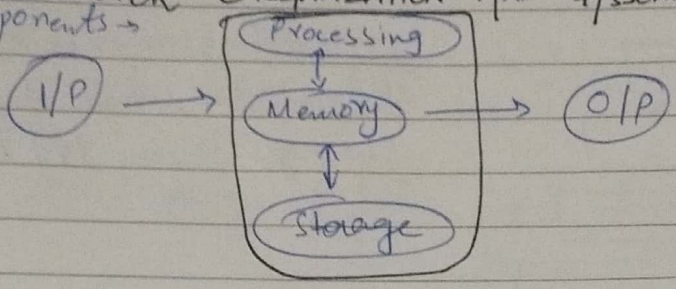
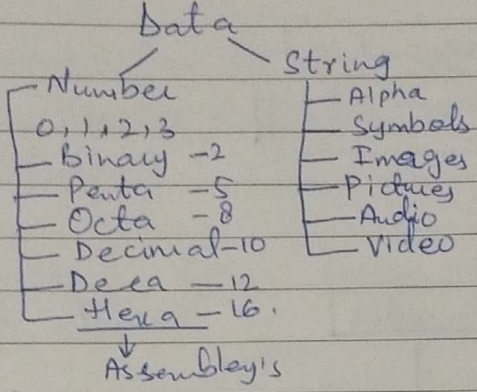
Computer Organization And Assembley Language

21/10/22

components →



Memory cells
track sectors



Assembley's

cost - Effective
- Time
- Security
- Price

Assembly language is a command based language which works on hexadecimal number system.

Mov (command) dl (address), al (value)

cut
ADD
Sub
Mul
Div

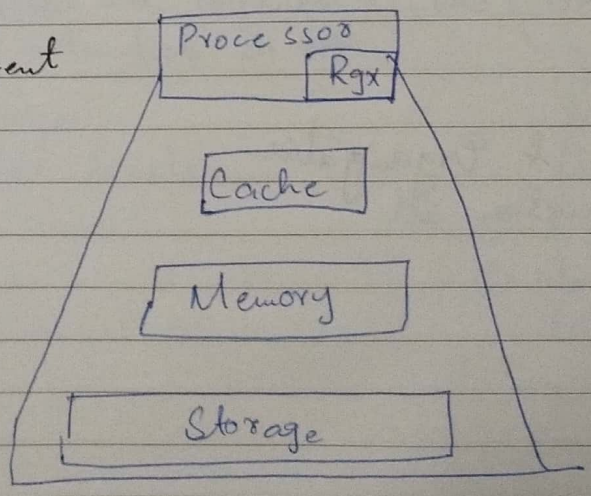
OBE
60 passing

Why Assembley language?

① assembley language is closed to machine language.
② To communicate b/w high & machine language.
③ To increase speed/optimize results.

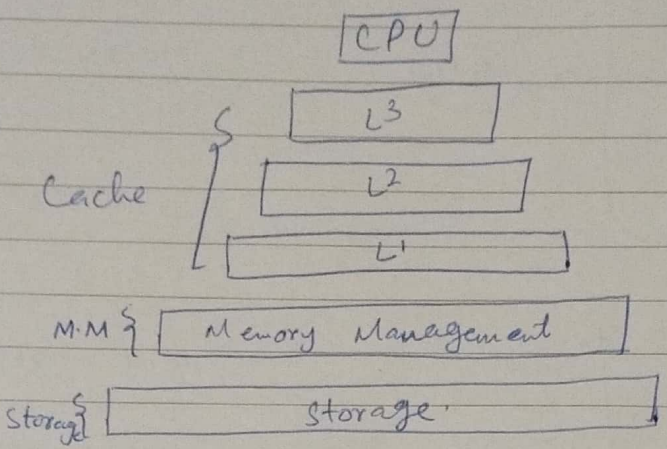
CS
↙ ↘
A.S S.S (Embeded Programming).

Memory Management



closer to processor, smaller in size, farther to processor, larger in size

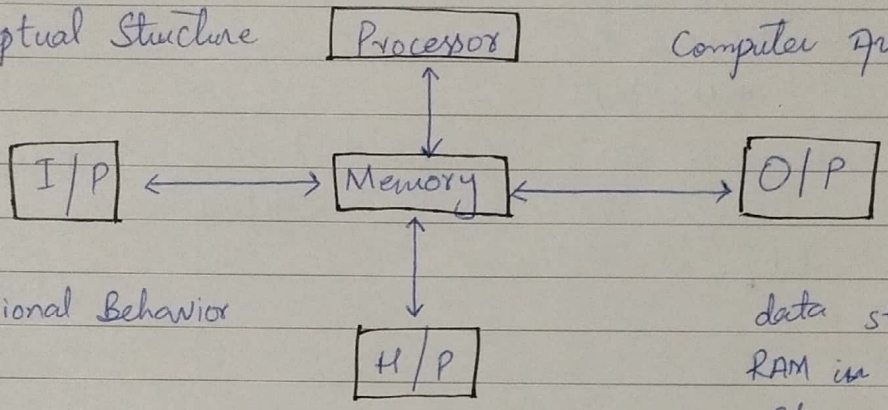
PAPERWORK



Caches are in Kbs
Registers are in bytes
Storage is in GB/TB.

25/10/22.

Conceptual Structure



Computer Architecture

Functional Behavior

data stored in RAM in the form of queue.

Computer architecture :- interconnecting hardware components in a such a logic that it'll be cost effective.

- network moving technology.

Buses

i) Data Bus → carries data
ii) Control Bus → data reaches its destination
iii) Address Bus → identifies the data's address for its destination

28/10/22

Architecture & Organization
└ Instruction set

COAL 28/10/22

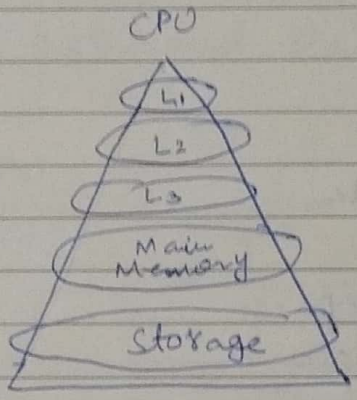
Functional View.

① Data movement → input & output of data.

② Storage → data input first stored, retrieving old data from storage (Hard disk)

③ Processing from/to storage:- get data from storage, process it (perform certain functions), then store it back

④ Processing from storage to I/O.



- insta phones.

8 bits = 1 byte

4 bits 4 bits

Quad Nibble] — 4 bits.

Dual Core Processor
Dual Processor

core → layer
↘ caches levels

① Multitasking
{Multiprogramming
↳ IDA
Time Division Accessibi
/Time Slicing
/Time Oriented Job.

② Multiprocessing
(at same time)

3/11/22

Moore's Law

$$P_n = P_o * 2^n$$

0
0
0

P_n => Power of processor in future
P_o => Power of processor at current time
n => number of years.
$n = N/2$

transistors act as switch
control temperature
increase speed.

Parallel Computing:

Parallel processing] interchangeable terms.
parallel programming
parallel tasking
Distributed computing
Cloud computing
File Distributive System

Buffering (temporary memory)

4/11/2022.

Big Data: Hadoop

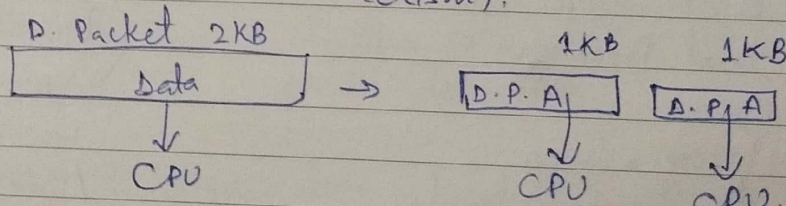
- multiple types of data.

Hadoop → Map Reduce
Shortest, fastest, secure.

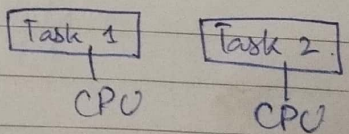
Classes of Parallelism:

① BLP (Bit-Level Parallelism).
adjust acc. to processor's size

② DLP (Data Level Parallelism).



③ TLP (Task Level Parallelism).



add (a, (b, c))^x

4/11/22

FLYNN'S TAXONOMY:

Michael Flynn divide into 4 categories (acc. to data & instructions)

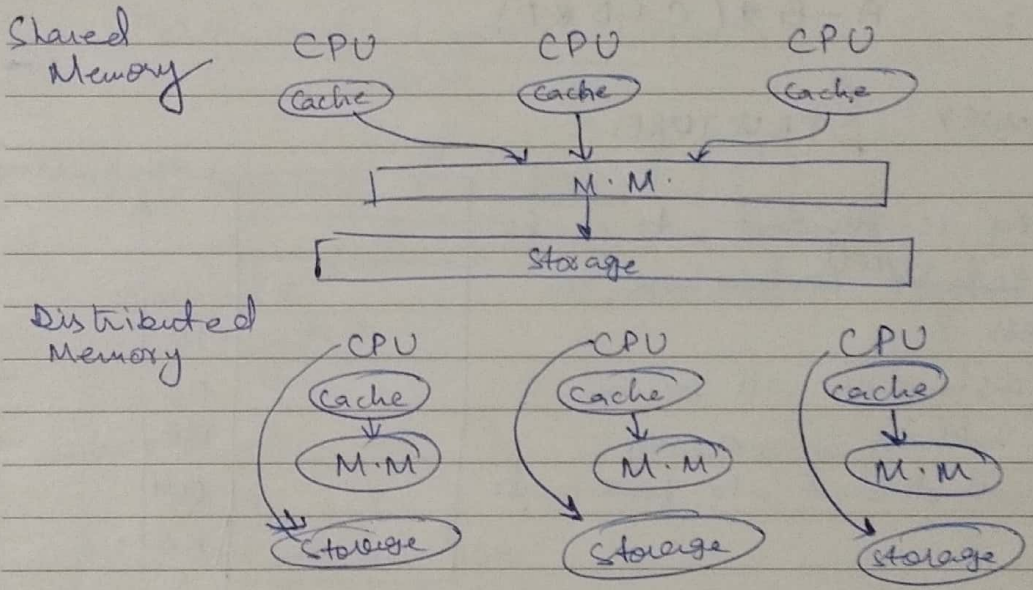
① SISD (Single Instruction Stream, Single Data Stream)
ex:- a = 2 just intruction to store data

② SIMD (Single Instruction Stream, Multiple Data Stream).
ex:- result = 2 + 4 in assembley (add : 2, 4)

③ MISD (Multiple Instruction Stream, Single Data Stream).
ex :- int & x ; pipelined (data linked)
sin(x) + cos(x) + tan(x).

④ MIMD (Multiple Instruction Stream, Multiple Data Stream).
↓ ex:- 3 * (2+2)
Shared Memory
Distributed Memory.

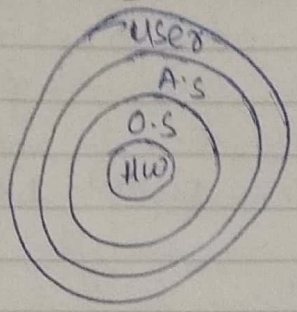
✓ add a,b
add c,d



. Mirroring → Real time Processing.

10/11/22

Instruction Set Architecture (ISA):-



- registers holds data for a time being & it also performs calculation.
- stack is a register.
- accumulator

- pointers find the address in memory.

Stack, Accumulator.

Reg - Memory.

11/11/22.

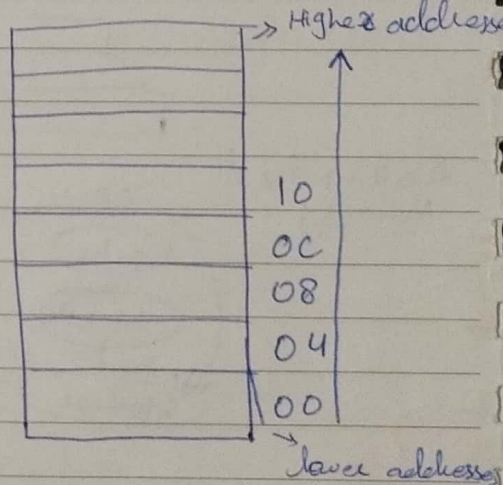
Memory-Memory Architecture.

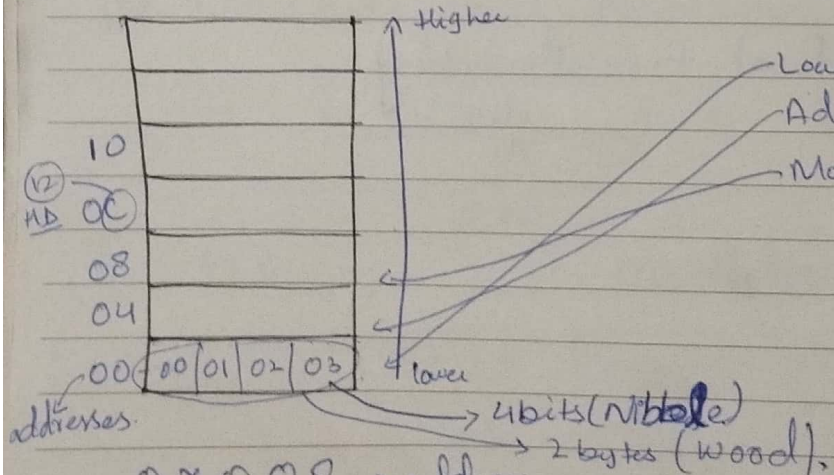
Load Store Architecture.

Assignment no. 1 Solve equations for all models of ISA :- $A - B * (C + D * E)$

Memory Structure:-

- memory is divided into cell chunks
- every ~~chunk~~ cell ~~contain~~ have an address
- first instruction will be on lower address in memory.
- every cell divided into four parts





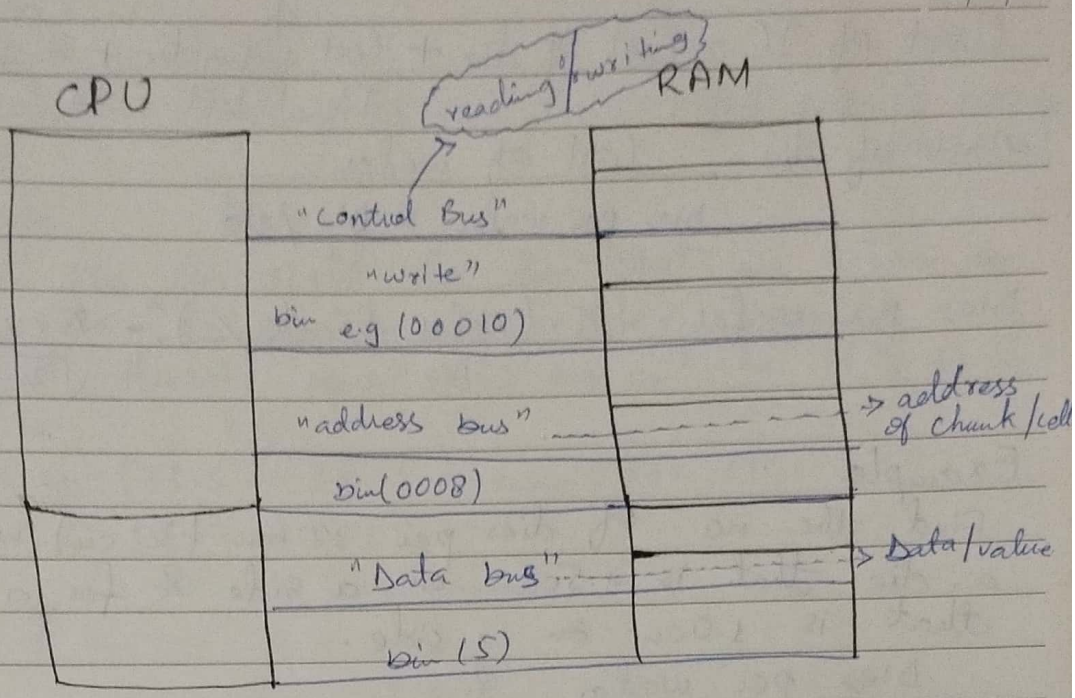
Load R_1, A

Add R_1, B

Mov R_2, R_1

0X008 → addresses

Hexa decimal



Fetch → Decode → Execute.
cycle

17/NOV/22

INTEGRATED CIRCUITS (IC)

- Discrete / Digital
- Continuous.
- Hybrid (mixed).

wafer sheet.

- modem → (it does) modulation
  conversion of digital into analog is
- modem → demodulation
  conversion of analog into digital is.

Mixed ICs.
analog → alternate current
digital → direct current

$$\text{Cost of IC} = \frac{\text{Cost of die} + \text{Cost of testing die} + \text{Cost of packaging \& final test}}{\text{Final test yield}}$$

- Germanium / Silicon is used.

$$\text{Cost of IC} = \frac{\text{Cost of die} + \text{Cost of testing die} + \text{Cost of packaging \& final test}}{\text{Final test yeild.}}$$

$$\text{Cost of die} = \frac{\text{Cost of wafer}}{\text{Dies per wafer} \times \text{Die yield}}$$

$$\text{Dies per wafer} = \frac{\pi \times (\text{wafer-diameter}/2)^2}{\text{Die area}} - \frac{\pi \times \text{wafer-diameter}}{\sqrt{2 \times \text{Die area}}}$$

Example

Find the no. of dies per 300 mm (30 cm) wafer for a die that is 1.5cm on a side & for a die that is 1.0cm on a side.

∴ Dies per wafer = ?

① area of Die = $1.5 \times 1.5 = 2.25\ cm^2$

(a) ② Die per wafer = $\frac{\pi * (W.D/2)^2}{D.A} - \frac{\pi * W.D}{\sqrt{2 * D.A}}$

$$D/W = \frac{3.14 * (30/2)^2}{2.25} - \frac{3.14 * 30}{\sqrt{2 * 2.25}}$$

$$D/W = \frac{3.14 \times 225}{2.25} - \frac{94.26}{2.12}$$

$$D/W = 314.2 - 44.46$$

$$D/W = 269.73 \text{ die}$$

(b) area of die = $1 \times 1 = 1\ cm^2$.

$$D/W = \frac{\pi * (30/2)^2}{1} - \frac{\pi * 30}{\sqrt{2 * 1}}$$

$$D/W = \pi * 225 - \pi * 30/\sqrt{2}.$$

$$D/W = 640.21$$

Install DOSBOX in your PC.

18/11/22

How die made?

Die yield = Wafer yield $\times$ 1 / (1 + Defects per unit area, N * Die area)

Example

Find the die yield for dies that are 1.5cm a side & 1.0cm on a side, assuming a defect density of 0.031 per cm^2 & N is 13.5.

$N \rightarrow (11.5 - 15.5)$ range for N.

a) 1.5 cm $\rightarrow$ $A = 1.5 \times 1.5$

$A = 2.25 cm^2$.

b) 1 cm $\rightarrow$ $A = 1 \times 1$

$A = 1 cm^2$.

(a) $D.Y = \frac{1 * 1}{(1 + 0.031 * 2.25)^{13.5}}$

$D.Y = 0.402$

(b) $D.Y = \frac{1 \times 1}{(1 + 0.031 \times 1)^{13.5}}$

$D.Y = 0.66$.

- Die area decreases die yield increases

DEPENDABILITY:

- Reliability
- Availability

Module Reliability:

FIT : Failure In Time

MTTF : Mean Time To Failure

MTTR : Mean Time To Repair

MTBF : Mean Time B/w Failures

Q. A system having failure in time 1000. Calculate its mean time to Failure, if the mean time to recover is 72 hours then what would be the mean availability of the system?

FIT = 1000 = 10^3.

MTTF = ?

MTTR = 72 hrs

M.A = mean availability = ?

$MTTF = \frac{10^9}{FIT}$ → billion main time nikalna

$MTTF = \frac{10^9}{10^3}$

$MTTF = 10^6$

$$M.A = \frac{MTTF}{MTTF + MTTR}$$

$$M.A = \frac{10^6}{10^6 + 72.}$$

$$\boxed{M.A = 0.99.}$$

Q. In an organization, ~~the~~ a system has failure in time 10000. Find its mean time to failure & if system takes 60 hours to run again, what would be the availability of the system?

FIT = 10000 → 10^4

MTTR = 60 hrs

MTTF = ? M.A.

$$MTTF = \frac{\cancel{FIT = 10^9}}{FIT} = \frac{10^9}{10^4} = \boxed{10^5}.$$

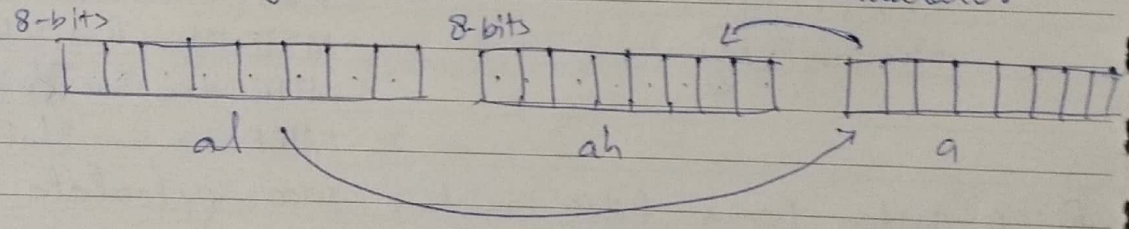
$$M.A = \frac{MTTF}{MTTF + MTTR} = \frac{10^5}{10^5 + 60} = \boxed{0.99}$$

REGISTERS:

fastest storage locations.

14 Registers

1st register → 8 bits → a-accumulator.



x2 { Ea → Enhanced accumulator. → 16 bits
x2 { Xa → Extra Enhanced accumulator. → 32 bits
x2 { Ra → Reg a → 64 bits

a → accumulator.

1. ax → accumulator (enhanced)
2. bx → base register
3. cx → counter register
4. dx → data register

G.P.R General Purpose Registers.

5. cs → code segment.
6. ds → data segment
7. ss → stack segment
8. es → extra segment.

} → Segment Registers.

9. si → segment index ~~index register~~.
10. di → data index

} source index, destination index → Index registers.

11. ip → index pointer
12. sp → segment/stack pointer

} Pointer registers Special.

13. f → Flag resiter.

14. b → base register.

G.P.R

- Accumulator :: Input / Output, Operations.
- Base :: Holds address of data.
- Counter :: Used in loop. (eg PC counter)
- Data :: Holds data for output.

Special.

- Instruction Pointer next instruction.
- Stack pointer: top instruction of stack.

- Flag register: Flag register holds if carry or is borrow

- Base register: top of the stack will be considered as base.

COAL 24/11/22

Commands for DOS BOX:-

path :-> mount C C:\MP. dir

=> C:

=> edit class_1.asm (Creating file) (and to open the file) save the file.

for comments:- ; ----

int -> interept. nasm is assembler

=> nasm File.asm. -> (to assemble the file)

↳ nasm File.asm -o File.com

com - command executable file

=> afd File.com -> afd is a debugger

entered in CPU's brain

0100	B80500	MOV AX,0005
↓	↓	↓
address of instruction	Machine Code	assembly code!

F1

one to one correspondence.

25/NOV/22

DOS's executeable file is com.

[org 0X0100]

DOS will start instructions from 0100.

B8 0500

↓

B8 -> is opcode for mov ax command

BB -> is opcode for mov bx

Friday
25/11/22

B80500 mov ax, 5

B8 → mov ax.
5 → 0500 — ??
Why 0500, why not 0005
{ISSUE 0500}

0005 0500

Intel ⇒ Lower → higher
Other ⇒ Higher → Lower

LSByte MSByte
↓ ↓
Lower Higher
Add Add

→ so, it is [5]

Mov ax, [0x4c00] → exit command
int 0x21 → program will interrupt itself then OS will come to move to the command for exit.

Interrupts
Hardware base & Software base
Maskable & Non-maskable
↓ ↓
ignores until work is done. can't ignore
↓ ↓
high priority low priority

Multiple interrupts.

Saad

1st/Dec/2022

Label/Title/Tag =

Label: mov ax, 5
Address label.

dw → define word.

Register size ⇒ word (16 bit register).
4 bits ⇒ nibble.

any declaration/instruction after interrupt has global scope.

global valiables are loaded first from hard disk to RAM. It is done by Loader.
after global decbration, it will move to first Program Counter's instuction & then so on.

```
mov ax,5  ; hard coded → immediate operands
mov ax,bx ; register to register
x mov 5,ax ; not possible
```

[ ] → calling data from memory

e.g.. mov ax, [num1] ; reading data from memory (memory to register).

. mov [num1], ax ; writing data from register into memory (register to memory)

. mov [num1], [num2] ; reading & writting at same time (nope!) (memory to memory)

:- First Code:

2-Dec-2022

```
[org 0x0100]

Mov ax, [num1]
Mov bx, [num2]
Add ax, bx

Mov bx, [num3]
Add ax, bx
Mov [num4], ax.

Mov ax, 0x4c00
int 0x21

Num1: dw 5
Num2: dw 8
Num3: dw 12
Num4: dw 0
```

Mov bx, [num1+2].

Mov bx, [num1+4]

Mov [num1+6], ax

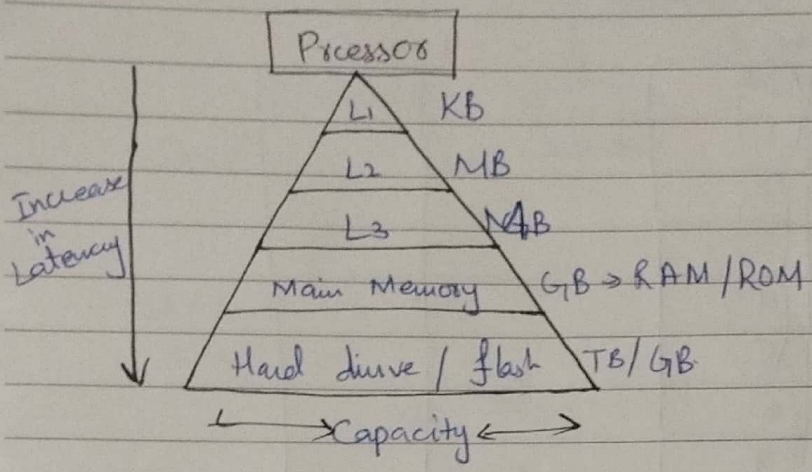
{ m1 0109

m1 → RAM's Memory

0109 → Address of instuction

2/12/22

Memory Hierarchy Design



Temporal → Virtual time
Space
] both are inversely proportional to each other

Speed in memory is vice versa.

→ mov ax, 2 → operands
op/instruction

L_1 : I cache | D cache

I → Instruction
D → Data.

L_2 : U - cache
L_3 : U - cache

U → unified / defined / reserved.

1. Direct Mapping: ✨ 1 to 1 Mapping
bytes / block
2. Set Associative Mapping
3. Full Associative Mapping.

M.M

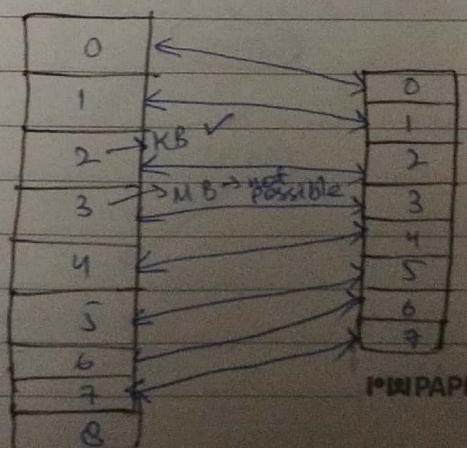
Direct Mapping

every block of cache will be mapped to M.M

Formula for calc. of mapping

M.M Block Address
MOD (%)
No. of Blocks in Cache

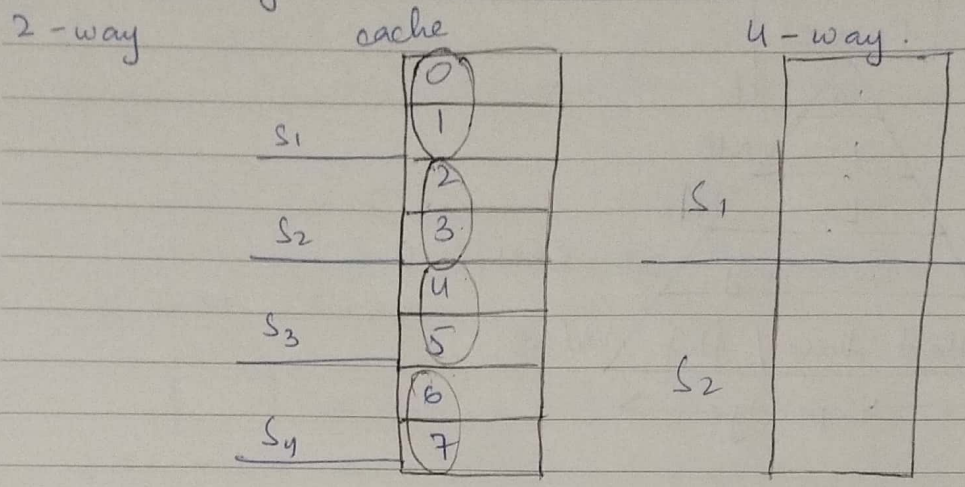
(decides which cache's block will be mapped to which M.M's block)



PAPERWORK

(2) set Associative Mapping:

n-way set association.



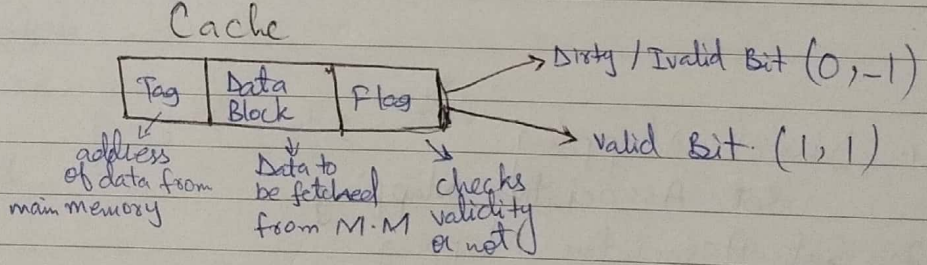
(3) Full Associative Mapping

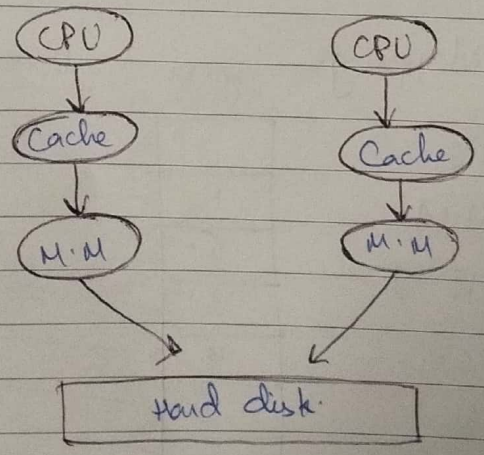
Full cache will be mapped to a block of M.M.

Miss / Hit

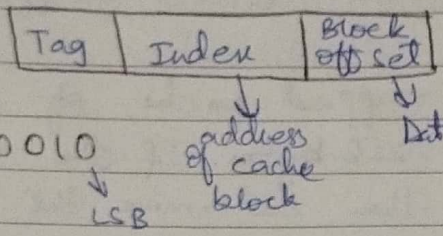
more miss, more penalty.

access time = miss time + penalty time



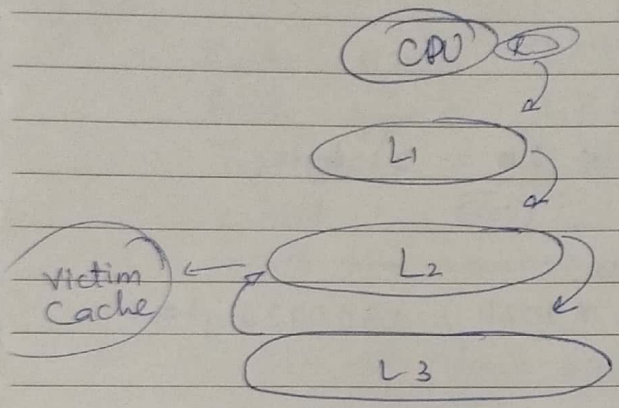


RAM

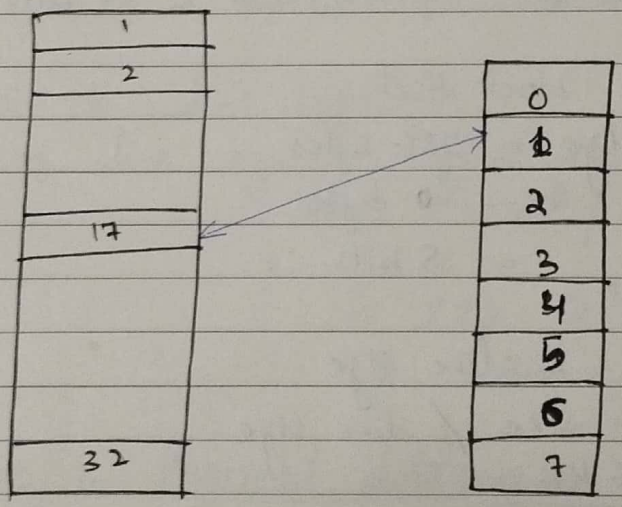


LRU => Least Recently Used

12 % 8 = 4



8/Dec/2022



Address of M.M.B MOD No of B of cache.

17 % 8

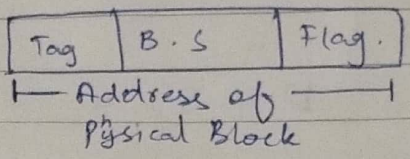
= 1

Practice Problems on Direct Mapping

Problem #1

Consider a direct mapped cache of size 16KB with a block size of 256 bytes. If the size of the main memory is 128KB - then determine the bits in the tag?

Solution



Cache size = 16KB.

B.S / F.S / Line size = 256 bytes

Block Frame

M.M.S = 128KB. = 2^7KB x 2^{10} = 2^{17} bytes.

(1024)

No. of Bits in tag = ?

No. of Bits in Physical Address:

M.M. Size. => 128Kbs x 1024 = 131072 bytes.

=> 2^{17} Bytes

=> 17 bits

131072 can be represented in 17 bits.

# of bits in block offset :

Block size = 256 bytes

= 2^8 bytes

= 8 bits.

# of bits in a line size.

=> Cache size / line size.

$$\Rightarrow \frac{16\,KB}{256\ bytes}$$

$$\Rightarrow \frac{2^4 KB}{2^8 B}$$

$$\Rightarrow \frac{2^4 \times 2^{10} B}{2^8 B}$$

$$\Rightarrow \frac{2^{14} B}{2^8 B}$$

$$\Rightarrow 2^{14-8}\ Bytes$$

$$\Rightarrow 2^6\ Bytes \Rightarrow 6\ bits$$

# of Bits in tag
= # of Physical address bits − no. of bits in line size − Block offset

= 17 − 6 − 8

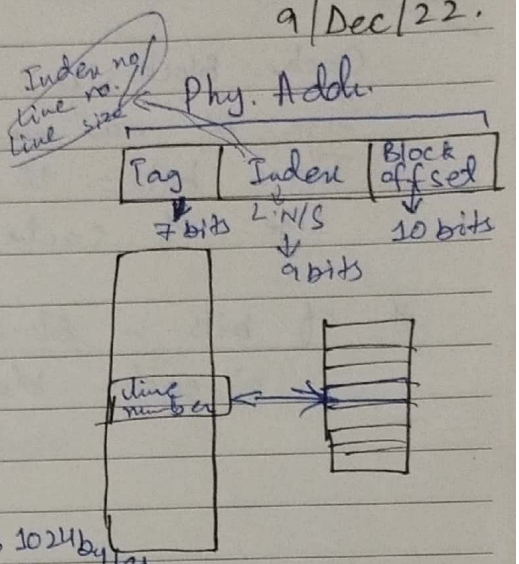
= 3 Bits

9/Dec/22.

Problem no. 02

Solution:

Cache size = 512 KB
Blok size = 1 KB
Tag bits = 7 Bits
S.M.M = ?
size of main memory.



# of Bits of Block offset.

⇒ Cache Block size = 1 KB. ⇒ 1024 bytes
⇒ 2^{10} bytes
⇒ 10 bits

# of Bits in line size.

# of line size = Caches Size / line size.
⇒ 512 KB / 1 KB.

⇒ 2^9 lines
⇒ 9 Bits.

# of Bits in Physical address.

⇒ Tag bits + L.size Bits + B.offset bits
⇒ 7 + 9 + 10.
⇒ 26 bits.

SMM

= # of Bits in Physical address.
26 bits.

$\frac{2^{26}}{2^{10}} \Rightarrow 2^{16}$ Kb ⇒ 2^6 mb = 64mb

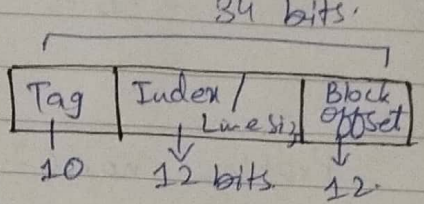
PAPERWORK

Problem # 03.

A direct mapped cache with block size of 4KB. The size of main memory is 16 GB & there are 10 bits in the tag. Find.

① Size of cache memory.

Solution



Cache Block size = 4kb

Size of main memory = 16GB.

Tag's bits = 10 bits

size of cache memory =

# of bits in Block offset

=> cache block size = 4kB = 4 X 1024 bytes

$\Rightarrow 2^2 \times 2^{10}$

$\Rightarrow 2^{12}$

=> 12 bits

# of bits in physical address.

Size of M.M. = 16 GB.

$\Rightarrow 2^{10} \times 2^{10} \times 2^{10} \times 2^4$ (Mb, KB, Byte)

$\Rightarrow 2^{34}$ bytes

=> 34 bits

Cache memory size

= # of Bits of line size = 12 bits.

=> Cache Block size = 2^{12} bytes. lines

= 4096 ~~bytes~~.

= 4kb

Cache memory size = Total no. of lines in cache X Line/Block size/Frame

$= 2^{12} \times 4KB$

$= 2^{12} \times 2^2 \times 2^{10}$

$= 2^{24}$ Bytes

$= 2^{14}$ kb

$= 2^4$ Mb = **16 Mb**

22/12/22

Consider a two way set associative mapped cache of size 16 KB with the block size of 256 bytes. If the size of the main memory is 128 KB then determine the bits in tags?

17 bits in Phy. Add.

Tag	L/S	B. offset
4	5	8

Solution

Set size = 2.
Cache size = 16 KB
Line / Block size = 256 bytes.
Main memory size = 128 KB.
# of bits in tag = ?

1. # of bits in physical address:
M.M size = 128 KB
= $2^7 \times 2^{10}$ bytes
= 2^{17} bytes
= 17 bits

2. No. of bits in block offset:
Line size = 256 bytes = 2^8 bytes
= 8 bits

3. No. of lines in cache.
= Cache size / Block size.
= $\frac{16\text{ KB}}{256\text{ bytes}} = \frac{2^4 \times 2^{10}\ \text{bytes}}{2^8\ \text{bytes}} = \frac{2^{14}}{2^8} = 2^6$ lines
= 64 lines.

4. No. of sets in cache:
set size = 2
$\frac{\text{No of lines}}{\text{set size}} = \frac{64}{2} = 32$ sets.
= 2^5 sets.
= 5 bits

5. No. of Tag bits.
= P.A bits − L.S bits − B.O bits
= 17 − 5 − 8
= 4 bits.

PAPERWORK

22/12/22

Let a 8-way set-associative mapped cache of size 512 KB with a block size of 1 KB. If 7 bits are used in tag then determine the size of main memory?

Solution

23 bits in P.A.

Tag	L.S	B.O
7	6	10

Set = 8

cache size = 512 KB

block size = 1KB | Tag bits = 7.

Size of MM = ?

1. No of bits in block offset:

Block / line size = 1KB = 2^{10} bytes

10 bits

2. No. of bits in lines

$$\frac{\text{Cache size}}{\text{line size}} = \frac{512\,KB}{1\,KB} = \frac{2^9 \times 2^{10}\ bytes}{2^{10}\ bytes} = 2^9 \text{ lines}$$

3. No. of sets.

$$\frac{512}{8} = 64 = 2^6 \text{ sets.}$$

6 bits

4. No. of bits in P.A.

= Tag bits + L.S bits + B.O bits

= 7 + 6 + 10.

= 23 bits.

5. Size of main memory.

23 bits.

2^{23} bytes. $\Rightarrow 2^{13}$ KB $\Rightarrow 2^3$ MB

$\Rightarrow$ 8 MB

22.12.22.

Consider a 4-way set associative mapped cache with a block size of 4KB. If the size of main memory is 16 GB and 10 bits are used in tag then determine the size of the caches?

Solution

Set = 4
Block size = 4KB.
Size of M.M = 16 GB
Tag bits = 10 bits.
size of cache = ?

34 bits in P.A.

Tag	L.S	B.O
10	12	12.

# of bits in block offset:
Block size = 4KB = $2^2 \times 2^{10} = 2^{12}$ bytes
$\Rightarrow$ 12 bits

# of bits in physical address.
size of M.M = 16 GB = $2^4 \times 2^{10} \times 2^{10} \times 2^{10} = 2^{34}$ bytes
$\Rightarrow$ 34 bits.

~~#~~ 2^{12} ~~bytes~~ sets $\times 2^2$ lines. $= 2^{14}$ lines.

2^{14} sets $\times 2^2$ KB $\Rightarrow 2^4 \times 2^2 \times 2^{10} \times 2^{10}$
$\Rightarrow 2^6$ MB
$\Rightarrow$ 64 MB.

12 bits in line size means 2^{12} sets. = 12 bits
No. of sets in cache = 2^{12} => 12 bits
Since cache is 4-way mapped each set contain 4 lines.
Hence, no. of lines in cache => $2^{12} \times 4$ lines
$2^{12} \times 2^2$
2^{14} lines.

Size of cache memory => Total lines x Block size.
2^{14} x 4KB
2^{14} x 2^2 KB
2^{16} KB
2^6 MB = 64MB

PAPERWORK

23/12/22

Problem - 07:

Consider a direct mapped cache with 8 cache block (0-7). If the memory block requests are in the following order. 3, 5, 2, 8, 0, 6, 3, 9, 16, 20, 17, 25, 18, 30, 24, 2, 63, 5, 82, 17, 24. Which of the following memory blocks will not be in the cache at the end of the sequence?

1) 3
2) 18 ✓
3) 20
4) 30.

Also, calculate the hit and miss ratio.

C.M

L-0	8 0 16 24	*
L-1	~~9~~ 17 25 17	
L-2	2 18 2 82	
L-3	3	*
L-4	20	
L-5	5	*
L-6	~~8~~ 30	
L-7	63	

M		M	
3 % 8 = 3	- M	17 % 8 =	1
5 % 8 = 5	- M	25 % 8 =	1
2 % 8 = 2	- M	18 % 8 =	2
8 % 8 = 0	- M	30 % 8 =	6
0 % 8 = 0	-	24 % 8 =	0
6 % 8 = 6		2 % 8 =	2
3 % 8 = 3	H	63 % 8 =	7
9 % 8 = 1		5 % 8 =	5 H
16 % 8 = 0		82 % 8 =	2
20 % 8 = 4		17 % 8 =	1
		24 % 8 =	0 H

Hit Rate = $\frac{3}{21}$

Miss Rate = $\frac{17}{21}$

Hit : Miss

$\frac{3}{21}$: $\frac{17}{21}$

Problem: 8

Consider a fully-associative mapped cache with 8 cache blocks (0-7). The memory block requests are in order.

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

If LRU replacement policy is used, which cache block will have memory block 7?

Also calculate hit ratio and miss ratio. (17 blocks)

Least Recent Used - LRU.

* → Hit

	C.M	
L-0	~~4~~ 45	
L-1	~~3~~ 22	
L-2	25	* *
L-3	8	* *
L-4	~~19~~ 3	
L-5	~~6~~ 7	
L-6	16	*
L-7	35	

Line 5 will have memory block 7.

Hit rate = 5/17

Miss rate = 12/17.

Hit : Miss

5/17 : 12/17

Problem: 9

Consider a 4-way set associative mapping with 16 cache blocks

Problem: 9

Consider a 4-way set associative mapping with 16 cache blocks. The memory block requests are in order:

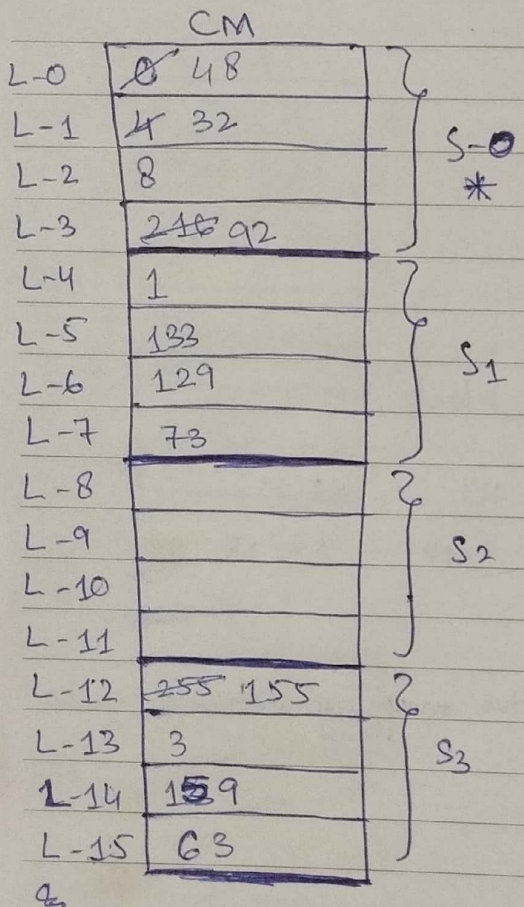
0, 255, 1, 4, 3, 8, 133, 159, 216, 129, 63, 8, 48, 32, 73, 92, 155.

If LRU replacement policy is used, which cache block will not be present in the cache?

1. 3
2. 8
3. 129
4. 216. ✓

17 blocks.

* → Hit



Assignment.

Problem- 11

Q1. Consider a cache has 4 blocks for the following memory refferences

5, 12, 13, 17, 4, 12, 13, 17, 2, 13, 19, 13, 43, 41, 19

what is the hit ratio for the following cache replacement algorithms, and miss ratio.

1. FIFO
2. LRU
3. Direct mapping
4. 2-way set associative mapping with LRU

Q2. A heirarchical memory system has the following specification, 20MB main storage with access time of 300ns, 256 bytes cache with access time of 50ns, word size of 4 bytes, page size 8 words. What will be the hit ratio if the page address trace of a program has the pattern 0, 1, 2, 3, 0, 1, 2, 4 following LRU page replacement technique?

Q3. Consider an array A[100] and each element occupies 4 words. A 32 word cache is used and divided into 8 word blocks. What is the hit ratio for the following code.

```
for (i=0; i<100; i++)
    A[i] = A[i] + 10;
```

DONE

29/12/22

Q.1 A main processor, processing 20 instructions of a processor. Each instruction requires 2.5 ns to get executed. Calculate the size of the main process, time taken to complete the process & time require for each instruction

Solution

① size = 20 Instructions

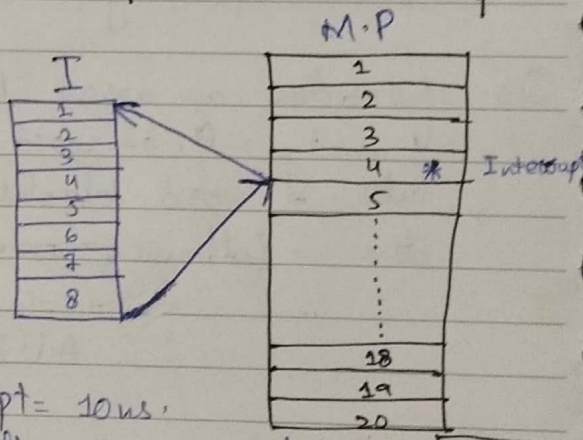
② Time = 2.5 × 20 = 50 ns for whole process

③ Time = 2.5 ns / each instruction

④ Waiting time for ~~each~~ instruction no. 3. = 2.5 × 2 = 5 ns

Q2 Using above numerical and interrupt occurs at 4th instruction of the main process, the time slot for this interrupt is 10 ns. If the size of the interrupt is 8 instructions then determine the time required for each instruction of interrupt, waiting time of the main process instruction after the interrupt, waiting time for the main process, total time required to complete the whole process.

Solution



M.P. size = 20 inst.

Time for each inst. = 2.5 ns.

Time for entire process = 50 ns.

size of interrupt = 8 inst.

Time required for entire interrupt = 10 ns.

Time required for each instruction of interrupt = $\frac{10}{8}$ = 1.25 ns

Waiting time of main processor = Time for entire interrupt = 10 ns

Waiting time of 5th instruction of main process => (4 × 2.5) + 10

= 20 ns

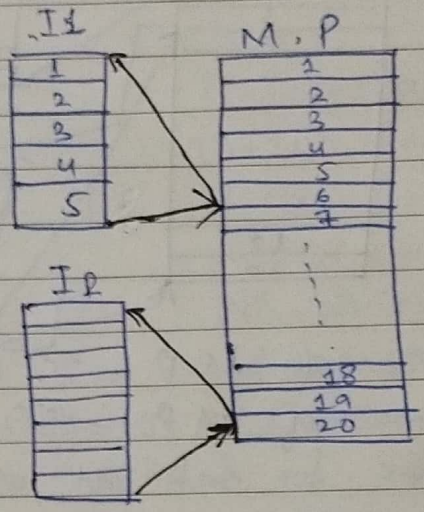
Total time of main process to get completed = 10 + 50

= 60 ns

29/12/22

Q3. Using the same numerical in the main process, two interrupts occur at a time at instruction no. 6. Interrupt 1 can be completed in 10 ns with 1.25 ns for each instruction, interrupt 2 consist of 5 instructions, each instruction requires 1.5 ns. Interrupt 2 is non-maskable.

Solution



M.P size = 20 inst.
T of MP = 2.5 x 20 = 50 ns
T of each inst. = 2.5 ns

Interrupt-2 (I_2)
Size of I_2 = 5 inst.
T. of I_2 = 5 x 1.5 = **7.5 ns**
T. of each inst. of I_2 = 1.5 ns

Waiting time of Inst. 7 = (6 x 2.5) + 7.5 = **22.5 ns**

Interrupt-1 (I_1)
Size of I_1 = 10 / 1.25 = **8 inst.**
T. of I_1 = 10 ns.
T. of each inst of I_1 = 1.25 ns
Waiting time for I_1 = 14 inst x 2.5 = **35 ns**

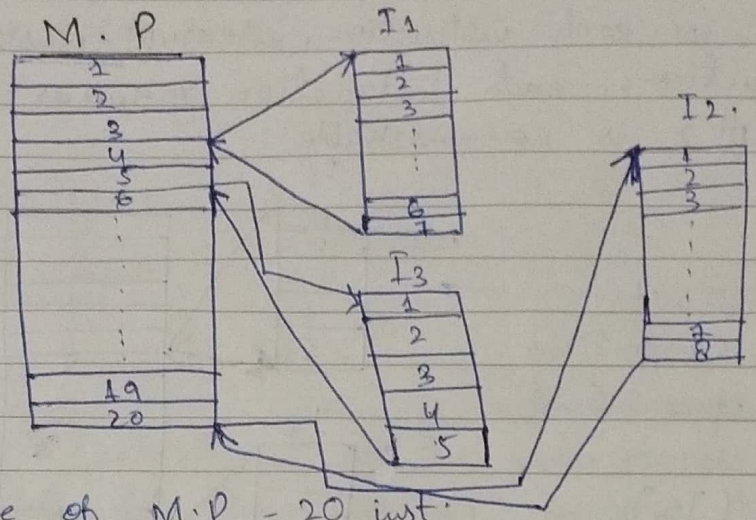
Waiting time of M.P = 10 + 7.5 = **17.5 ns**

Total time to complete the M.P = 50 + 17.5 = **67.5 ns**.

30/12/22

Q.1: A processor processes different jobs in which it execute a process of 20 instructions

Solution



Size of M.P = 20 inst.
Time of M.P = 50 ns.
Time for each inst. of M.P = 20/50 = 2.5 ns

Size of I_1 = 7 inst.
Time for each inst of I_1 = 1.5 ns
Time for I_1 = 7 x 1.5 = 10.5 ns

Size of I_3 = 5 inst.
Time for each inst. of I_3 = 1 ns.
Time for I_3 = 5 x 1 = 5 ns.

Time for I_2 = 8 ns.
Time for each inst. of I_2 = 1 ns.
Size of I_2 = 8 inst.

?? Waiting time of M.P = 10.5 + 5 = 15.5 ns.
most prev. question

Total completing time = 50 + 15.5 + 8 = 73.5 ns

Waiting time for I_2 = (15 x 2.5) + 5 = 42.5 ns.

30/12/22

PIPELINING:

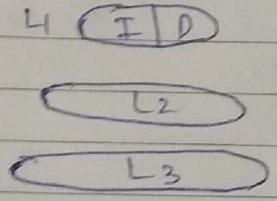
- Series / sequential processing.
- Pipelining increases throughput.
- Dependable on tasks.

∴ Throughput number of tasks executed in a unit time.

	IF	ID	EX	MEM (XX)	WB
I_1					
I_2					
I_3					
I_4					
⋮					
I_n					

∴ IF - Instruction Fetch
∴ ID - Instruction Decode
∴ EX - Execution
∴ MEM - Memory (optional)
∴ WB - Write Back

Every instruction has to pass through all 5 stages, but MEM is optional.



- Instruction Cache
- Data Cache.

$$\text{Speed up } (S) = \frac{\text{Non-pipelined execution time}}{\text{Pipelined execution time}}$$

$$\text{Efficiency } (\eta) = \frac{\text{Speed up}}{\text{No. of stages in pipelined Architecture}}$$

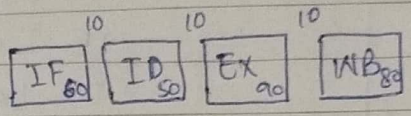
$$\text{Efficiency } (\eta) = \frac{\text{No. of boxes utilized in phase-time diagram}}{\text{Total no. of boxes in phase-time diagram}}$$

$$\text{Throughput} = \frac{\text{No. of instructions executed}}{\text{Total time taken.}}$$

Problem-01.

Consider a pipelined system having four phases/stage with a duration 60, 50, 90 and 80 ns. If the latch delay is 10ns then calculate pipeline cycle time, non-pipeline execution time, speed up ration, pipeline time for 1000 tasks, sequential time for 1000 tasks and throughput.

Solution



① Pipeline cycle time:

$P.C.T = Max(Ph_1, Ph_2, Ph_3, Ph_4) + Delay\ time.$

$= Max(60, 50, 90, 80) + 10.$

$= 90 + 10$

$\boxed{P.C.T = 100\ ns}$

∵ Delay time is constant due to pipelining

② Non-pipeline execution time:

$nP_T = \sum_{i=0}^{N} Ph_N$

$= Ph_1 + Ph_2 + Ph_3 + Ph_4$

$= 60 + 50 + 90 + 80$

$\boxed{nP_T = 280\ ns}$

③ Speed up $= \frac{nP_T}{P_T} = \frac{280}{100} = \boxed{2.8}$

④ Pipeline Time for 1000 tasks.

= Time taken for 1st task + Time taken for remaining 999 tasks.

$= 1 \times 4 \times 100 + 999 \times 100$

$= 100300\ ns$

⑤ Sequential time for 1000 tasks

= Totals task × Time taken for 1 task (^nP_T)

$= 1000 \times 280$

$= 280000\ ns$ ✓

Q 4 instructions are executed in a 4 stages pipelining. Assume that cache instructions requires different stage delay (in terms of number of clock cycles) as mentioned in the given table.

	IF	ID	EX	WB	
I_1	1	3	2	1	→ 7
I_2	2	2	3	1	→ 8
I_3	1	1	1	1	→ 4
I_4	2	1	1	2	→ 6.

How many clock cycles will be required to complete these 4 instructions in the given pipelining system.

	t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10} t_{11} t_{12} t_{13} t_{14} t_{15} t_{16} t_{17}
IF	I_1 I_2 I_2 I_3 I_4 I_4
ID	I_1 I_1 I_1 I_3 I_2 I_2 I_4
EX	I_1 I_1 I_2 I_2 I_2 I_3 I_4
WB	I_1 I_2 I_3 I_4 I_4

P_T = 13 cycles.

nP_T = 7+8+4+6 = 25 cycles

speed up = $\frac{nP_T}{P_T} = \frac{25}{13} = 1.99$

(6) Throughput = $\frac{\text{No. of instructions executed}}{\text{Total time taken}}$

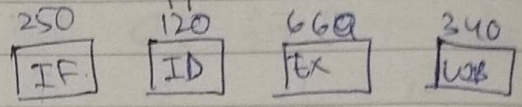
$= 4 / 100300$

≈ 0.00997

5/01/23

Q) A 4 stage pipeline system has the stage delays as 250, 120, 660, 340 ns respectively. Registers are used between the stages and have a delay of 5ns each. Assuming constant clock rate the total time taken to process 1000 data items on the pipeline will be

---



Solution

① Pipeline cycle time

$P.C.T = Max(250, 120, 660, 340) + 5$

$P.C.T = 660 + 5$

$P.C.T = 665$ ns.

② Non-pipeline execution time.

$$nP_T = \sum_{i=0}^{N} Ph_N$$

$nP_T = 250 + 120 + 660 + 340.$

$nP_T = 1370$ ns.

Time taken for 1st data item + Time taken for 999 Data Items.

$1 \times 4 \times 665 + 999 \times 665$

$2660 + 664335$

Time taken to process 1000 data items. 666995 ns.

$$\text{Speed up} = \frac{nP_T}{P_T} = \frac{1370}{665}$$ ✓

Assignment no. 2. 5/01/23

Q①: The stage delays in a 4-stage pipeline are 800, 500, 400, 300 ps (pico seconds). The first stage is replaced with a functionality equivalent design involving two stages with respective delays 600 and 350 ps. The throughput increase of the pipeline is 33.33 %.

Solution :-

Q②: A non-pipelined single cycle processor operating at 100 MHz is converted into synchronous pipelined processor with 5 stages requiring 2.5 ns, 1.5 ns, 2 ns, 1.5 ns and 2.5 ns respectively. The delay of the latches is 0.5 seconds. The speed up of the pipeline processor for the large number of instruction is 3.33

Q Consider a 4-stage pipeline processor. The number of cycles needed by the 4 instructions I_1, I_2, I_3, I_4 in stages S_1, S_2, S_3, S_4 is shown below.

	S_1	S_2	S_3	S_4
I_1	2	1	1	1
I_2	1	3	2	2
I_3	2	1	1	3
I_4	1	2	2	3

what is the number of cycles needed to execute the following loop?

for (i=1 to 2) {I_1, I_2, I_3, I_4}.

Solution

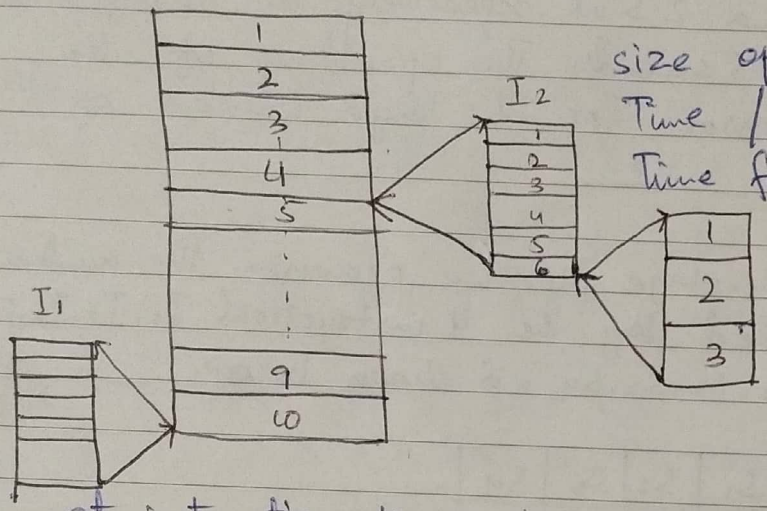
	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}
S_1															
S_2															
S_3															
S_4															

	t_1	t_2	t_3	t_4	t_5	t_6	t_7 *	t_8	t_9	t_{10} *	t_{11}	t_{12} *	t_{13}	t_{14}	t_{15}	t_{16} *	t_{17}	t_{18}	t_{19}
S_1	I_1	I_1	I_2	I_3	I_3	I_4	I_1	I_1	I_2	I_3	I_3	I_4							
S_2			I_1	I_2	I_2	I_2	I_3	I_4	I_4	I_1	I_2	I_2	I_2	I_3	I_4	I_4			
S_3				I_1			I_2	I_2	I_3	I_4	I_1	I_1		I_2	I_2	I_3	I_4	I_4	
S_4					I_1				I_2	I_2	I_3	I_3	I_3	I_4	I_4	I_1	I_2	I_2	I_3

	t_{20}	t_{21}	t_{22}	t_{23}
S_4	I_3	I_3	I_4	I_4

P_T = 23 cycles.

MP



6/01/23

size of MP = 10 inst.
Time / inst = 2.5ns
Time for M.P = 10 × 2.5 = 25ns

at instruction 4 we have 2 interrupts - Interrupt 2 is non-maskable. based on 6 instructions - The total time required to process this interrupt is 8ns. Interrupt 1 requires 4ns to process. Each instruction may be executed with 1.5 ns - When we were attempting interrupt 2, interrupt 3 occurs with 3 instructions of 6ns whole.

I_2
size of I_2 = 6 inst.
Total time for I_2 = 8ns.
Time for each inst = $\frac{8}{6} = \frac{4}{3}$ ns = 1.3 ns
waiting time for I_2 = 0ns

I_3 size of I_3 = 3 inst
Total time for I_3 = 6ns
Time for each inst = 6/3 = 2ns
waiting time for I_3 = 0ns since we don't know or undefined when did it happen

Total time to complete I_2 = 8+6 = 14ns

Wait Time for inst 7 = (6×2.5) + 8+6 = 29 ns

Total time for I_1 = 4ns
Time for each inst = 1.5ns
Size of I_1 = 4/1.5 = 2.6ns ≈ 3ns
Waiting time for I_1 = (6×2.5) + 8+6 = 29ns

Waiting time of M.P = 14 ns
Total time of completion of M.P = 25 + 8+6+4 = 43ns

Pipelining Problem:

Q. Consider a pipeline processor with 4 stages - IF, ID, EX and WB. If the IF and ID and WB stages take 1 clock cycle each to complete the operation. The no. of clock cycle for the execution stage EX depends on the instruction. The ADD and SUB instruction needs 1 clock cycle and the MUL instruction needs 3 clock cycle in the execution stage, operand forwarding is used in the pipelined processor. What is the number of clock cycle is taken to complete the following sequence of instructions?

ADD
MUL
SUB

Solution

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
IF	ADD	MUL	SUB							
ID		ADD	MUL	SUB						
EX			ADD	MUL	MUL	MUL	SUB			
WB				ADD	–	–	MUL	SUB		

Total clock cycles = P_T = 8
Non-pipeline = nP_T = 14
Speed up = $\frac{P_T}{nP_T} = \frac{8}{14} = 1.75$

PIPELINE HAZARDS:

- Structural Hazards → Architectural
- Data Hazards → Cannot save multiple instructions on single data
- Control Hazards → empty space between instructions.

STRUCTURAL:-

- If MIPS had only one memory was used to.
  - Fetch instruction
  - Load / Store data

DATA DEPENDENCE & HAZARDS:

a = 2+1
b = a+1
data dependibility

Data Dependencies
Add S0, T0 T1 (write in S0 in cycle 5)
Sub T2, S0, T3 (we S0 as input in cycle 5)
data is dependent, solution is to insert nopes / bubble so on that cycle processor is unused for that instruction, however it is time taking but it is a solution.

19/01/2023

$T_n = n * t_n$
$T_k = (k * n-1)\, t_p$.

T_n = Total time / clock cycle for non-pipelined
T_p = Total time for pipeline
k = no. of stages
n = no. of tasks
t_p = time / clock cycle for pipeline.
t_n = Time required to complete tasks.

Q① Determine the no. of clock cycles that it takes to process 200 tasks in a six segment pipeline.

Solve

k = 6
n = 200
no. of clock cycles = ?

no. of clock cycles $\Rightarrow T_p = (6 + 200 - 1)$ clock cycles.
$T_p = 205$ clock cycles.

Q②: A non-pipelined system takes 50ns to process a task. The same task can be processed in a six segment pipeline with a clock cycle of 10ns. Determine the speed up ratio of a pipeline for 100 tasks. What is the maximum speed up that can be achieved?

Mid marks > 21.48

Solve

$t_n = 50ns$.
$k = 6$
$t_p = 10ns$.
$n = 100$

Speed up = Non-pipelined exec. time / Pipelined exec. time

$T_n = 100 \times 5 = 500ns$.
$T_p = (6 + 100 - 1)\ 10 = 1050$.
Speed up = $\frac{5000}{1050} = 4.76$.

20/01/2023

ILP → Instruction Level Parallelism : Khud pathao.
Multiple operation / instructions can be executed in parallel.
2 approaches to exploit ILP.
→ Hardware approach → dynamically.
→ Software approach → statically at compile time.

① Hardware approach: An approach that relies on hardware to help discover or exploid parallelism dynamically.

② Software approach: An approach that relies on software technology to find parallelism, statically at compiletime.
· software is hardware based

① Data Dependency — Data Dependence.
② Name Dependency
③ Control Dependency

How to measure dependency? in instructions?
→ Memory locations → Registers

PAPERWORK