COA · CU Instruction Pipelining

CU - control unit → fetch, Decode

Q. Why processors becomes so fast?

→ increase size of data bus

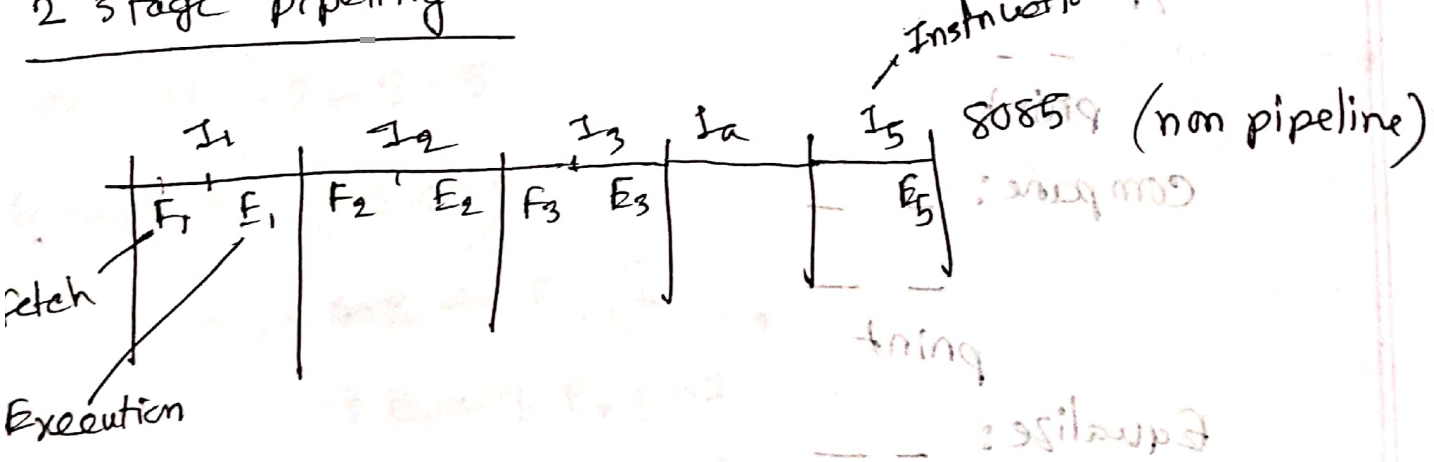
→ ↑ clock frequency

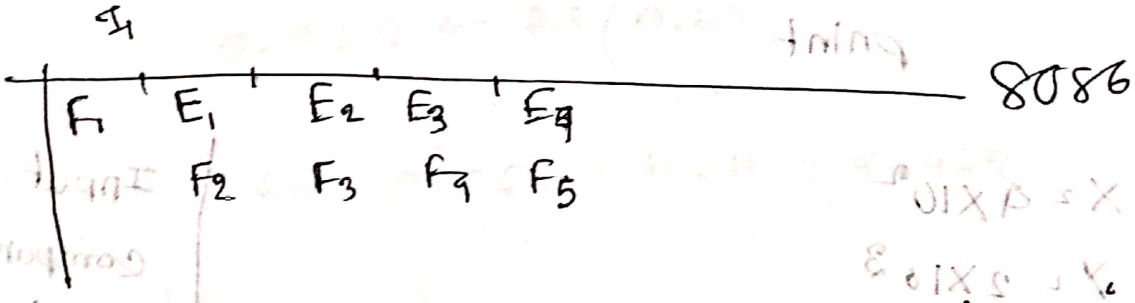
→ Pipelining

8085 - Non pipelined

8086 - Pipelined

2 stage pipeling





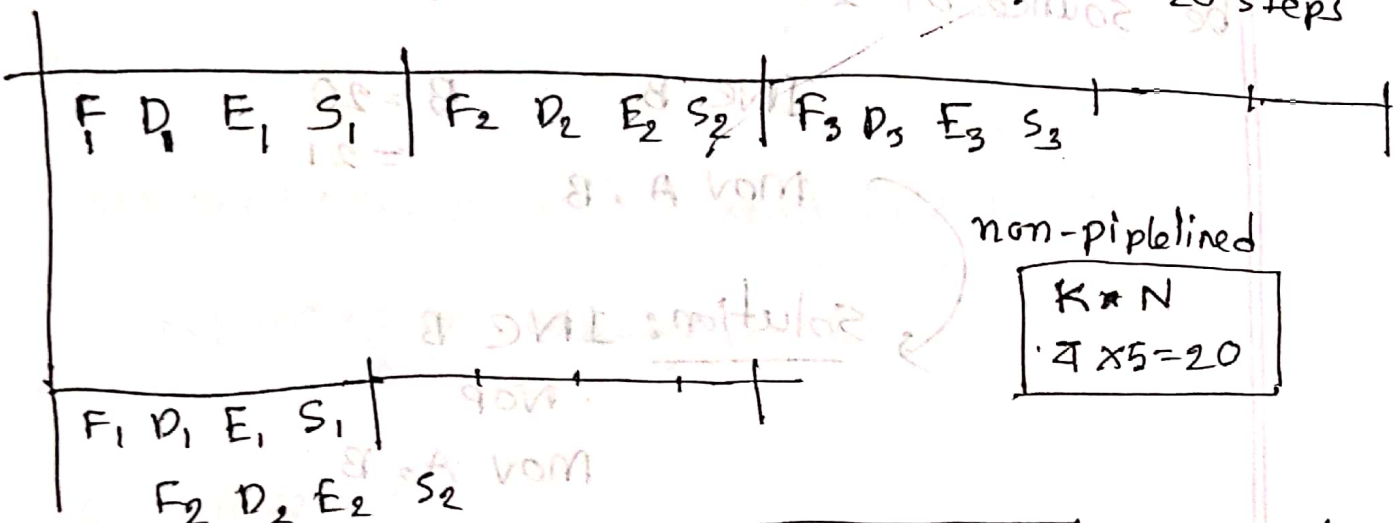
As we are taking 2 main process of an instruction and working on the two instructions at same time → so becomes fast. This process is called pipelining.

Lecture -14

Date: 28.05.22

4 Stage Pipelining

5 instructions – 4 stele = 20 steps



non-piplelined

K * N

4 x 5 = 20

* K + (N-1)

4 + 4

= 8

N = No. of instructions

K = 4 stage

Advantages : Fast

Dis Advantages : i) Control Hazard → removed by branch prediction algo 95%

* Barranching

ii) Data dependency

iii) Structural hazard

* Span or empty space for jumping is called pipeline bubble. If stages increases, discard increases.

Data dependency: destination of 1st instruction should not be source of 2nd instruction.

```
INC B        B = 20
MOV A, B       = 21
```

Solution: INC B
NOP
MOV A, B

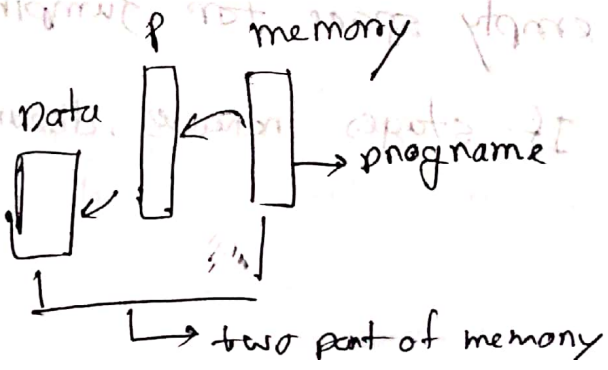
Structural hazard:

Sometimes data bus also used in execution at that time they cannot fetch other instructions.

```
Move, A, 20
Add, A, [1014]
```

RISC - It has many registers and lessen the use of memory.

PIC - Peripheral Inten Controller (better)



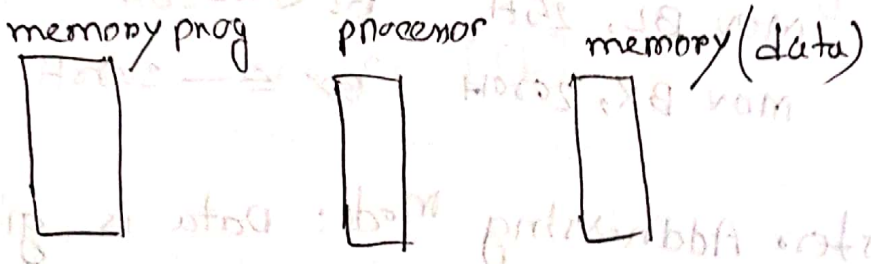
Lecture-15

Date: 20.05.22

- Overlapping of fetching and execution can only happen, if execution doesn't require memory.

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

ii) PIC 18 processor



two separate memory for program and data.

while executing if we need, use other memory. So, fetching and executing never clash with each other.

Addressing Mode

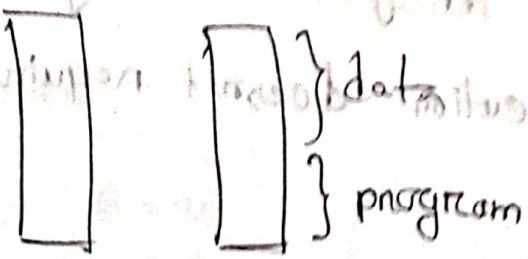
The manner by which an operand is giving instruction

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

ADD AL, 25H → operation
↳ operands

Set of instructions

- fetch
- decode
- execution



Addressing Mode

(1) **Immediate Addressing mode:** Data is given in instruction. Used for initiazing.

```
MOV BL, 25H      BL ← 25H
MOV BX, 2000H    BX ← 2000H
```

(11) **Register Addressing Mode:** Data is given in register. Used for used data again and again.

```
MOV BL, CL ;     BL ← CL
MOV BX, CX ;     BX ← CX
```

(111) **Direct Addressing mode:** Address is given in instruction

```
MOV BL, [2000H]   BL ← Data of address [2000H]
MOV BX, [2000H]

MOV AL, [2000H]  —— Direct
MOV BL, [3000H]  ——  "
ADD AL, BL       ——
```

```
mov BL, [2000H]
mov CL, [3000H]
Add BL, CL
Add BL, 55H
mov [4000H], BL
```

(v) Indirect Addressing mode: Address is given in register.

1) Register indirect: address in reg.

```
mov CL, [BL]
```

Lecture-16

Date: 04.06.22

Indirect addressing mode:

Q. Get data from location 2000 in CL

Q. Get data of hundred location starting from 2001.

Immediate: initialize

reg: move between reg

direct: need to access few memory location

Indirect: looping. (next values)

```
MOV CL,[BLH]
INC BL
```

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

ii) reg relative: addr = addr + displacement

MOV CL, [BX+B2H]

iii) Base indexed: addr = base + index reg, MOV CL, [BX+SI]

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

MOV CL, [BX+SI+03H]

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

reg relative: Ex-

Get the 100th value after BX.

MOV CL, [BX+100H]

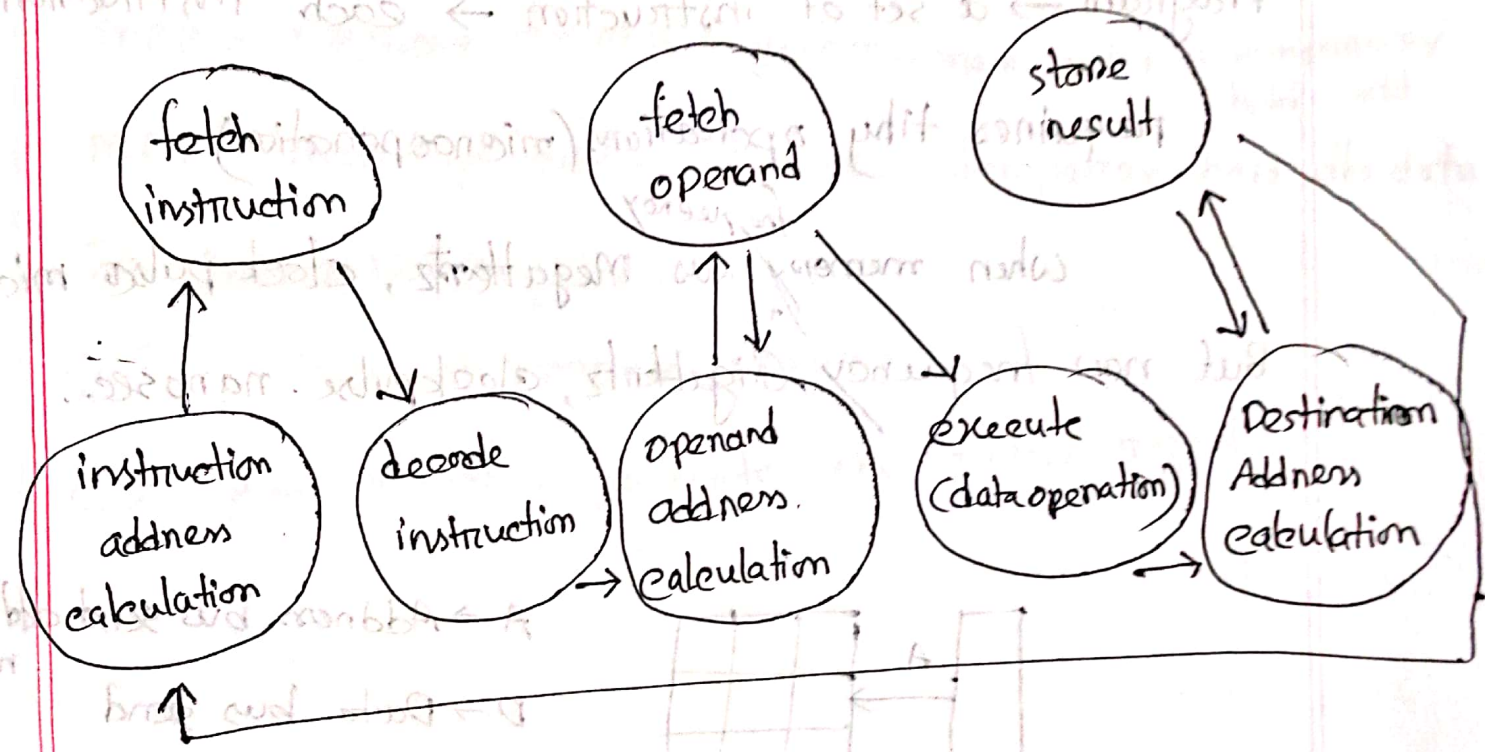
Base indexed: Get the value of 50 data after 50H.

MOV CL, [BX+SI]

INC SI

* One act as base
* other act as an index.

Instruction cycle state: transition diagram:



Lecture-17

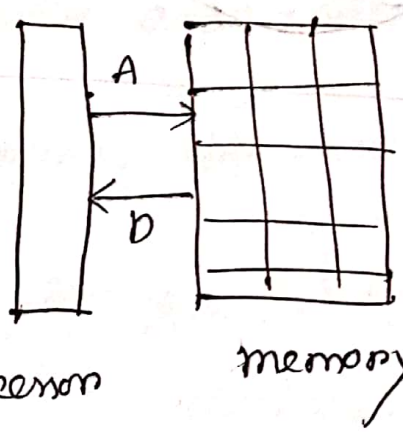
Date: 06.06.22

Program → a set of instruction → each instruction requires fifty operations (microoperation)

frequency

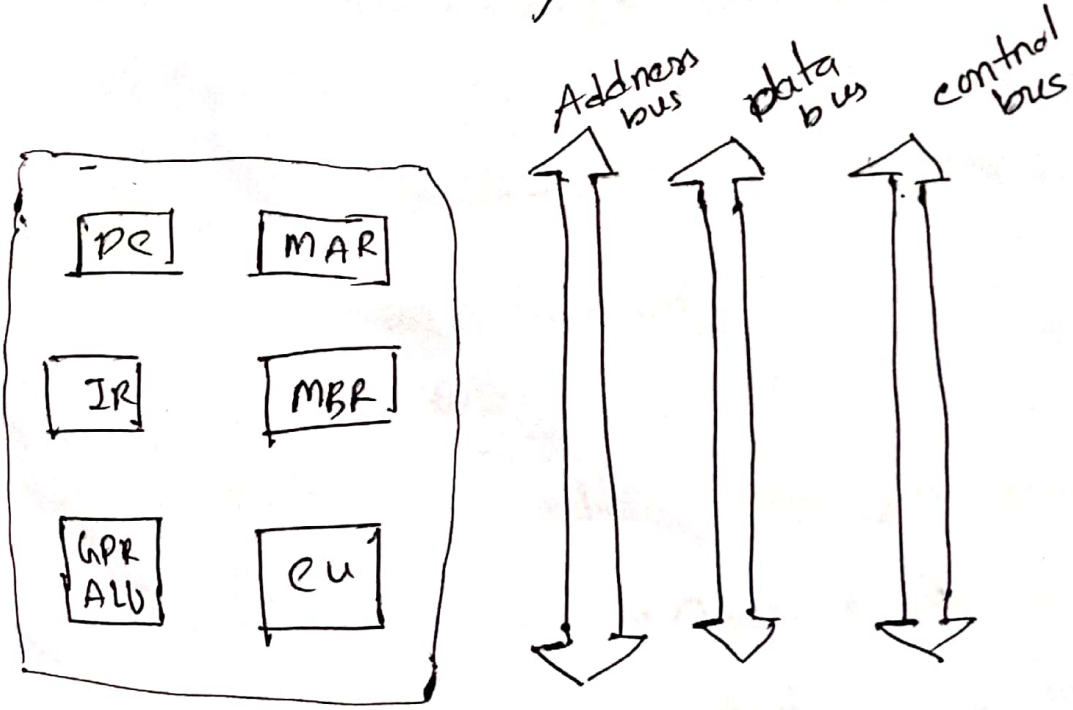
. when memory was Megahertz, clock pulse microsec.

But now frequency Gigahertz, clockpulse . nanosec.



A → Address bus send add to memory

D → Data bus send the instructions to processor.



PC - Program Counter - fetch the next instruction

MAR - Memory Address Register - mem add reg temporary hold add

MBR - Memory Buffer Register - temporay hold the data

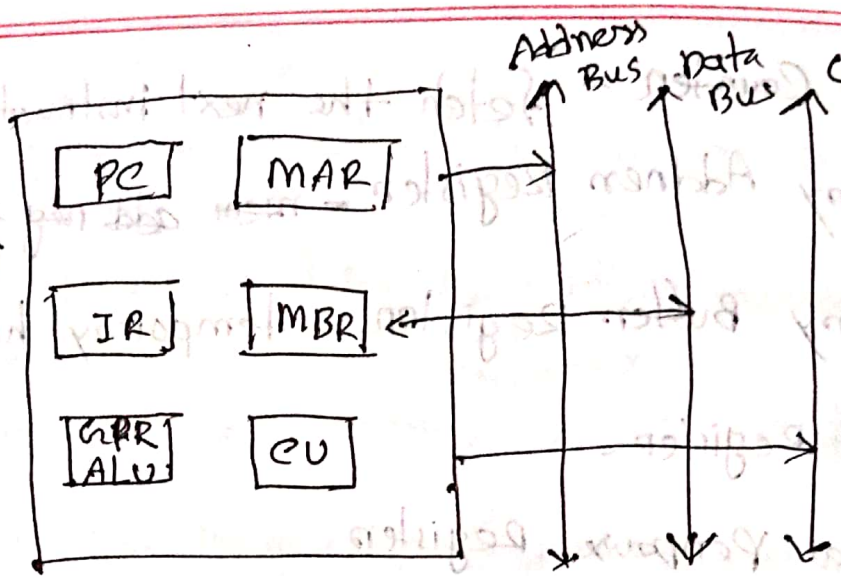
IR - Instruction Register -

GPR - General Purpose Register

CU - control Unit - generate the control signal.

Lecture-18

Date: 08.06.22



- Program is a set of operations and each operation needs a set of tiny activities. These tiny activities are known as microoperation.

- Address goes to memory by address bus. PC fetch the next instruction. Then from PC to MAR, then to address bus.

- Then instruction comes to MBR from data bus, then from MBR to IR

micro-operation for fetch:

T1 : MAR ← PC

T2 : MBR ← mem (instruction)

T3 : IR ← MBR

PC ← PC+1

each clock pulse is a triggen to proceed to do a new activity

micro operation instruction

1) Immediate addressing mode

MOV BL, 25H

T1 : MAR ← PC

T2 : MBR ← mem (Inst)

T3 : IR ← MBR

PC ← PC+1

T4 : BL ← 25H (IR)

when ADD BL, 25H

T4 : BL ← BL + 25H (IR)

2) Reg addressig mode

MOV BL, CL

T1 : — same

T2 : —

T3 : —

T4 : BL ← CL

→ T4 : BL ← BL + CL

3) Direct addressing mode : MOV BL, [5000H]

T1 : — T2 : — T3 : —

~~T4 : BL ← [5000H] (IR)~~

T4 : MAR ← IR (add) add (IR)

T5 : MBR ← mem (dat

T6 : BL ← MBR [5000H]

4) Indirect addressing mode:

mov BL, [CX]

T1: —
T2: —
T3: —
T4: MAR ← ~~BX~~ CX
T5: MBR ← mem
T6: BL ← MBR

T4: MAR ← IR(add)
MBR ← mem(add)
Reg ← MBR[add]

5) Implied addressing mode:

T1: —
T2: —
T3: —
T4: CF ← 1

Ep: STC [set the canny flag]

CT2:

Adder, Booth mult, division, subtractor, ~~pipelining~~

⇒ Why not use tradition mult, piping hazard, booths mul/div calculation

Lecture-19

Date: 18.06.22

Cache Memory Mapping Technique

1) Fully Associative mapping

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

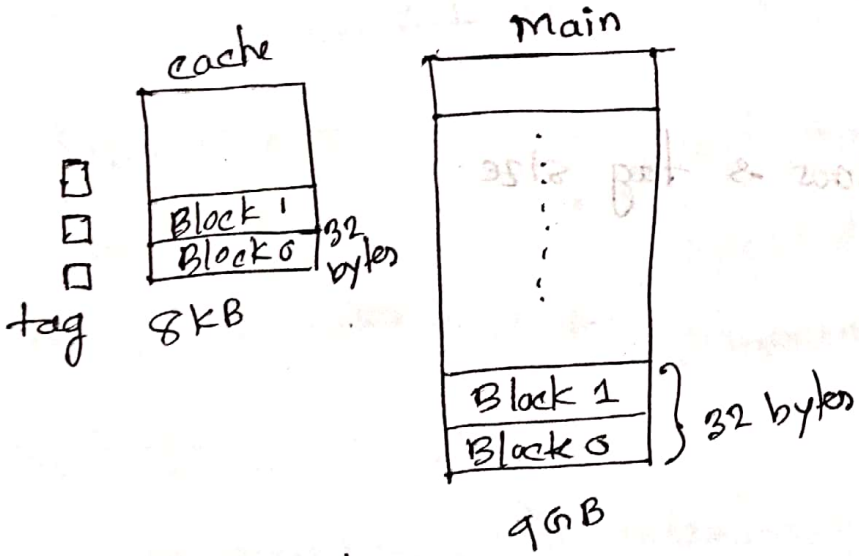
cache: lies between processor and memory

95% ↑ hit ratio

hit ratio:

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

For pentium processor →



- Block size = 32 bytes

No of block in main memory = $\frac{4GB}{32 \text{ bytes}}$

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

$= 2^{27}$

- tag: will indicate which block from memory is present in cache. (27 bit tag required)

- No. of block in cache = $\frac{8kB}{32 \text{ bytes}}$

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

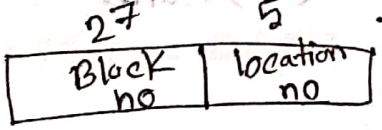
$= 2^8 = 256$

Cache directory is the collection of all tags.

tag = 27

search = 256 worst chase - if missing

hit ratio = best



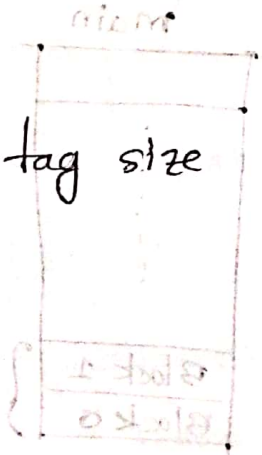
→ to represent the block no and location no.

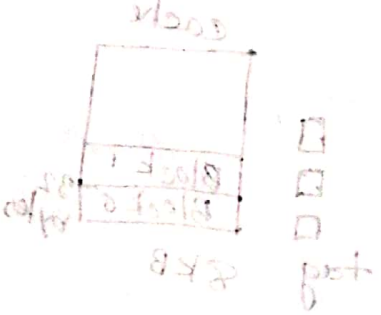
8/3 - represent location - 3 and no of block - 8

if we get this address in 3rd hit in cache.

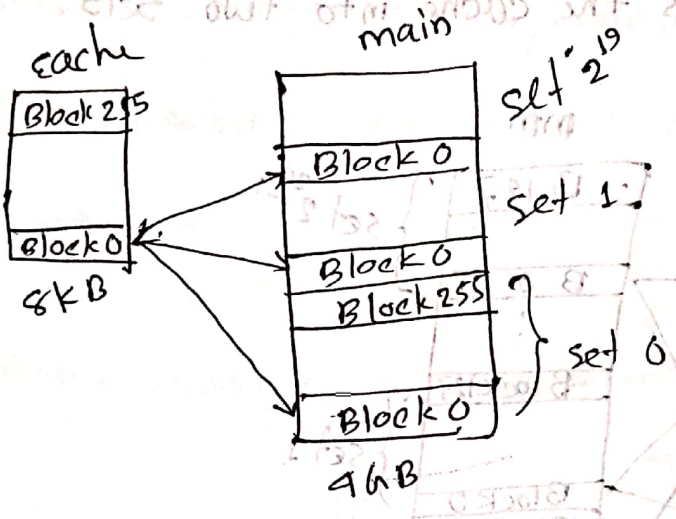
then the 3/3 will be the location and block of cache.

drawback - slow & tag size



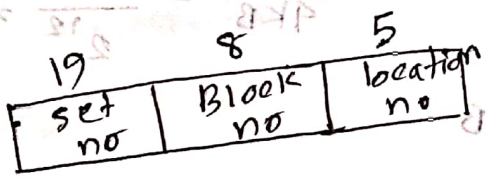


11) One way Associative mapping:



Size of set = 8kB

No of set in main memory = $\frac{4GB}{8kB} = \frac{2^{32}}{2^{13}} = 2^{19}$



→ tag represents set no

tag = 19 bits

search = 1

hit ratio = worst

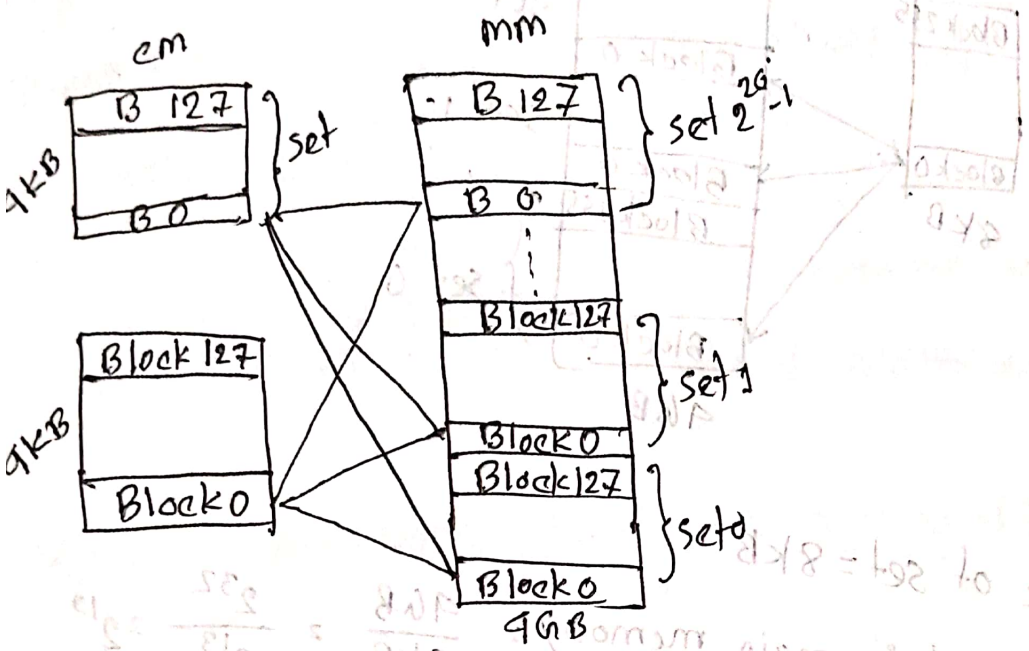
worst case: repeated use of block 0 of different set.

- In description - all calculation should be included with an example.

Lecture-20 Date: 20.06.22

Two way associative Mapping:

it divides the cache into two sets.



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

set size = 4KB

mm size = 4GB

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

size of block = 32 bytes = 2^5

set	block	location
20	7	5

tag = 20 → represent set no.

search = 2

Hit ratio = way beten

S B L
1 1 1
2 3 5

worst case: there are some blocks in repeated manner.

Memory Hierarchy

Primary memory / physical / main mem. (original)

RAM → constant power supply, volatile } chip memory, expensive, fast

Rom → non-volatile

Secondary mem (to increase storage capacity)

HD →
FD (floppy disk) } magnetic disk

CD →
DVD → } optical disk

pendrive - semiconductor

} writable, non volatile, cheap, need large power

HD, FD, CD, DVD

data acquining rate =

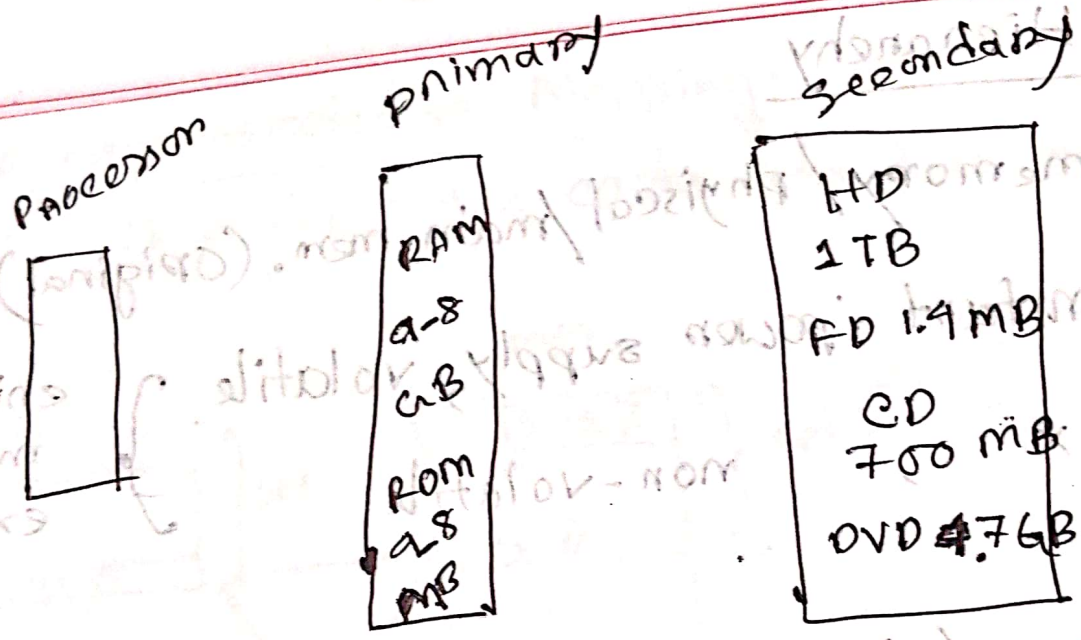
rate of disk rotation (rpm)

(fast + cheap) → secondary mem

/ for operation \ for storage

Q. How speed increase of . by increasing

RAM size?


Processor
primary
RAM
4-8
GB
ROM
2-8
MB
Secondary
HD
1TB
FD 1.4MB
CD
700 MB
DVD 4.7GB