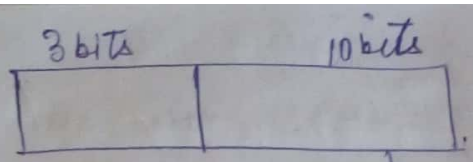
To select a peticular line / word of a Frame = 10 bit
To select a Frame we need 2 bit
∴ 10+2 = 12 bit required



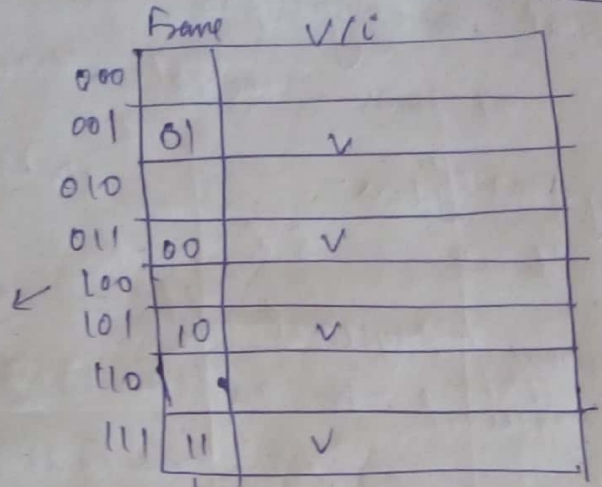
logical address :-

2=4

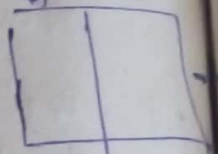
Logical address



Memory mapping Page Table

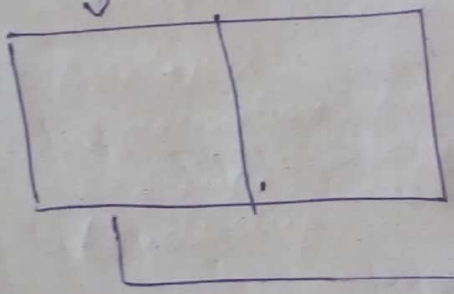


It Takes care of which page frame is in active or which is not active



Memory address register

Page Table Buffer



virtual replacement Algorithm

(i) First come first serve

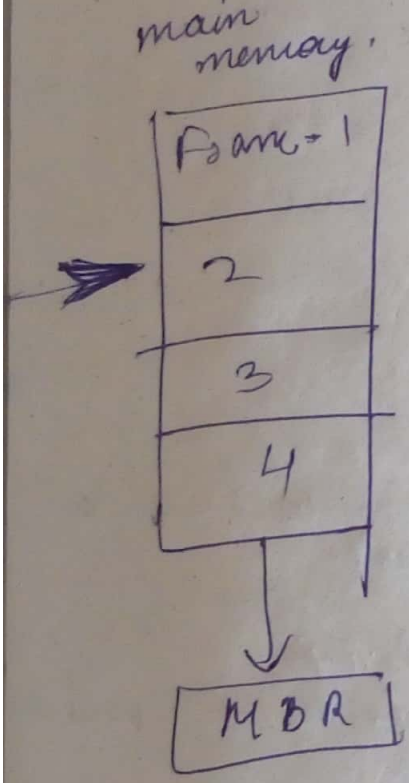
(ii) Optimal replacement algorithm (intelligent one)
Due to the experience of OS this algo works.
It will replace a page on a Frame if that will not be used for a longer time.

(iii) LRU:- Least Recently Used:-
which has not been used for a long period of time, that page will be replaced from frame by new page.

(iv)

Segmentation – itself.

Paging is not visible to user.
Segmentation is visible to user
decided



18/9/18

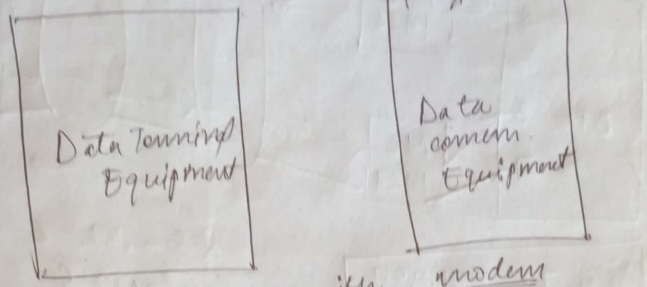
COA | Serial Standardised I/O Interface

Interface is a connector

(v) We also use standardised I/O interface

(vi) For serial comm, very widely used and early developed standardise I/O interface is RS-232-C .. The comm. may be synch or asynchronous

(vii)



Example

(1) connectn b/w sys with modem

DTE ↕ DCE

~~connection~~

comm. Interface

(2) connection b/w printer ↕ system

DTE DCE

(3) 25-pin D-type connector — DCE

and its communicaⁿ capability is 80 K BAUD with 50 fit distance.

positive true Logic in TTL :- +5 V = 1

0 V = 0

TTL

Transistor-Transist Logic

It was developed prior to TTL.

Signal Level:- +3 V to +15 V = 0 Logic DCL

of -3 V to -15 V = 1. ECT

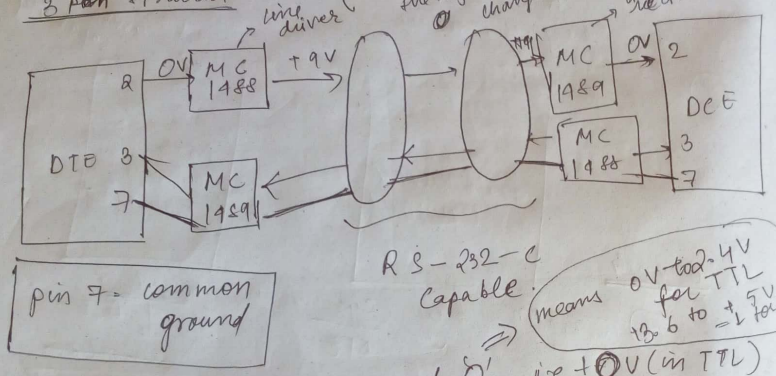
RS232C

~~20 K BAUD~~

It is a negative ~~true~~ true Logic

Minimum 3 pins required = pins: 2, 3, 7 for data communicath, may be erroneous.
For error free 9 pins are required and all other pins are unnecessary.

3 pin structure



pin 7 = common ground

RS-232-C Capable.

means 0V to 2.4V for TTL
+3.6 to +15V for TTL

⇒ ie +0V (in TTL)

(1) when DTE wants to send '0' but because of RS-232-C, 0V to 9V and MC-1489 changes and 2 recieves '0'.

(2) may or may not be error free in 3 pin structure.

Assignment :-

(1) IEEE 488 Interface.

(2) USB.

Data Transfer b/w CPU and Peripheral :-

Diff. modes are there :-

1. Programmed I/O type data transfer
2. Interrupt initiated I/O data Transfer
3. Direct memory Access (DMA) data Transfer

(1) :- various peripheral devices are connected to processor i.e. which devices are ready for data tranfer, That info. shows the with the process.

Every time before DT, a prog. to be executed whether the device

(a) wastage of CPU Time because CPU time is very imp. performance of sys. dec.

To, avoid :-

(2) Interrupt :- Every device has interrupt flag. And when interruptFlag = set (one interrupt recieved) by the CPU.

The CPU will suspend its present work and keeping the return address in stack and the prog is executed.

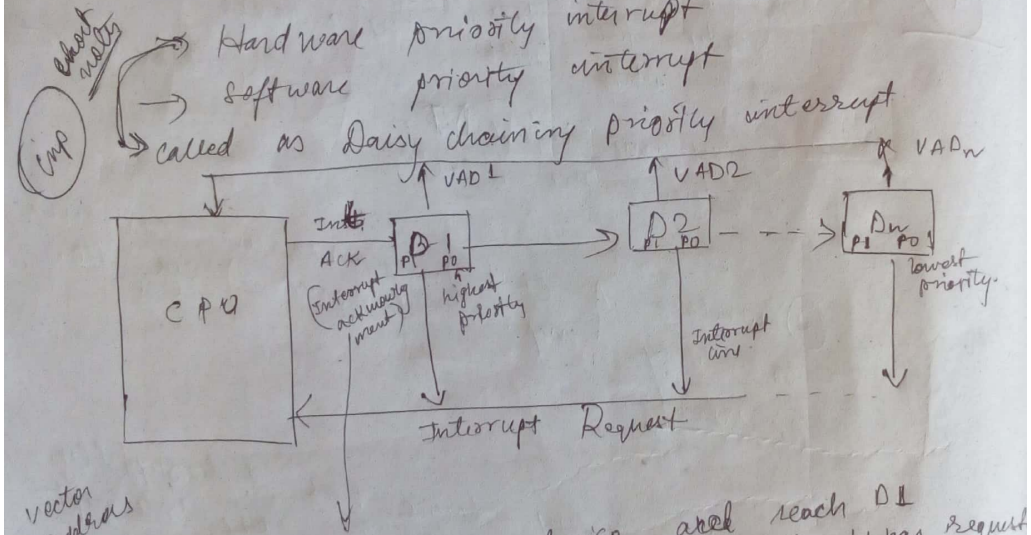
Depending on Interrupt :- (Interrupt service routine).

More than one device set interrupt flag.
CPU receives ∴ Priority wise decision is made for execution.

The highest speed device = 1st Priority
lowest n = lowest priority

highest Priority = disk | lowest Priority = keyboard.

Electro mechanical > Electro magnetic > electronic

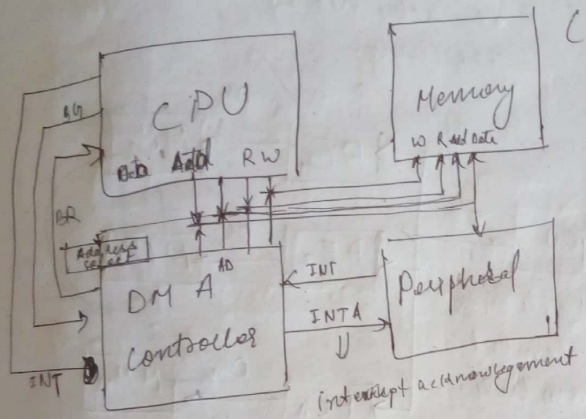


will pass through device and reach D1
∴ P1 (D1) (Priority I/P) = 1 (as it has request)
therefore P0 = 0. After the completion of task then P0 = 1
then the next device passes to next device. If no request then P1 = P0 = 1.

Disadvantage:
Last / least priority may get its chance at the end or may not get chance.

Assignment :- Software based priority interrupt.

DMA data Transfer :- without making CPU busy.



(Boom Hamature)

BR :- Bus Request
BG :- Bus Grant

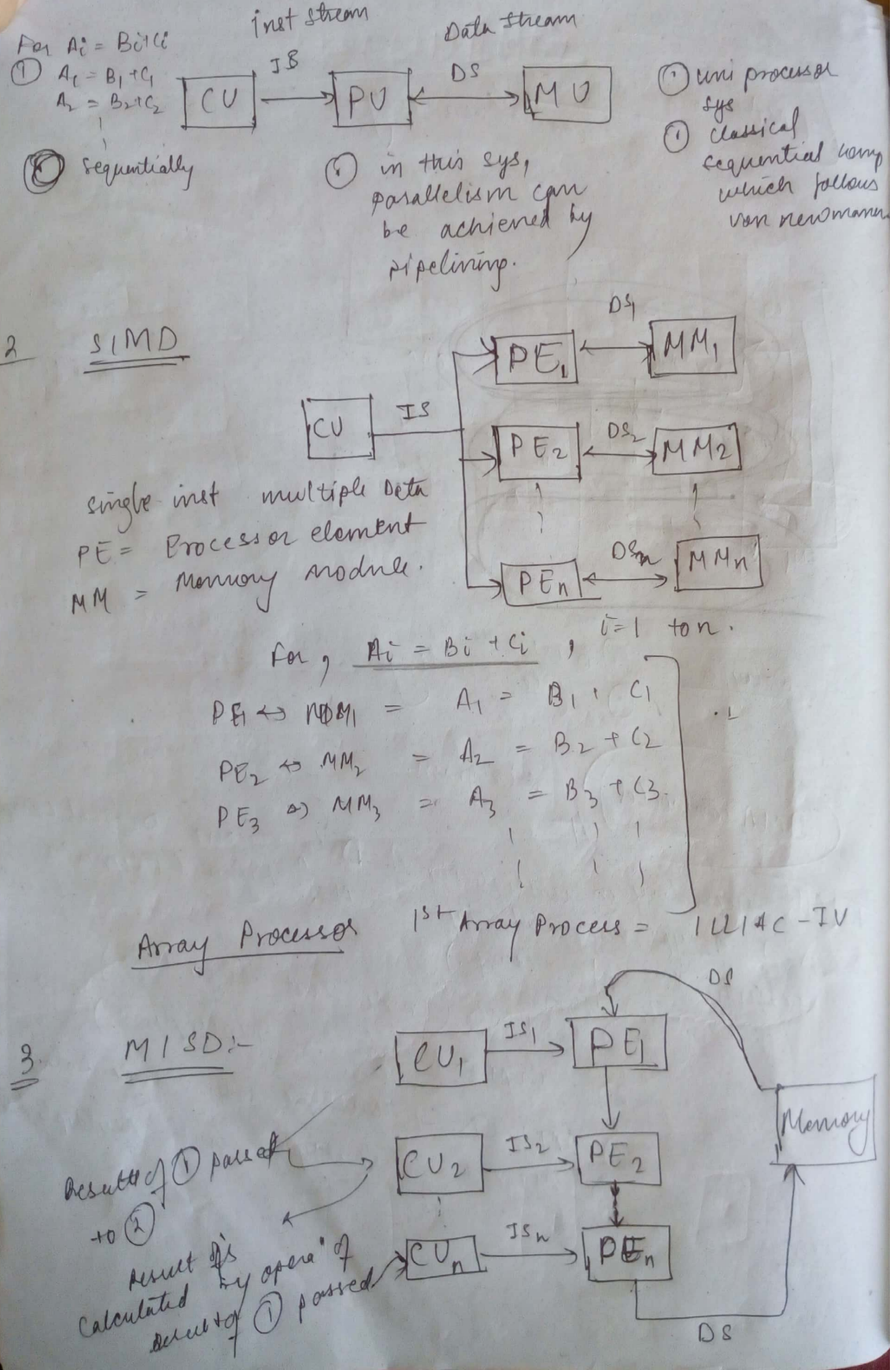
2 types :- ① Burst mode
② cycle stealing :- To steal the bus cycle from CPU.

DMA controller Diagram.

19/9/18 Depending on multiplicity of Data stream Digital comp is of 4 type suggested by M. J Flynn (Flynn's classification of comp.)

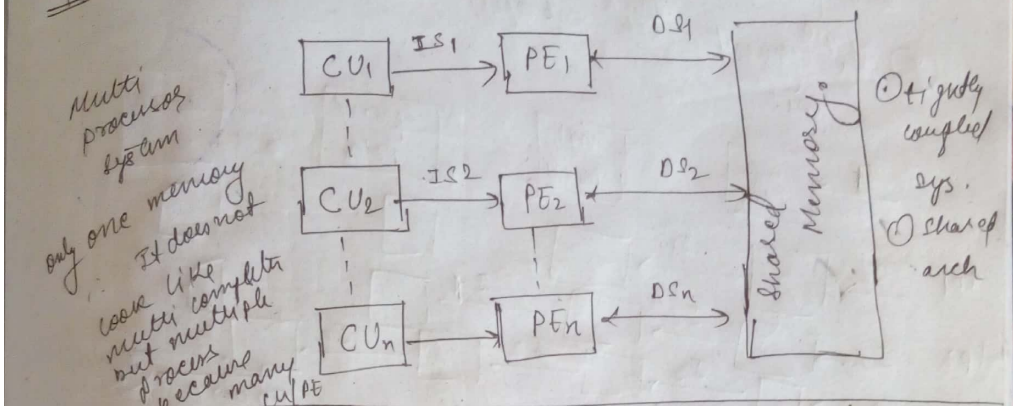
1) SISD (single Inst single Data)
2) SIMD (" multiple ")
3) MISD (multiple inst single data)
4) MIMD (multiple inst multiple data)

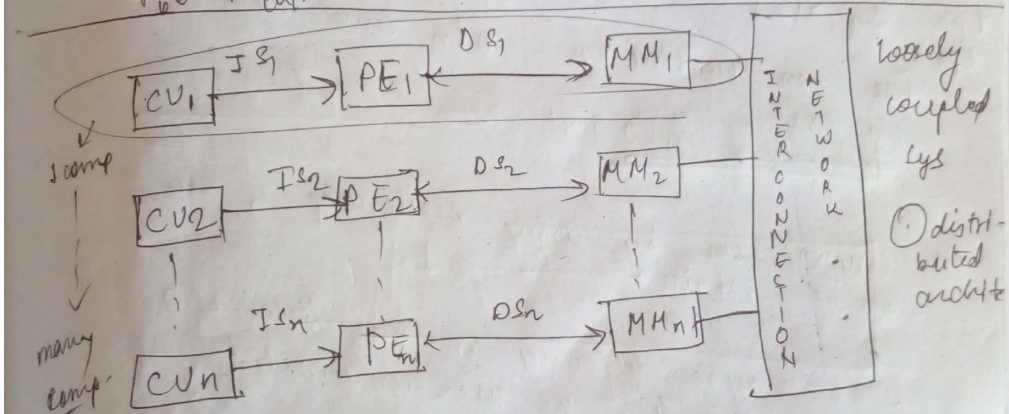
SISD :-



⊙ This architecture has not been used (hypothetical type.)
⊙ Gives the feeling of pipeline

4 MIMD :-





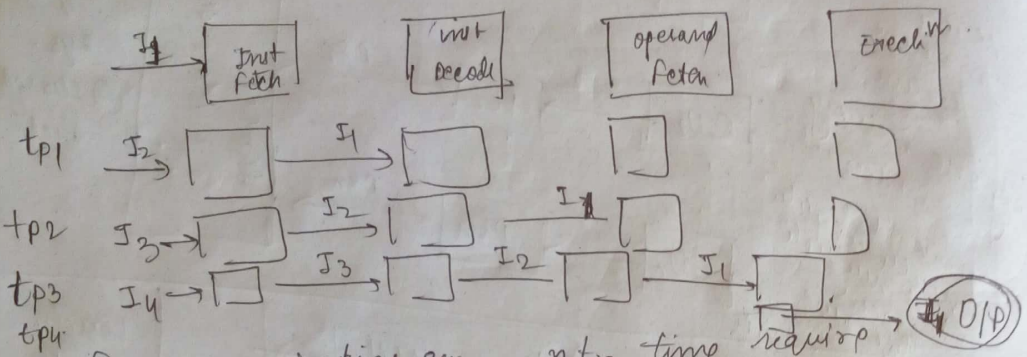
also called Multiple SIMD, multi comp. system

Diffence = interconnection network that is a communication topology.

PIPELINING

Sub operations to complete a instruction (inst cycle)

(*) if we segment each sub operation into individual hardware unit



For non pipelining sys $n t_n$ time required

n = inst

t_n = time for 1 inst

in Uniprocessor by using pipelining, parallelism can be attain

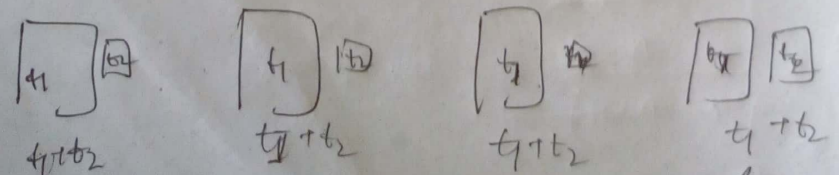
There may be difference in time factor also diff hardware units, we use intermediate register,

time taken by segmented register = t_p highest

this t_p must be decided after taking time

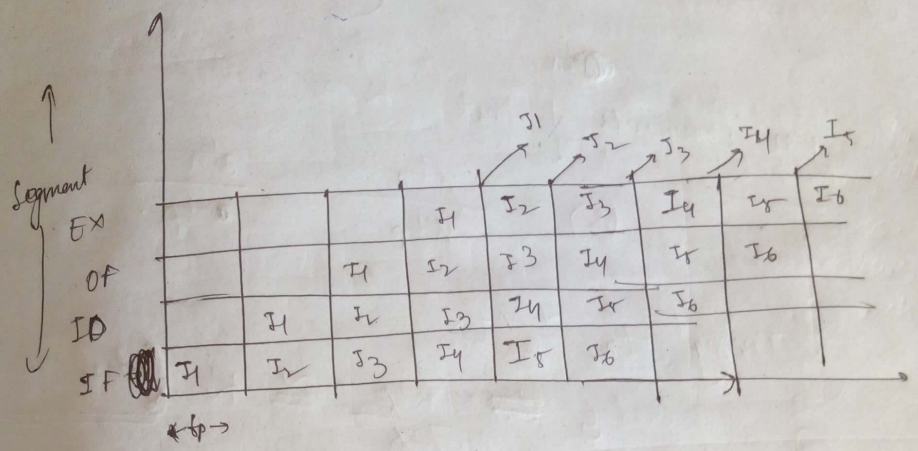
for every segment operation for

any instruction.



The value which is highest = t_p

Time Space Diagram of Pipelining :-

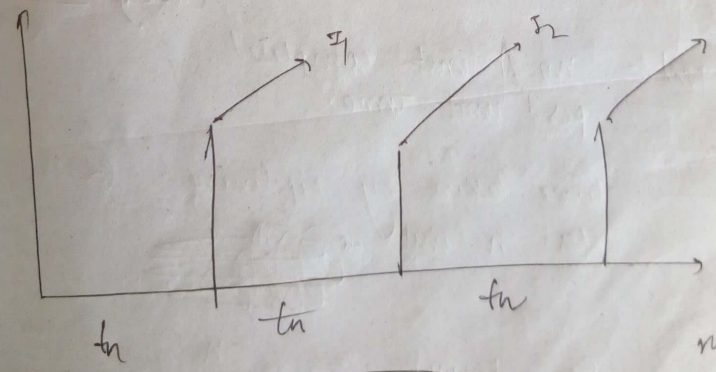


K = no. of segment
tp = segment time
n = no. of inst.

time ⟶

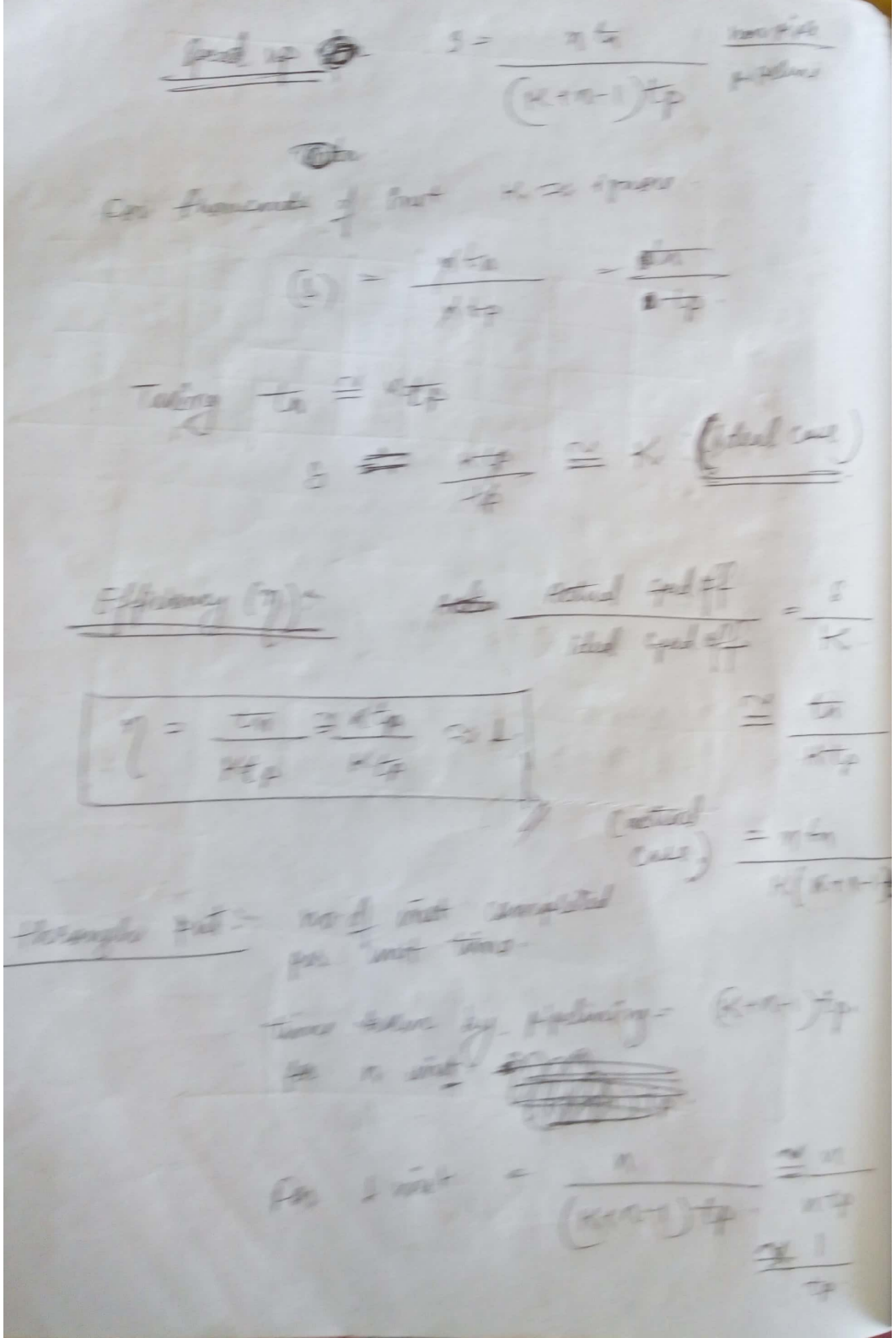
After K tp time we get 1st O/P.

total time = K tp + (n-1) tp



non pipelining

time n tn



C	+	—	—	—
D	+	—	+	—

Range J = 1.

critical incident:- evaluting individual in critical situation eg:- in breakdown, the behaviour of the empl. in such a case.

Assessment tender:-

20/4/18

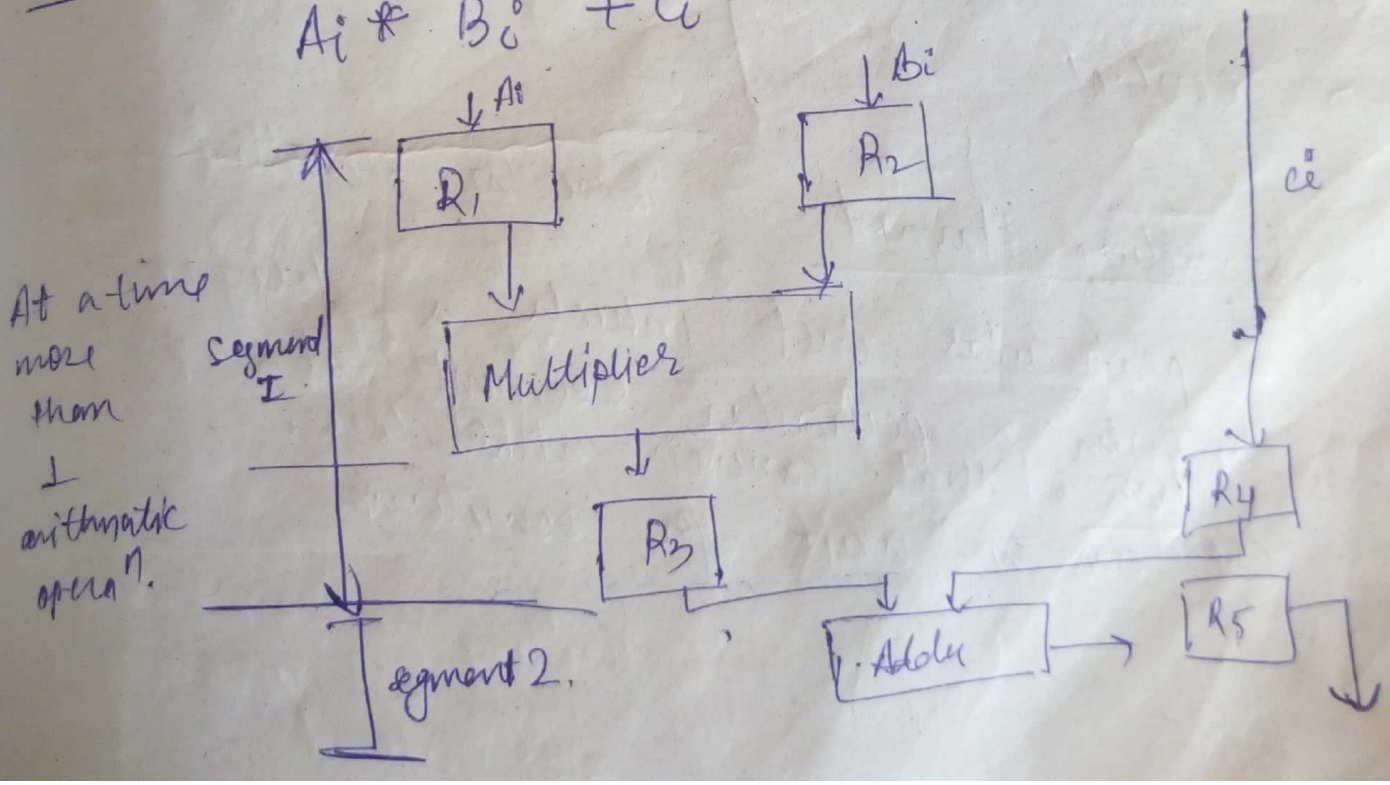
COA

2 types of pipeling :-

1. Inst. Pipeline (previous discussion.)
2. Arithmatic pipeling

Arithmatic pipeling :-

$A_i * B_i + C_i$



(*) we cannot generate a general purpose comp. for arithmetic pipeline.

(*) It is a specialised comp / system.

(*) cost is high where a particular exp / calcn is repeated regularly.

(*) Designed acc. to organisation requirement

Assignment Q Design arithmetic pipeline for floating pt Add / Sub. and demark their stages

At a designated time, which inst should be performed in what particular segment cannot be decided - This is called Pipeline Hazard.

3 types of pipeline hazard

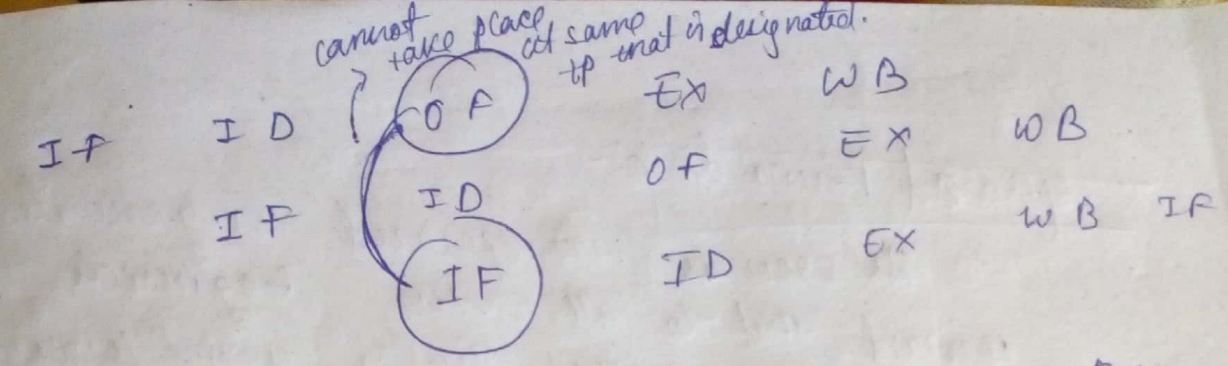
→ Structural Hazard

→ Data "

→ control

Structure Hazard :- Due to structure or architecture

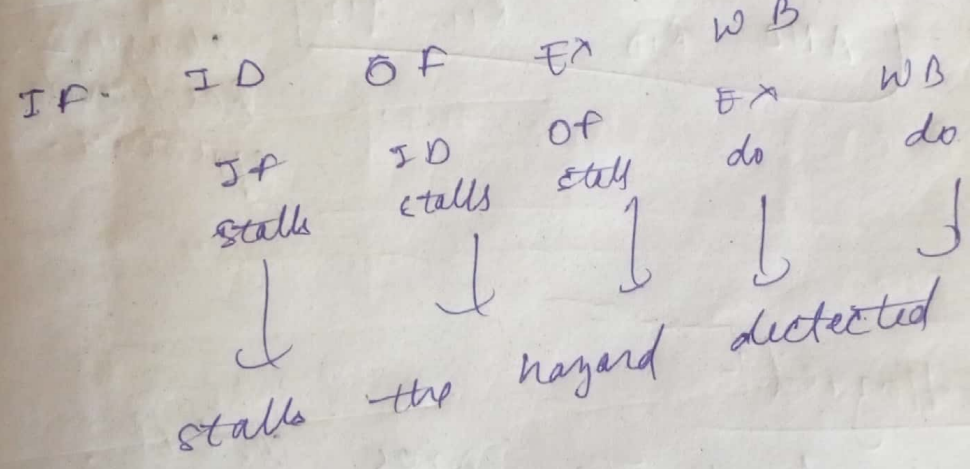
a particular inst which is designated to enter a particular segment at a particular clock but that cannot at that clock. It has to wait for sometime



If at that time, we use splitted cache one for data other for Inst, than this hazard can be avoided

Multiplicity of functional units :- multiple no. of processing element & memory module.

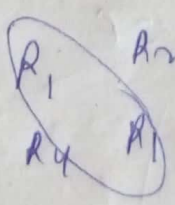
Hardware interlocks :- which stalls the clock cycle.



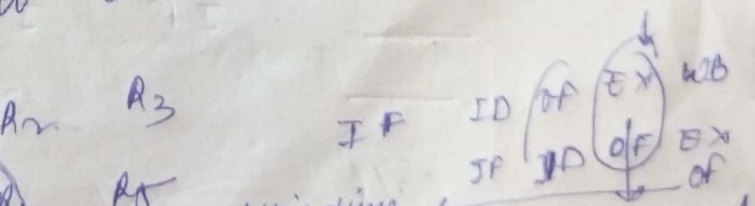
Data Hazard :- in pipeline sys

① Generally occ due to data dependency

Add R_1 R_2 R_3
Sub R_4 R_1 R_5



when data hazard occurs, the



data dependency { at this time R_1 is available for Subs R_1 is operand it is only available after execution is done at tp 4.

operand Forwarding:-

As soon as the operand is generated, forward that to all the segments where it is necessary. This is operand forwarding

Compiler scheduling:-

Before execution compilation is done, compiler changes/or reschedule the operations to avoid data hazard

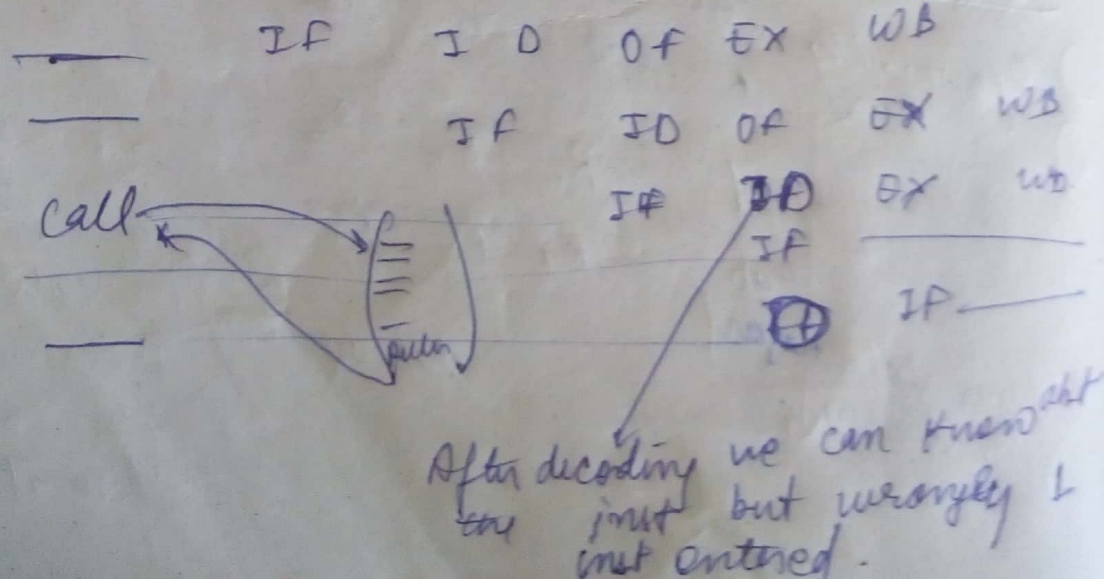
Add	R_1	R_2	R_3
sub	R_4	R_1	R_5

AND	R_{10}	R_{11}	R_{12}

independent from above 2.
∴ Reschedule the operation.

Control Hazard:-

Hazard due to prog control.



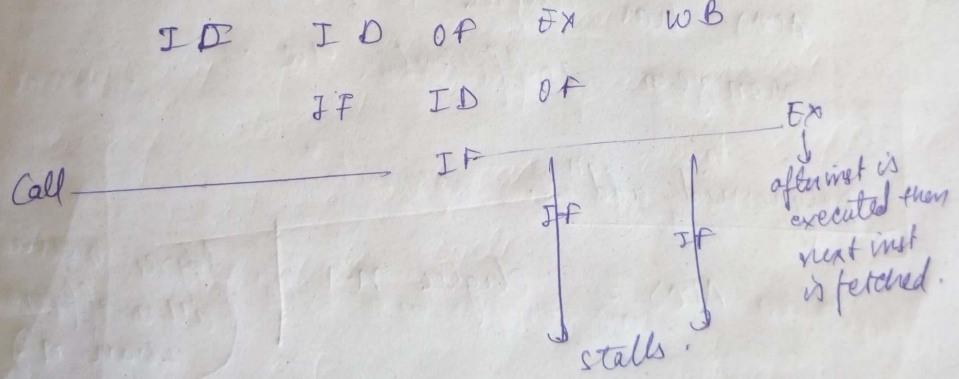
after decoding during operand fetch
If it is a JUMP 1600, then inst at address 1600, is only executed but the 2 inst corrupted entered.

(2) One to flow of control (i.e Program control)

Avoiding techniques:-

(1) Compiler scheduling

(2) hardware interrupt



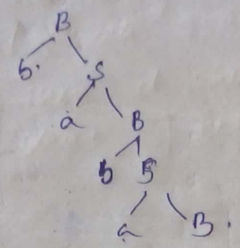
Delay inst is given to waste the inst cycle to resolve the hazard generated

$\sin(\omega t + \kappa)$

10/1/18 DAA:-

Polynomial and F Fourier Transform :-

$T(n) = O(n)$



Interrupt:-

stack/internal register/ timing device
① hard ware :- generated by harware, I/O device/power supply/ Eg n

② soft ware

Eg:- Power supply
Damage of SMPS
Timing:-

(Asynchronous) external	internal
① I/O	⊙ During any prog executⁿ
② Timing	⊙ overflow of data
③ power supply	⊙ assigned a stack/ queue where data is entered when overflow or underflow condⁿ
⊙ start of data Transfer	⊙ division by 0
⊙ end of data transfer	⊙ invalid upcode

SWI n
RST n

① Maskable interrupt
② unmaskable

Eg:- Maskable
RST 7.
Non mask
TRAP
TRAP

interrupt

External :- asynchronous

Internal :- synchronous

Software interrupt :- It is an inst. given by an which interrupts the system.

Every inst has an opcode to temporarily suspend the work/execution.

SWI n
RST n → software instruction which are given by user to prog for some specific purpose.

① maskable interrupt
⑥ unmaskable "

Eg → ~~If any inter~~

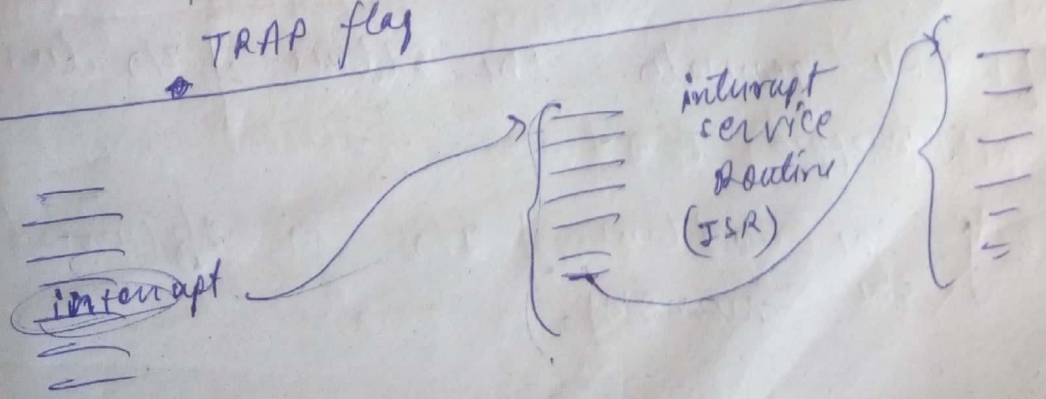
maskable interrupt (here also there are some priority)

Eg:- RST 7.5, RST 6.5, etc.

Non maskable interrupt

TRAP

TRAP flag



Two types of ISR :-

① vectors
② Non vector

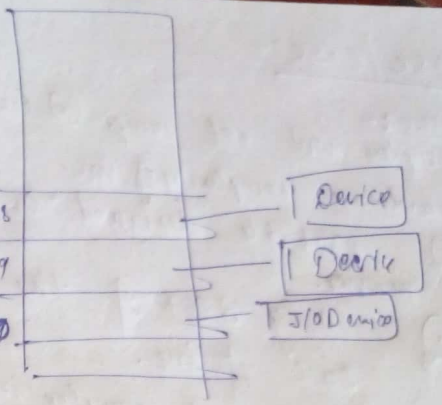
The device which generates interrupt, provides the start address of the ISR and the ISR is executed.

The interrupt that is generated by the device is first decoded (such ISR are standard interrupt) and then it automatically goes to the start address of the ISR and is executed

Types of addressing (2 types of I/O mapping :-

① I/O mapped IO / isolated I/O
② Memory mapped I/O.

① Generally some I/O has generated address,
② If CPU generates the address of I/O device then IO mapped I/O.
③ If the address of I/O is obtained from the memory.



inst is :-

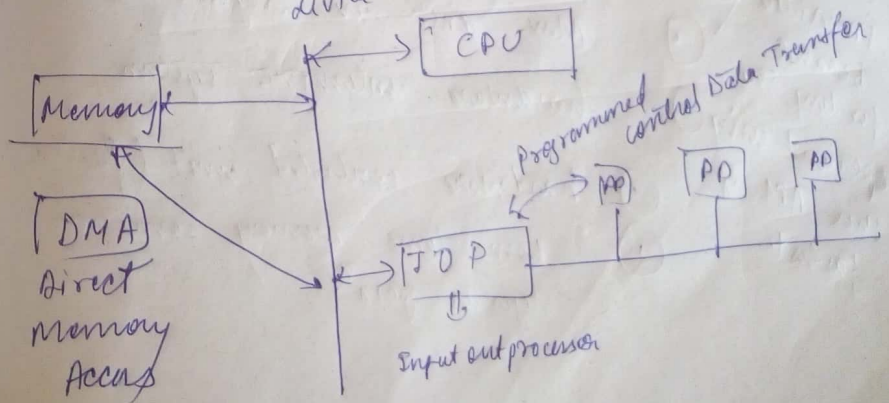
① STA 0008
↓
store the value of Accumulator in 0008
ie communicated to the device connected to it.

② LDA 0010.
Load at the address 0010 means that the device connected to the address 0010 loads its value to the address & passes it to other loch.

IO mapped I/O

IN 007 — 0E H / 0F H — in/out command from the direct address the device

↓ address of device



RISC/CISC:-

CISC:- Complex inst set comp:- ① designed during the hardware complexity inc.

① probable designed to enhance the performance.

Only 20% of the CISC inst is used 80% of time.

80% of complex inst are used by used for 20% time

To reduce of CISC architecture

pentium + Intel Processor = CISC

RISC:- reduced inst set comp:-

① very simple inst and all inst are fixed size inst.

All inst are single cycle inst except load and store.

② compilation is easy but program length and storage is more

③ Max. inst are register addressive

④ hardware intensive system

⑤ hardware controlled unit (fastest)

⑥ some time memory sometime register operand

motorola = RISC

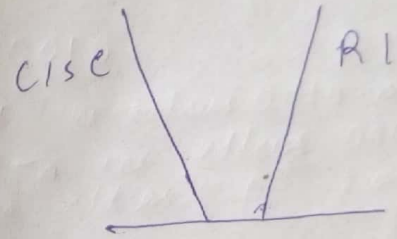
Register AM
implied AM
immidiate AM
because finding the operand using this AM is fastest

① Microprogrammed CU is flexible

controversy:- How to decide which is faster? RISC/CISC controversy.

① ~~Different arch~~

Different compiler / Different Hardware arch / diff inst set. so there is nothing common. It is difficult to decide.



② RISC/CISC convergence

① ~~Replacement Algo~~

② pipelining