COA CU Instruction Pipelining

one of the reason why processor works fast.

COA → Computer Organization Architechture

CU → Control Unit (Processor এর একটা unit)

↳ Fetch & Decode

Q: Why processor becomes so fast?

→ Increase data bus size (Now 64 bit is used)

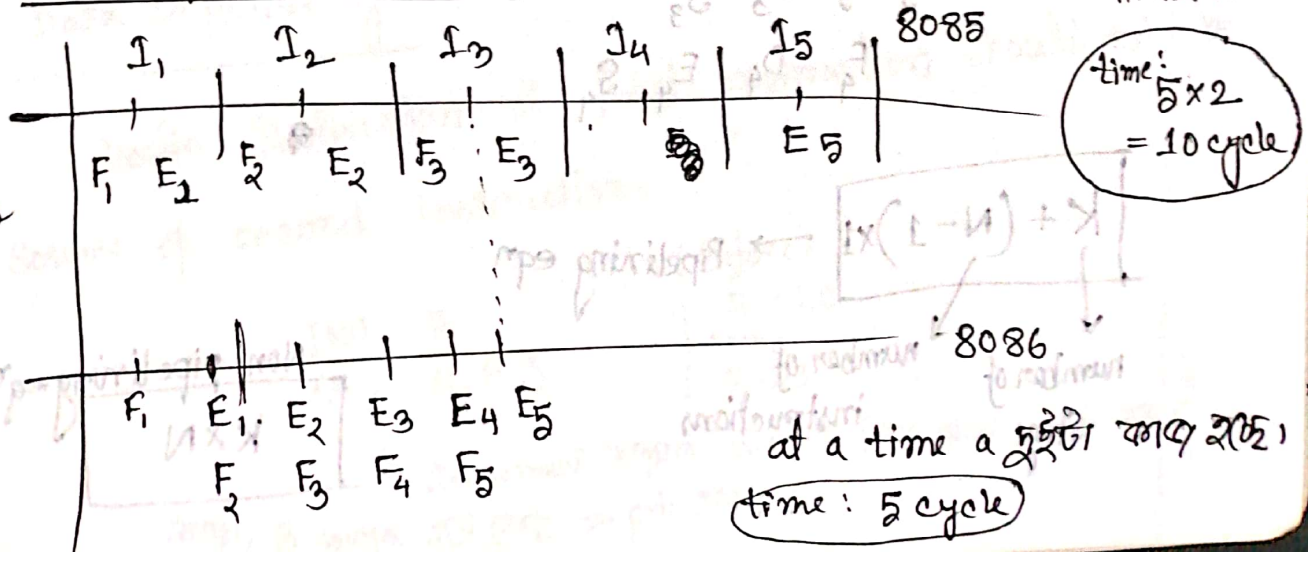
→ Increase clock frequency / Trigger

→ ~~Pipeling~~ Pipelining

8085 → Non pipelined

8086 → Pipelined

2 stage Pipelining:



Decode খুবই কম্ time নেয় so লিখি নি।

time: 5 × 2 = 10 cycle

at a time a দুইটা কাজ হচ্ছে।

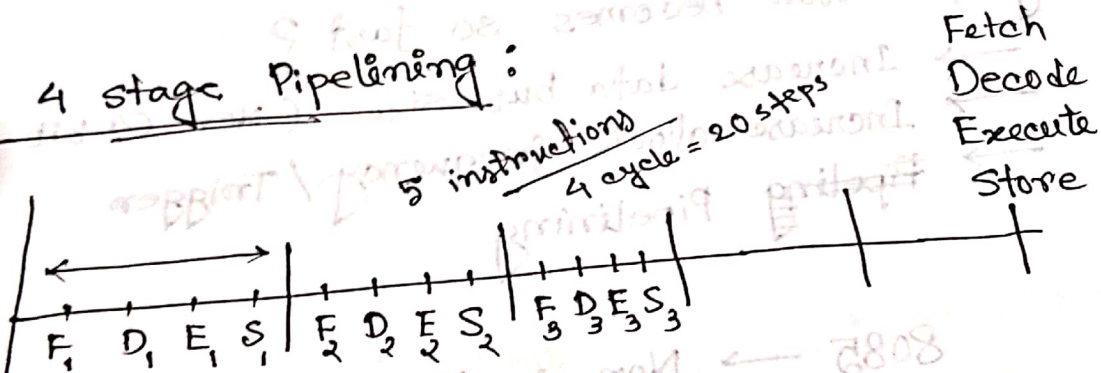
time: 5 cycle

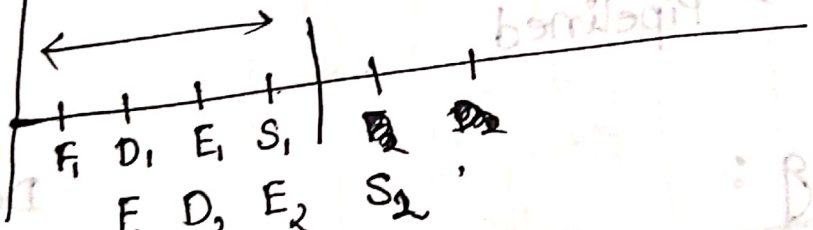
Week-6
Lec-(14)
Subject
Date 28.05.22 Sat

As we are fatching two main process of an instructions & working on the two instructions at the same time, so it becomes fast, this process is called pipelining.

Q: figure of pipelining and non-pipelining. (exam imp)

4 stage Pipelining:





F2 D2 E2 S2
F3 D3 E3 S3
F4 D4 E4 S4

Pipelined.

$K + (N-1) \times 1$ → Pipelining eqn

Non pipelining eqn

$K \times N$

number of instructions

number of

Subject .....
Date ..... Time .....

Advantage & Disadvantage of 4-stage

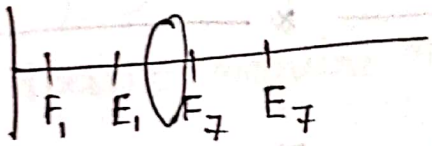
↓ fast

↓
1) Control Hazard (Broach Prediction)
2) Data Dependency
3) Structural hazard

Control Hazard :

branching

A spam or empty spot was created which is called Pipelin Bubble.



এভাবে pipeline failed

By Branch prediction algorithm it was overcome. It 95 accurac

If stages increased & discard increases.

* Data Dependency :

~~Destin~~ Destination of first instruction should not be Source of second instruction.

INC B
MOV A, B

Before,
B = 20
now,
B = 21

↳ Increment হওয়ার আগেই নিয়ে নিলে prb হবে।

এখানে B ... so prb হয়ে গেছে।

Subject.....
Date:.....Time:.....

Advantage & Disadvantage of 4-stage

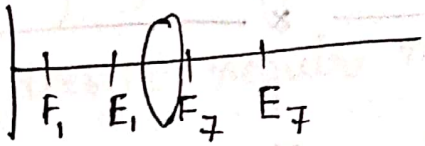
↓ fast

↓
1) Control Hazard (Broach Prediction)
2) Data Dependency
3) Structural hazard

Control Hazard :

branching

A spam or empty spot was created which is called Pipelin Bubble.



এভাবে pipeline failed

By Branch prediction algorithm it was overcome. It 95% accuracy

If stages increased & discard increases.

* Data Dependency :

~~Destin~~ Destination of first instruction should not be Source of second instruction.

```
INC  B
MOV  A, B
```

Before, B = 20
now, B = 21

↳ Increment হওয়ার আগেই নিয়ে নিলে prb হবে।

আবার ... prb হয়ে যাবে।

Solution :

```
INC  B
NOP
MOV A, B
```

NOP → No operation

কাজ করে,
fetch, execution
operation; prb
হয় না।

* Structural Hazard :

```
ADD  A, B
ADD  A, [2000]
```

to solve this structural hazard

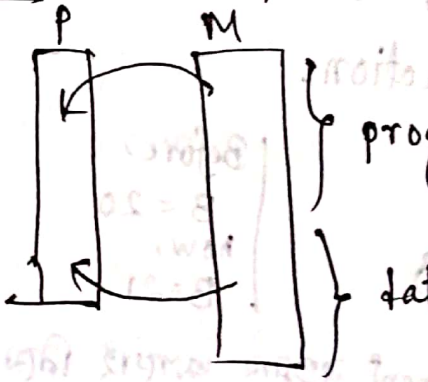
1. RISC Processor → full form
2. PIC "

Characteristics :

RISC has more register. It lessen the dependency on memory.

PIC → Peripheral Inter Controller

Before



Solution : IN B
NOP
MOV A, B

NOP → No operation
কোনো কাজ করে না, fetch, execution operation-এ problem হয় না।

* Structural Hazard :

A ADD A, B
ADD A, [2000]

solve this structural hazard

---

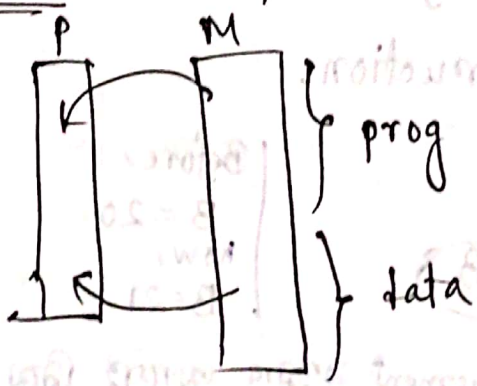
1. RISC Processor → full form
2. PIC "

Characteristics :

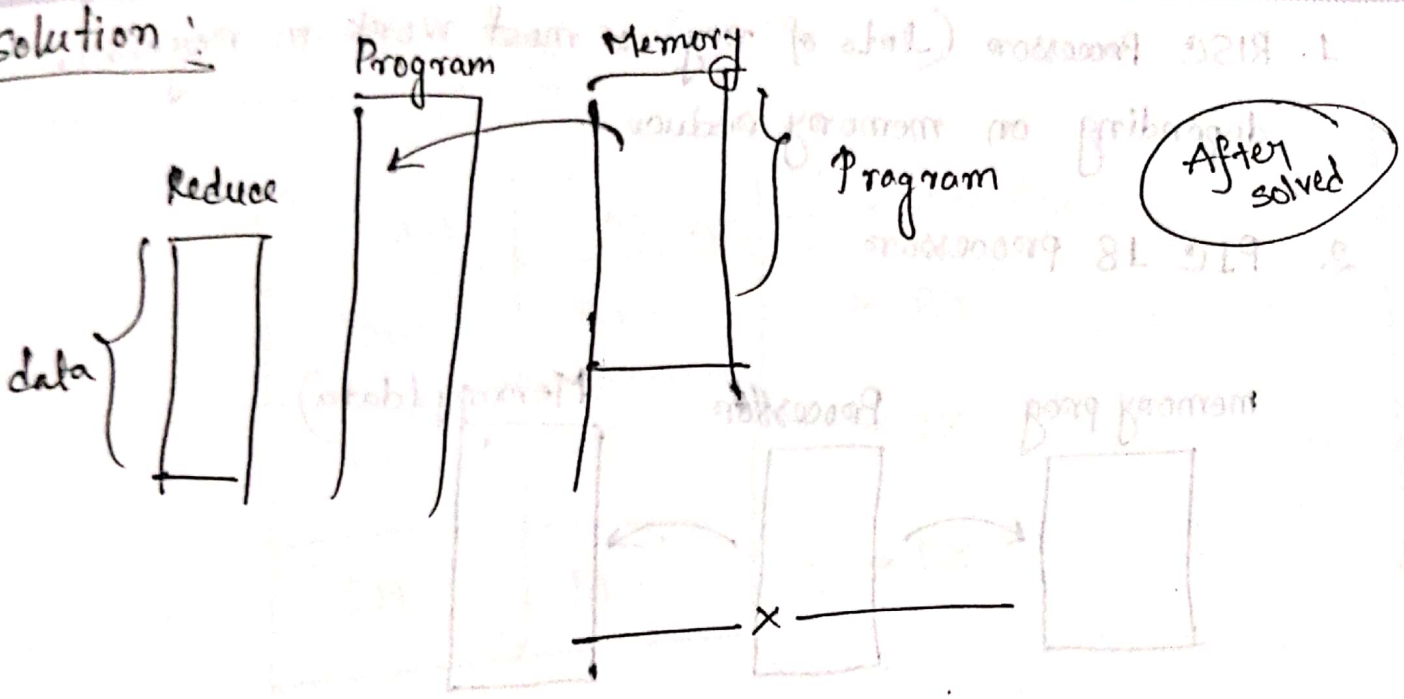
RISC has more register. It lessen the dependency on memory.

PIC → Peripheral Inter Controller

Before



Solution :



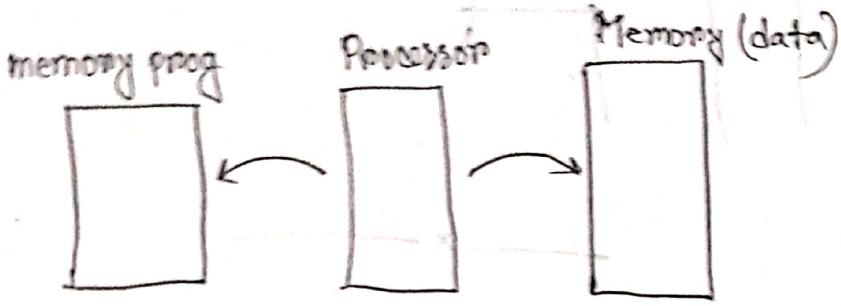
Overlapping of fetching & execution can only happen if execution doesn't require memory.

Pipelining is imp as it helped to increased the speed of Microprocessor.

Subject: Lec-15
Date: W-6
30.05.22

1. RISC Processor (lots of reg) so most work in register, depending on memory reduce.

2. PIC 18 processor



two separated memory for program & data while executing if we need data use other memory, so fetching and executing never clash with each other.

Addressing Mode

Addressing mode is the manner by which an operand is given instruction is called.

operand: the number on which the operation is done is called operand.

Add, AL, 25H

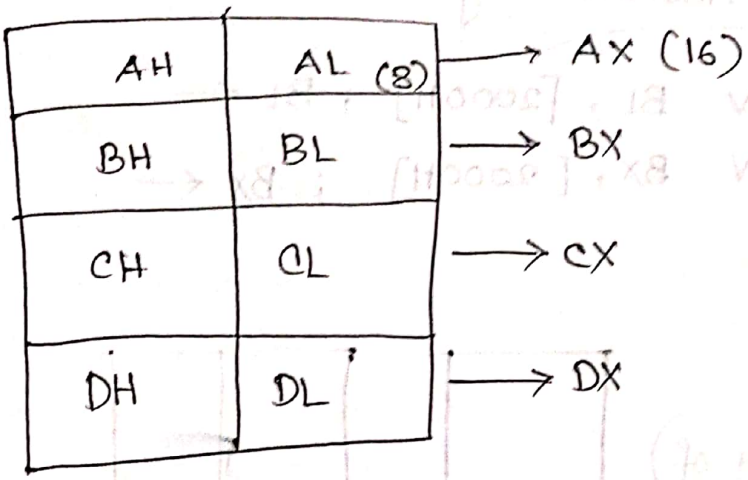
operands

Operation → SUM, MOV, ADD

- Data from imidiate
  Add AL, 25H
- Register
  Add AL, BL
- Indirect/memory location
  Add AL, [BL]

Subject.....
Date:..... Time:.....

General purpose register :



25 H → 0010 0101

1235 H → 0001 0011 0011 0101

① Immediate Addressing Mode :

Data is given in instruction

[To initialize]

MOV BL, 25H ; BL ← 25H

MOV BL, 2000H ; BX ← 2000H

যেখানে instruction এর সাথেই data লেখা থাকে so it can go to immediate operation.

② Register Addressing Mode :

Data is stored in Reg.

MOV BL, CL ; BL ← CL

MOV BX, CX ; BX ← CX

processor এর ভিতর কাজ হচ্ছে । It's smaller & faster.

Q: কোন addressing mode কখন use করবে?

Subject.....

Date:.....Time:.....

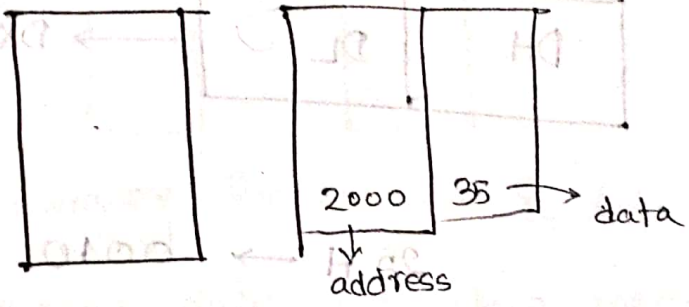
(III) Direct Addressing Memory:

Address is given in instruction

```
MOV  BL, [2000H] ; BL ←
MOV  BX, [2000H] ; BX ←
```

It's an address so 16 bit. -though the Reg is 8 bit.

(BL gets the content of 2000H)



don't

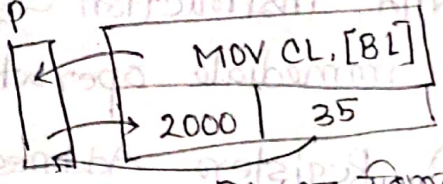
* When we know the data, we use this mode.

(IV) Indirect Addressing MODE: (BEST BEST)

Address is given in the register.

1) Register Indirect: addr. in register

```
MOV CL, [BL]
```



BL এর ভিতর add.

2)

best way because it's easier to increament & decremont.

Subject.....

Date:.....Time:.....

* ADD 25 + 35

```
MOV AL, 25H
MOV BL, 35H
ADD AL, BL
```

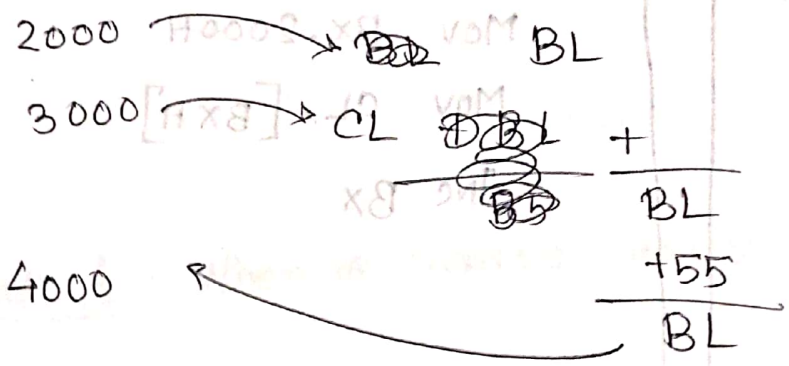
← Reg A.M

* [2000H] , [3000H]

```
MOV AL, [2000H]
MOV BL, [3000H]
Add AL, BL
```

→ Direct Add. M

*




```
MOV BL, [2000H]
MOV CL, [3000H]
ADD BL, CL
ADD BL, 55H
MOV [4000H], BL
```

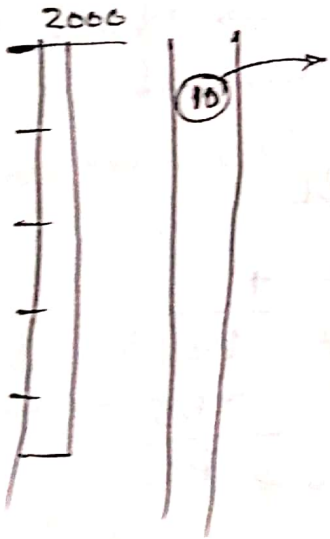
→ It's also a direct addressing mode as address is given in the ins.

Sat
Subject: Week - 7
Date: Lec - 16 Time:
04.06.22

* Operation Destination, Source
↓
MOV AX, BX

Indirect Addressing Mode :

immediate : initialize
register : move between reg
direct : need to access few memory location



MOV CL, [2000H]
MOV BX, 2000H
MOV CL, [BXH]
INC BX

Give next 100 data from 2000 → Indirect
Give the data of 2000 memory location → Direct

Q: get data from location 2000 in CL.

Q: get data of hundred location starting from 2000.

Subject.....

Date:.....Time:.....

Indirect Addressing Mode :

i) Reg indirect : add in reg, MOV CL, [BX]

ii) reg relative : adder = adder + displacement

MOV CL, [BX + 02H]

iii) Base indexed : ~~adder = MOV CL, [BX + 02H]~~

adder = base + index reg

MOV CL, [BX + Si]

iv) Base relative plus indexed : add = base + index + disp

MOV CL, [BX + Si + 03H]

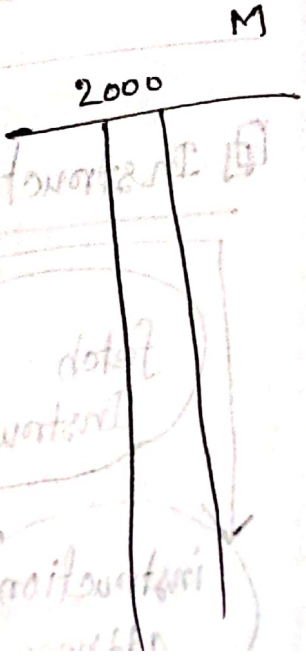
— x —

Indirect : When we need to access a bulk amount of data.

SI = Source Index Register

MOV CL, [BX + 50H]

MOV CL, [BX - 1100H]

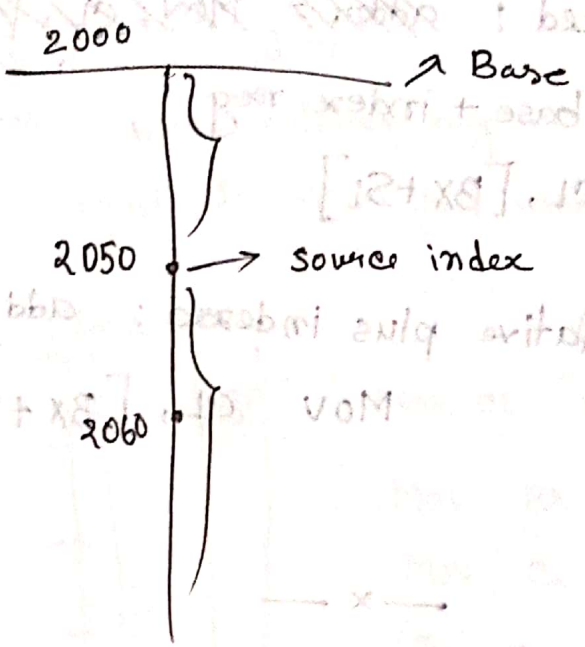


Subject:
Date: Time:

Base indexed :

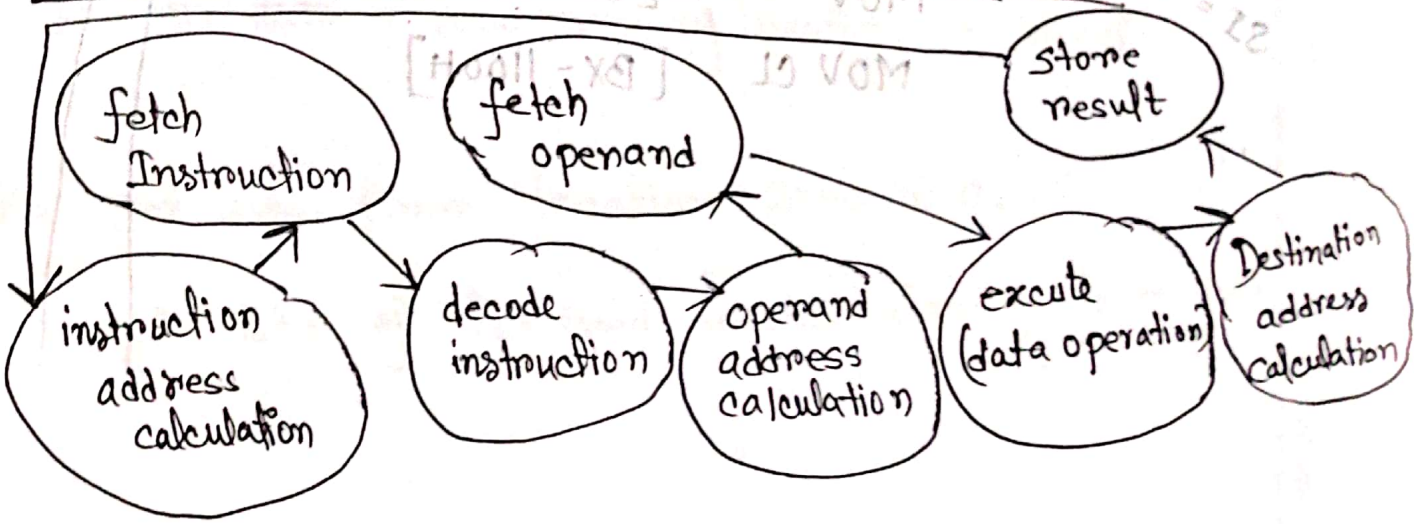
i) one register act as a base

ii) other reg act an an index



Indirect Addressing Mode :

Instruction cycle State: transition diagram :



Sweet Bird

```
int main()
{ cin >> a >> b;
  c = a+b
  cout << " done"; }
```

| a = 2
| b = 3

O/P :
done.

—x—

program → set of instruction

* To execute these program, tiny operations are need that are called Micro operations.
(each instructions requires tiny tiny operations)

* Control unit control signal generate করে।

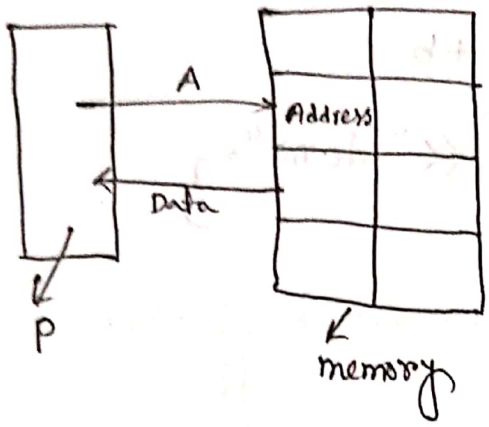
Q: একে micro operation কেন বলে?

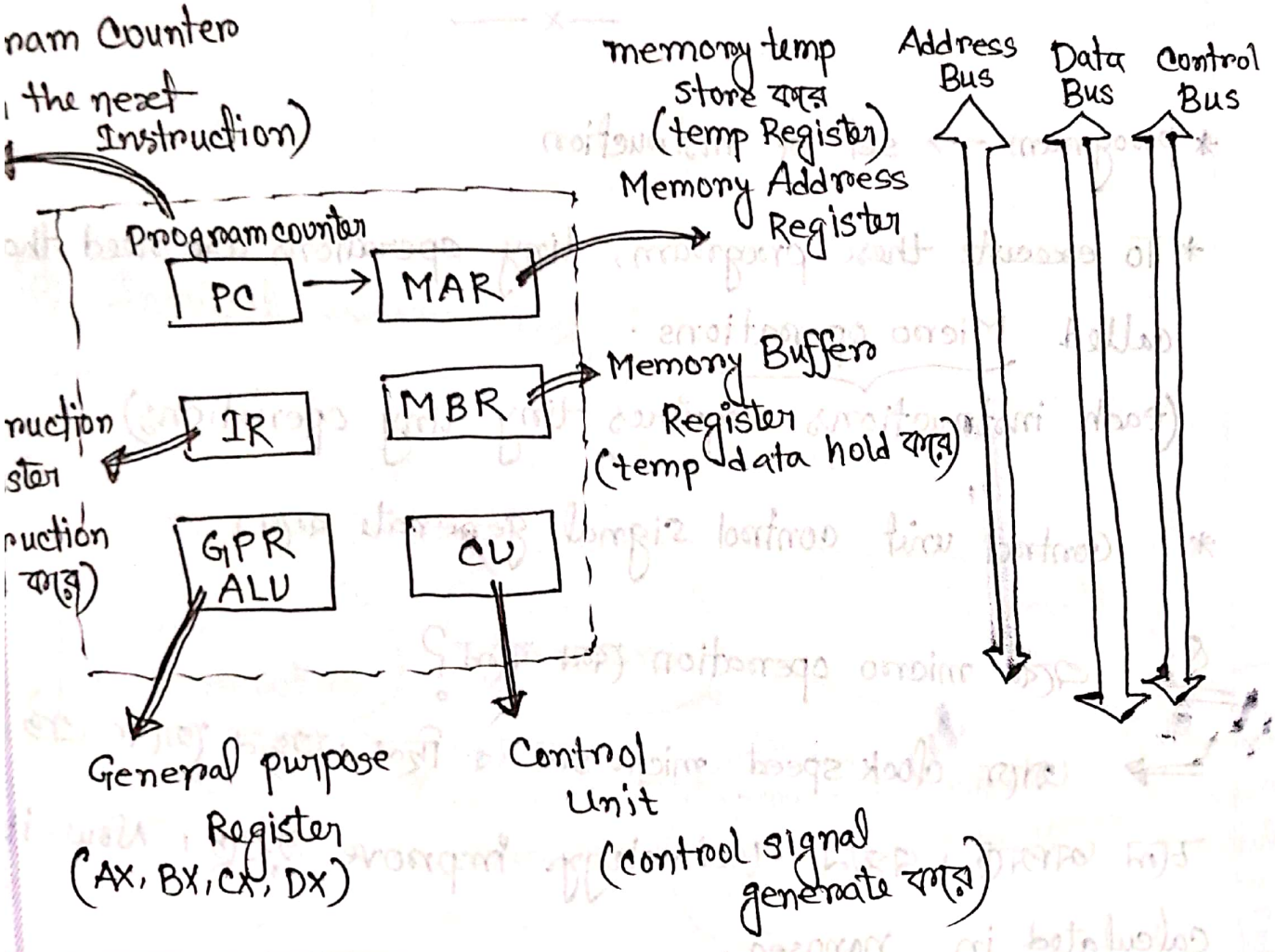
⟹ আগে clock speed micro sec এ ছিল, তখন থেকে এই নাম চলে আসছে। এখন technology improve হয়েছে। Now it's calculated in nanosec.

Subject: .....

Date: ..... Time: .....

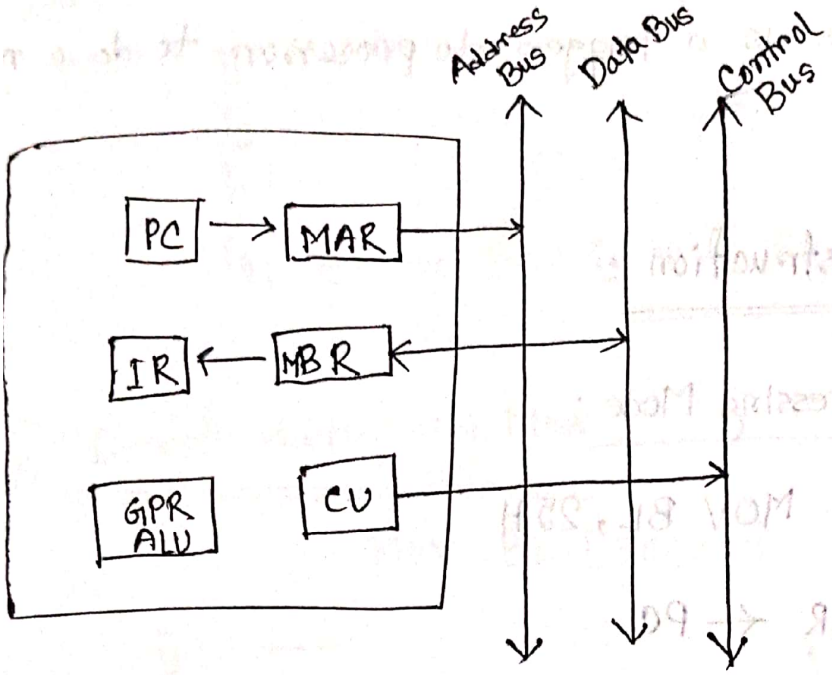
Sytem Bus → Address Bus
→ Data Bus
→ Control Bus



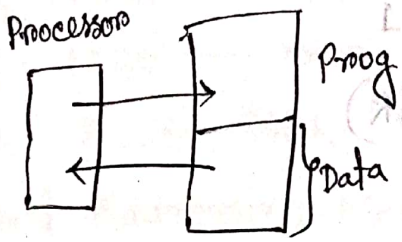


Imp Topic for Marks 4.

Control Unit (micro operations)



Q: What is μ-operations?



Steps of Micro(μ) operations:

* μ-operation for fetching:

T_1 : MAR ← PC

T_2 : MBR ← mem (instruction)

T_3 : IR ← MBR
PC ← PC + 1

→ sequencially execute করানোর জন্য।
→ Independent তাই 4টা না করে 3টা Transaction হবে।

Subject.....
Date:.....Time:.....

clock pulse দ্বারা নিয়ন্ত্রিত transaction controlled হয়।

* each clock pulse is a trigger to processor to do a new activity.

μ-operation instruction :

(1) Immediate Addressing Mode :

MOV BL, 25H

T_1 : MAR ← PC

T_2 : MBR ← mem (inst)

T_3 : IR ← MBR
PC ← PC+1

T_4 : BL ← 25H (IR)

(2) Reg Addressing Mode :

MOV BL, CL

← same as before

T_1 : —

T_2 : —

T_3 : —

T_4 : BL ← CL

↑ data given in register.

Subject.....

Date:.....Time:.....

Q: Add BL, 25 H ?

⟹ T_1 :

T_2 :

T_3 ;

T_4 : BL → BL + 25H (IR)

3) Direct Addressing Mode :

MOV BL, [5000 H]

← address instruction এ দেয়া থাকে।

T_1 : —

T_2 : —

T_3 : —

T_4 : MAR ← IR (address)

T_5 : MBR ← mem

T_6 : BL ← MBR [5000]

Q: ADD reg, [add] ?

T_1 :

T_2 :

T_3 :

T_4 : MAR ← (address) (IR)

T_5 : MBR ← mem (data)

T_6 : Reg ← Reg + MBR

4) Indirect Addressing Mode :

MOV BL, [CX]

← address register এ দেয়া থাকে।

T_1 : —

T_2 : —

T_3 : —

T_4 : MAR ← CX

T_5 : MBR ← mem ~~data [BX]~~

T_6 : BL ← MBR ~~[BX]~~

ADD, SUB etc.

Subject .....
Date: ..... Time: .....

(5) Implied Addressing Mode :

carry flag 1 করতে হবে,

T_1 :

T_2 :

T_3 :

T_4 : CF ← 1

E_p : STC

↓

set the carry flag

~~datafloo~~

—— × ——

CT-2

Booth's Algo

Booth's Multiplication, Division

~~Pipelinging~~

Adder, Substractor

Lec - 5 - 10 , Lec - 13, 14

Mid Break starts Here ....

Yayyy ☺

After Mid.

Subject: Sat 18.06.21
Date: lec-19 Time: W-8

* cache : lies betn processor and memory.

Cache Memory Mapping Technique

1) fully associative mapping:

Slow

95% ratio (hit)
cache বড় হলে — কাজকর্ম ভালো

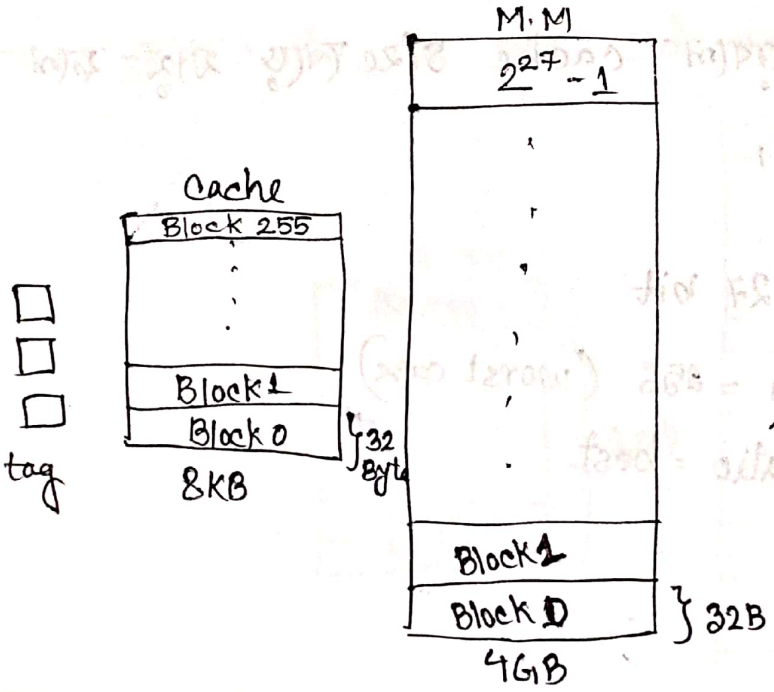
The process how block from main memory is placed on cache ⟶ called cache memory mapping.

* hit ratio:

processor copies a block of data into cache while fetching, that's how hit ratio rise.

hit ratio বেশি হলে ভালো।

example of pentium processor;



cache memory 8KB
Main memory 4GB
↳ pentium

exam

Size of Block = 32 bytes

No. of B. in MM = $\frac{4GB}{32 \text{ bytes}}$

$= \frac{2^{32}}{2^5}$

$= 2^{27}$

No. of Block in cache = $\frac{8\,KB}{32\,byte}$

$= \frac{2^{13}}{2^5} = 2^8$

$= 256$

flexible method & slow

tag : main memory থেকে কোন block cache রাখা হয়েছে।
tag will indicate which block from main memory is present in cache.

number changed (3) বসানো লাগে

tag লাগে 27 bit for pentium.

(27) → or 2^{27} পর্যন্ত রাখা লাগে

cache directory

↓

expensive

এই method use করলে cache size বেড়ে যায়- ফলে expense বেড়ে যাচ্ছে।

tag = 27 bit

Search = 256 (worst case)

hit ratio = best

27bit	5bit
Block num	Location num

= 32 bit

⑤ ☐
☐
☐ tag

8|3 ——— tag

8 no. block এর 3 no. location

এই location catch এর কত number Block সে "

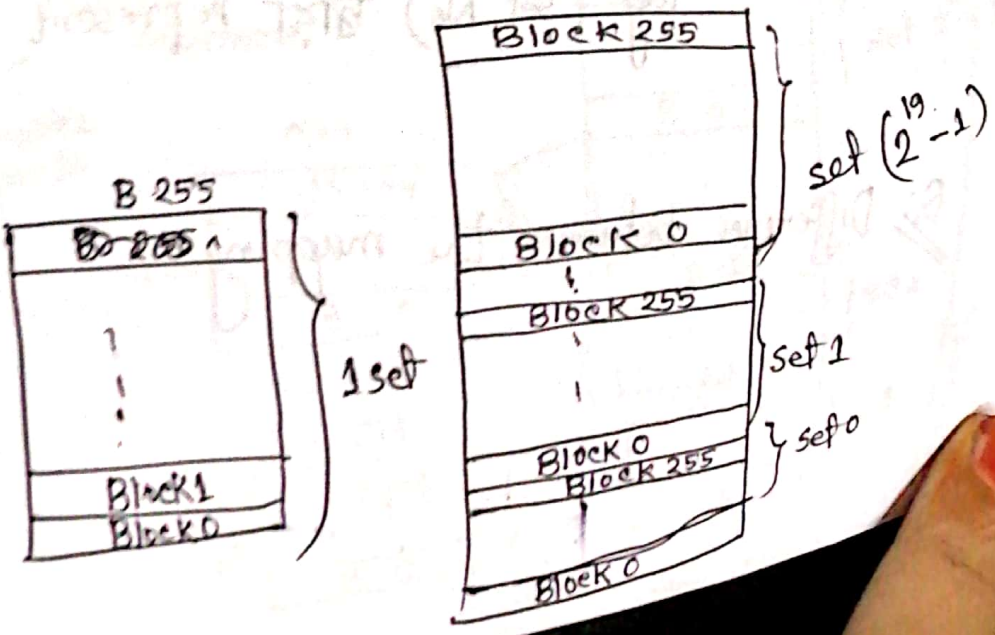
Q: cache location ? = 3|3 (example থেকে)

Q: drawback ? → slow, 256 বার search করতে লাগবে "

② One way associtive Mapping :

just একটা way থাকবে "

এখানে cache কে একটা set consider করি



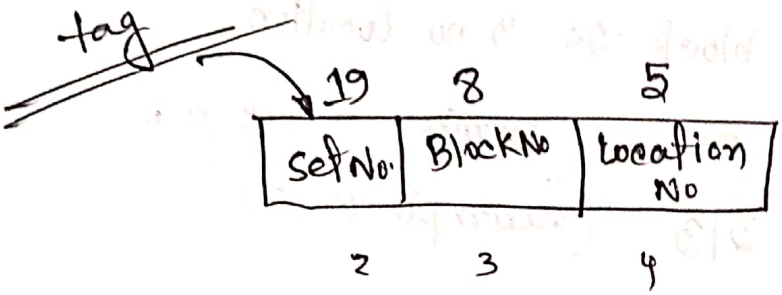
size of set = 8 kB

No. of set in M.M = $\frac{4\,GB}{8\,kB}$

$= \frac{2^{32}}{2^{13}}$

$= 2^{19}$

unflexible
rigid
fast



tag = 19 bit

Search = 1

hit ratio = worst

worst case senario : same block ↺ alternatively use

repeated use / alternative use

tag (set No) দ্বারা represent করা হয়।

Q: Difference between the mapping.

Subject: Lec-20, W-3
Date: Mon Time: 20.06.22

(3) Two way associative method :

It devides the cache into two sets.

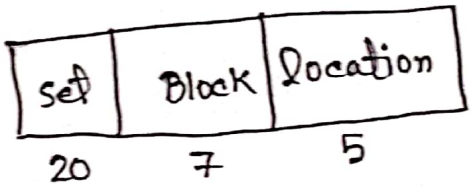
No. of set in M.M = $\frac{4GB}{4KB} = \frac{2^{32}}{2^{12}} = 2^{20}$

set size = 4KB

M.M = 4GB

No. of Blocks in cache/set = $128 = 2^7$

size of block = 32 bytes = 2^5



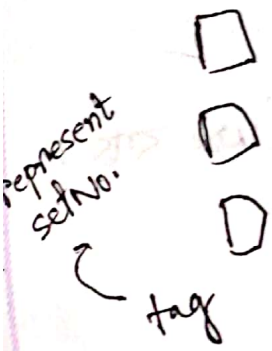
tag = 20

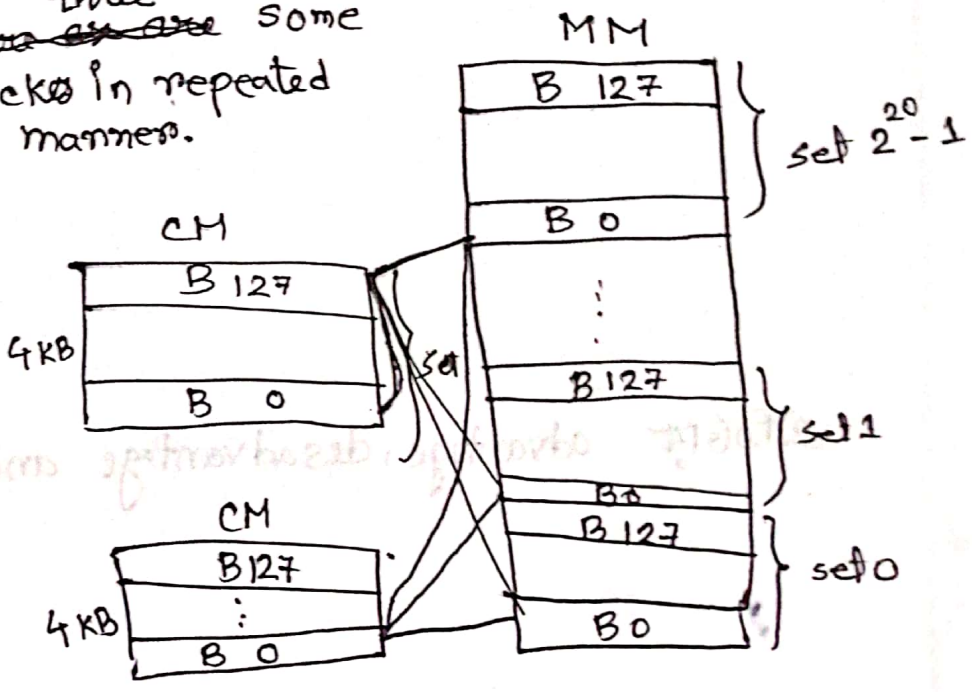
search = 2

Hit ratio = way better

worst case : three some blocks in repeated manner.

two solve the prob, we can devide cache in some parts.





Memory Hierarchy

Primary memory / physical / main mem : (original)

RAM → Constant power supply voltage
ROM → ×, non-volatile

} chip memory, expensive, fast

Secondary Memory : (to increase storage capacity)

HD →
FD (floppy disk) } magnetic disk
CD →
DVD → } optional disk

} portable, non-volatile, cheap, need larger power

portable

HD, FD, CD, DVD, pendrive (chip memory)

data acquiring rate =

rate of disk rotation (rpm)

(fast + cheap) → secondary memory

↓ for operation

↓ for storage

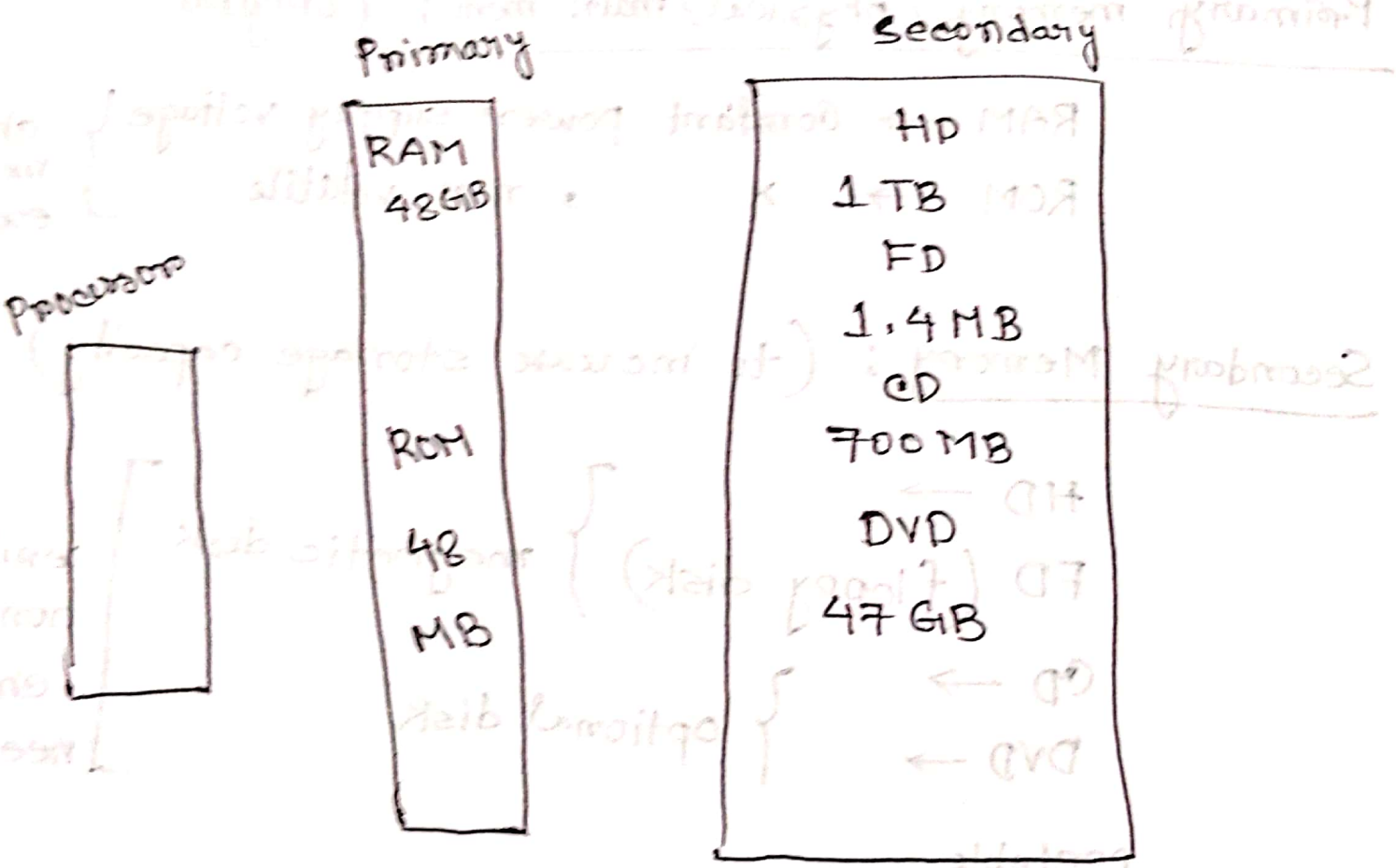
Q: How speed increase of computer by increasing RAM size?

i.e. RAM occupied হয়ে গেলো, speed কমে যায়, RAM বাড়ালে space বাড়ে, so speed বাড়ে।

Subject:

Date: Time:

Memory Hierarchy



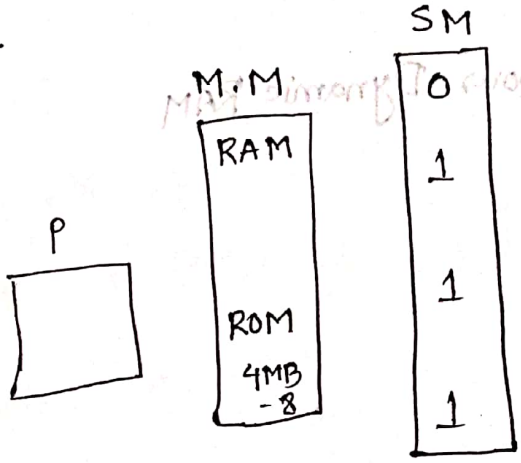
Mon
Subject: dec-21
Date: 27.06.22 Time:
LW-9

Memory Hierarchy

pen-drive (flash ROM) - writtable & non-volatile

SSD → Solid state disk → No Rotation

ROM :



Viva

BIOS এর কাজ কী?

* BIOS থাকে ROM এ।

BIOS → Basic Input Output System → Booting Progra

OP Run → load in RAM → a program fetch OP in RAM

→ Booting Program (BIOS) → ROM (Primary)

SRAM Static RAM	DRAM Dynamic RAM
Speed fast	slow
cost high	low
used in cache	M.M
power consumption low	high
use flip flop	use capacitor

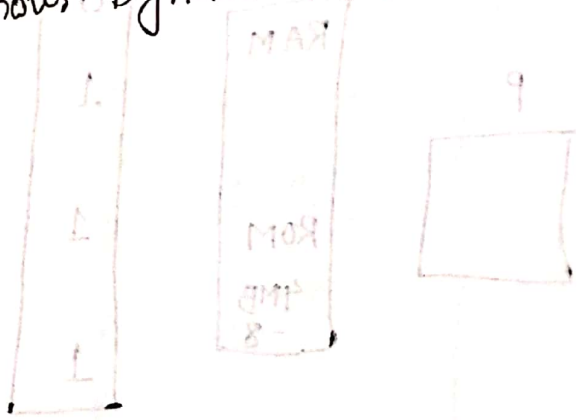
RAM

DRAM

SRAM

SDRAM

Synchronous Dynamic RAM



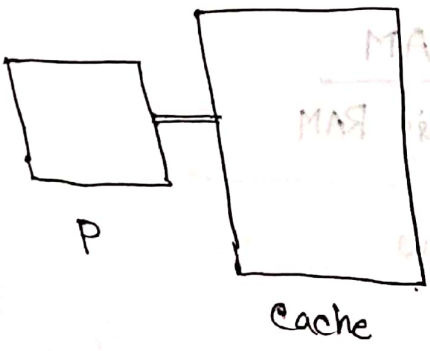
Processor

Cache

Main Memory

2nd Memory

Bus slow তাই দ্রুত information আদান প্রদান possible না।



Pentium
1993

→ এই prob solve করতে এটা আনা হয়।
এটা fast. কেন??