Cache Mapping: The process of bringing data of main memory blocks into the cache block is known as cache mapping.

3 mapping techniques:

1. Direct
2. Associative
3. Set Associative

1. Direct Mapping: Each block from main memory has only one possible place in the cache organization.

Every block j of the main memory can be mapped to block i of the cache using formula: $i = j \bmod m$

3 fields: Tag, Line, Block offset.

To map the memory address to cache: The BLOCK field of the address is used to access the cache's BLOCK. Then the tag bit in the address is compared with the tag bit of the block.

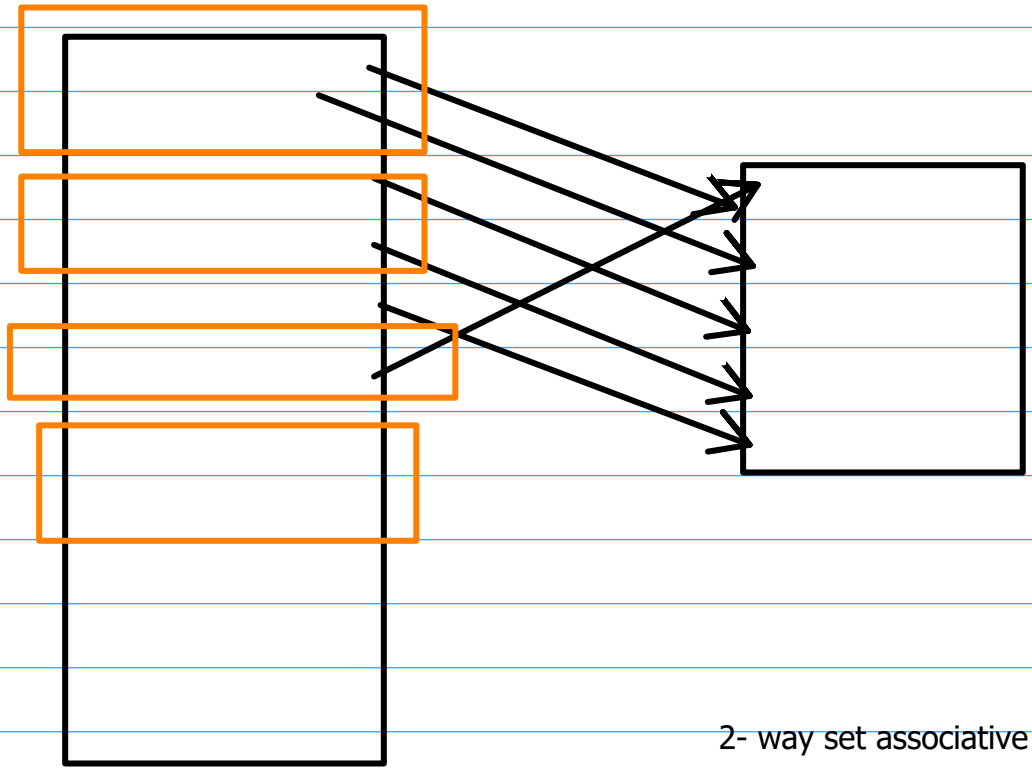
2. Associative Mapping:The mapping of the main memory block can be done with any of the cache block. The memory address has only 2 fields:
Word and tag

3. Set Associative mapping: It is the combination of advantages of both direct and associative mapping.

Here the cache consists of a number sets, each of which consists of a number of blocks.

The relationship is: $n = w * L$
w: number of sets
L: number of lines in each set

n: number of blocks



2- way set associative mapping

Consider a 2-way set associative mapped cache of size 16 KB with block size 256 bytes. The size of main memory is 128 Kb. Find

1. Number of bits in tag
2. Tag directory size.

Answer:

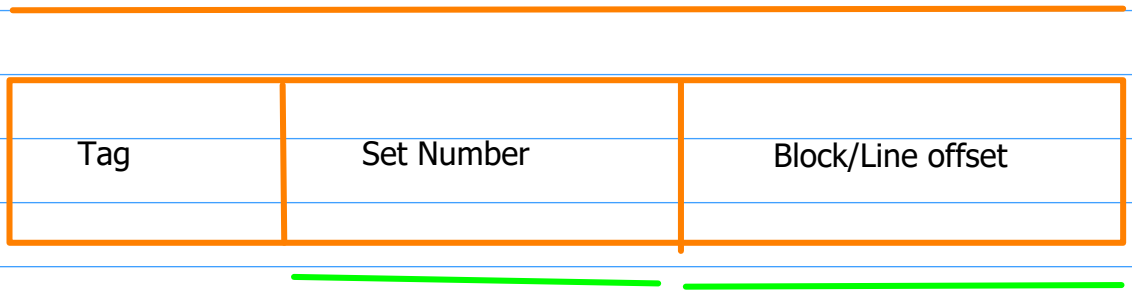
Set size = 2
Cache memory size = 16 Kb
Block size = Frame size = Line size = 256 bytes
Main memory size = 128 Kb.

Main memory size= 128 Kb = 2 ^ 17 bytes
Number of Bits in Physical Address = 17 bits

Block size = 256 Bytes = 2 ^ 8 Bytes
Number of bits in Block offset = 8 bits

Number of lines in cache size = Cache size / Block size
= 2 ^ 14 / 2^8
= 2^6 = 64 lines

Number of sets in Cache = Total number of lines in cache / Set size
= 64/ 2
= 32 sets = 2 ^ 5 sets.



Tag = 17 - 8 - 5 = 4

Tag directory size = Number of lines in cache x number of bits in Tag
= 64 x 4 bits
= 256 bits
= 256 / 8 = 32 bytes

27th November, 2023

n instructions , k stage pipeline, no of cycles = (n+k-1) (in pipelined processor)

Performance of a pipelined processor

Consider a k segment pipeline with clock cycle time T_p. Let there be n tasks to be completed in the pipelined processor.

ET_pipelined = n+k-1 cycles
= (N+k-1) Tp

In the same case, for a non-pipelined processor, the execution time of n instructions will be = (nk)Tp

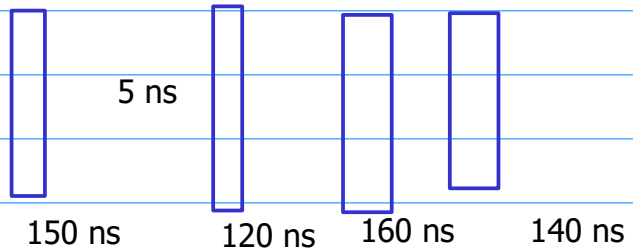
Speed up = t_old / t_new
= (N * K * Tp) / (n+k-1) * Tp
= nk / n+k-1

A 4 stage pipeline has the stage delays as 150, 120, 160 and 140 ns, respectively. Register that are used between the stages have a delay of 5 ns each. Assuming constant clock rate, the total time taken to process 1000 data items on this pipeline will be ____ microseconds

n = 1000

k = 4

Tc = 160+5 = 165 ns



Execution time Tp = (n+k-1) Tc
= (1000+4-1)165
= 165495 ns = 165.495 ms

Pipelining Hazards

hazards that arise in the pipeline to prevent the next instruction from executing during its designated clock cycle.

1. Structural Hazards: Hardwares cannot support certain combinations of instructions (two instructions in the pipeline require the same resource.)

2. data Hazards: Instructions depends on result of prior instruction still in the pipeline.

R1 = 1

Add R1, R2, R3 Instruction 1 R1 = 3+4 = 7

JUMP

SUB R4, R1, R5 Instruction 2 R1 = 8

Resource 1

1 2 3 4 5 6 7 8

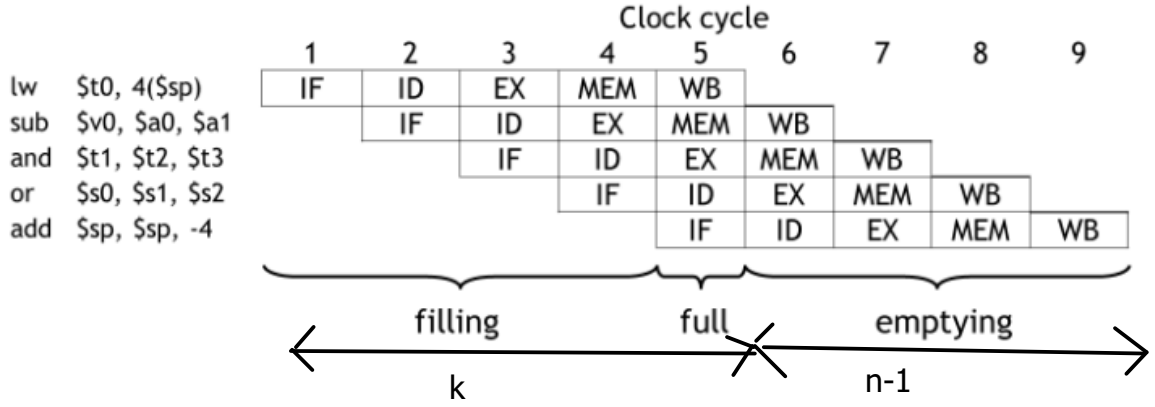
IF ID EX Mem WB

R1

IF ___ ___ _ ID Ex mem Wb

Control Hazard: Caused by delay between the fetching of instructions and decisions about changes in control flow (Branches and jumps)

29th November, 2023



Speed up = nk / n+k - 1

Throughput: No of instructions executed with respect to time.

If I use pipeline, what will be my execution time? -> (n+k-1) t_c

Throughput = n / (n+k-1)t_c

If my no of instructions is infinite, then the throughput will be 1/t_c <-- Ideal throughput

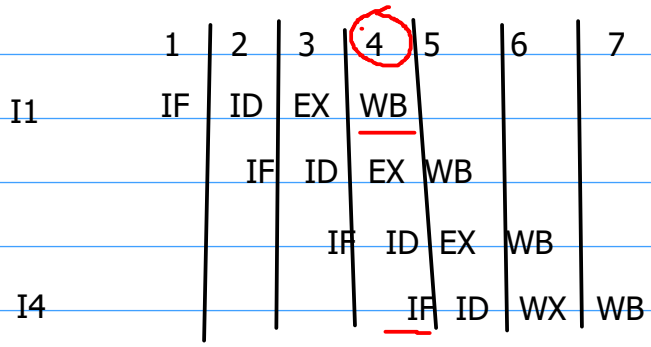
Efficiency or Utilization of CPU = nk / (n+k-1)k = n/(n+k-1)

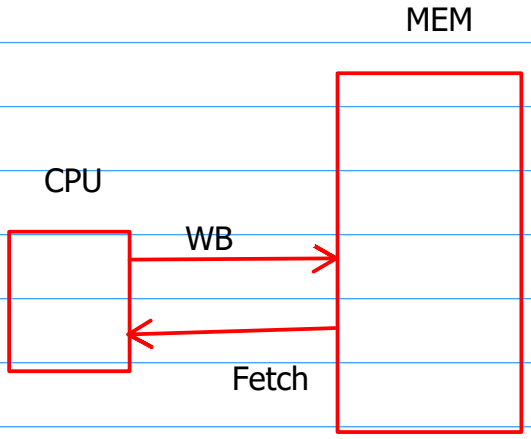
What will be the relation between speedup and efficiency?

Speedup = Efficiency x k

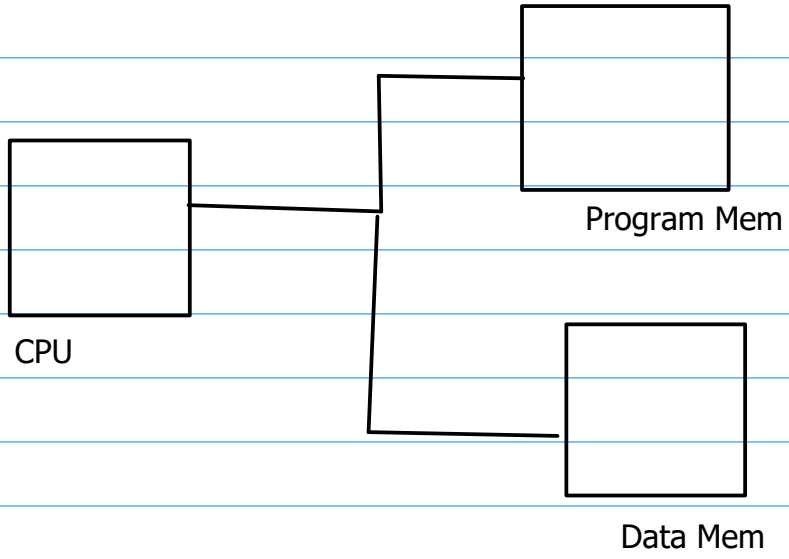
Structural Hazard

Consider a 4 stage pipeline: IF, ID, Ex, WB





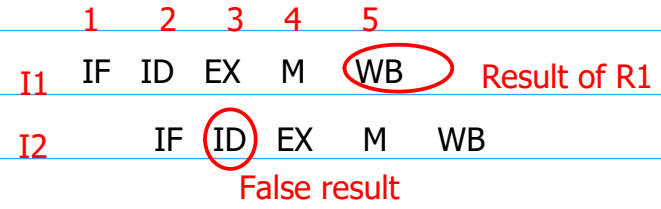
Havard Architecture



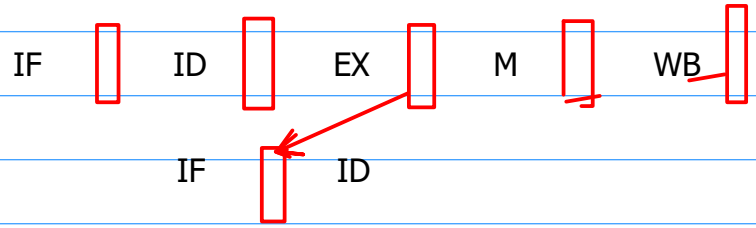
(Mul) I1	IF	ID	Ex	Ex	Ex	WB	
(ADD) I2		IF	ID	_	_	Ex	WB

Data Hazard

```
I1   :    ADD   R1, R2, R3  ;   R1 <-  R2+R3
I2   :    ADD   R4, R1, R5  ;   R4 <-  R1+R5
```



Operand Forwarding



1. RAW (True/ Actual Dependency)

```
ADD   R1, R2, R3    [Previous Instruction]
ADD   R4, R1, R5    [Next Instruction]
      OP  IN
```

If there is common operand between OP of prev instr. and IP of next instr. , then there will be RAW hazard

2. WAR

```
ADD  R1, R2, R3
ADD  R2, R1, R3
```

If there is common operand between ip of prev instr. and OP of next instr. , then there will be WAR hazard

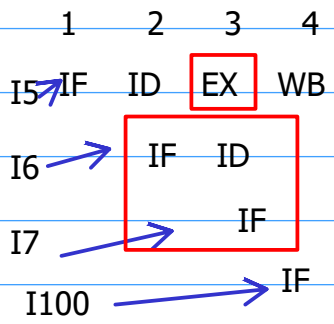
3. WAW

```
ADD  R1, R2, R3
ADD  R1, R4, R5
```

Control Hazard

Why ? -> set of instructions

Address	Instructions
1000	I1
1004	I2
1008	I3
1012	
1016	I5 (Jump 8000)
1020	I6
.	
.	
.	
8000	I100



One way to handle control hazard is branch prediction.

Machine learning -> main use case is to predict some values based on trained model.

Weather Prediction

2000 -> 34

2001 -> 33

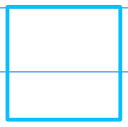
.

.

.

2020 -> 37

2025 -> ? <Predict>



1> Train (On available data)
2-> Predict

Chatgpt -> LLM (Trained huge number of text) -> it can generate desired output

NN

Intel I5 -> architecture , overview process design

serverless -> What is serverless? Serverless computer 's architecture

CLoud -> What is cloud?

Consider a non-pipelined processor with a clock rate of 2.5 ghz and average cycles per instruction of 4. The same processor is upgraded to a pipelined processor with 5 stages but due to the internal pipeline delay, the clock speed is reduced to 2 ghz. Assume there is no stalls in the pipeline. What is the speed up?

Cycle time in Non-pipelined processor:

Frequency = 2.5 ghz
Cycle time = 1 / f = 1/ 2.5 ghz = 1/(2.5 x 10 ^9) = 0.4 ns

Non-pipeline Execution time
= Number of clock cycles taken to execute one instruction
= 4 clock cycles
= 4 x 0.4 ns = 1.6 ns

Cycle time in pipelined processor
= 1/2x10^9 = 0.5 ns

Pipeline Execution time

Since there are no stalls in the pipeline, so ideally one instruction is executed per clock cycle. = 1 x 0.5 ns = 0.5 ns

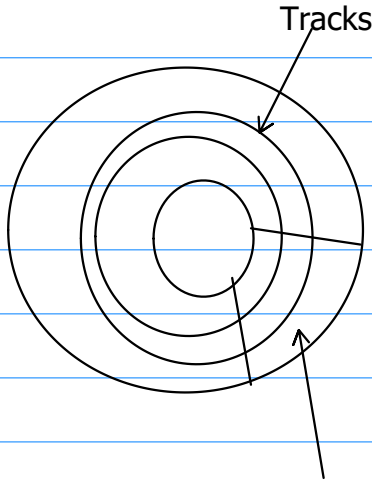
Speed up = Non-pipeline execution time / pipeline execution time
= 1.6 ns / 0.5 ns
= 3.2

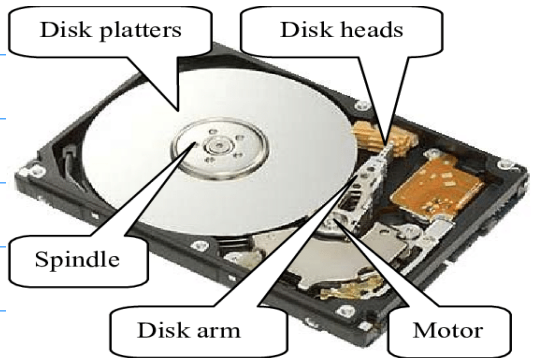
Given, Total execution time and number of instruction
Throughput = Number of instructions executer per unit time

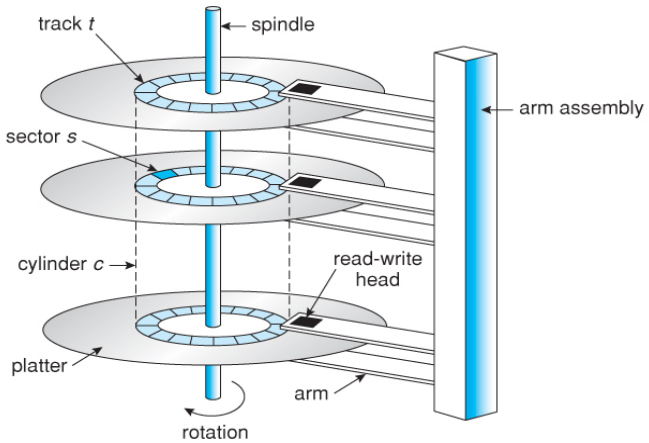
Memory - 6th December, 2023

Cpu <---> Cache Memory <----> Main Memory ----> Secondary memory
Caching Techniques
SRAM Vs Dram

Components of Hard disk:
-> Platter
-> Spindle
-> Read/Write head
-> Tracks
-> Sectors



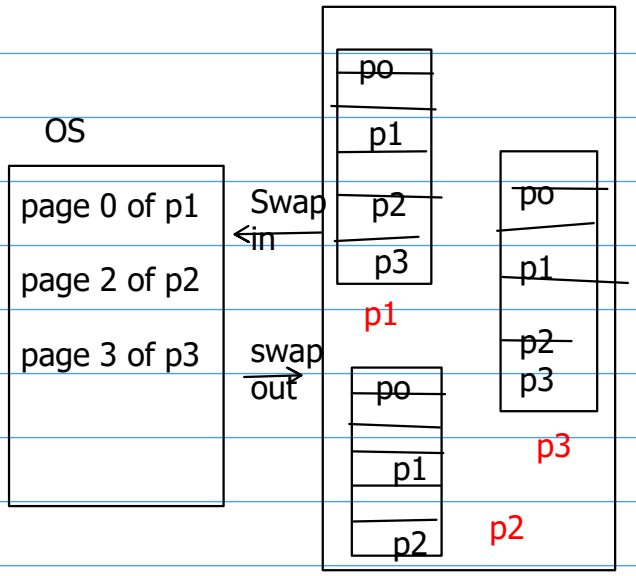




Virtual Memory

RAM 2 GB 16 GB

Game -> 3 GB 100 GB



Virtual memory is a storage allocation scheme in which secondary memory can be addressed as though it were part of the main memory.
-> illusion

Swapping is a process out means removing all of its pages from memory, or marking them so that they will be removed by the normal page replacement process.

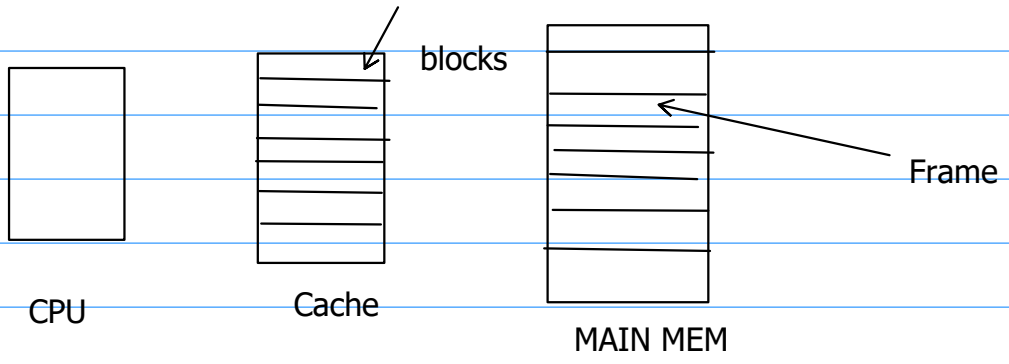
When a process is busy swapping pages in and out, then this situation is called thrashing.

This concepts high degree of multiprogramming.

Page table -> data structure used by virtual memory system to store mapping between logical address and physical address.

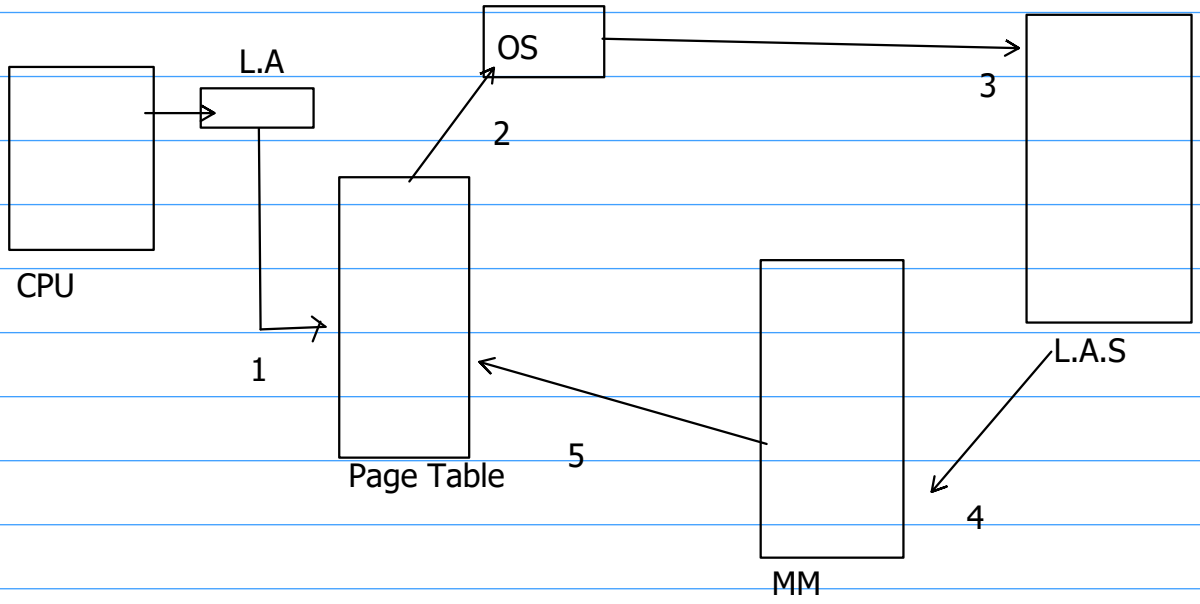
Logical addresses are generated by the CPU for the pages of the processes therefore they are used by the processes.

Physical addresses are the actual frame address of the memory. They are generally used by the hardware or more specifically by RAM subsystem.



Demand Paging

The process of loading the pages into memory on demand (whenever a page fault occurs) is known as demand paging.



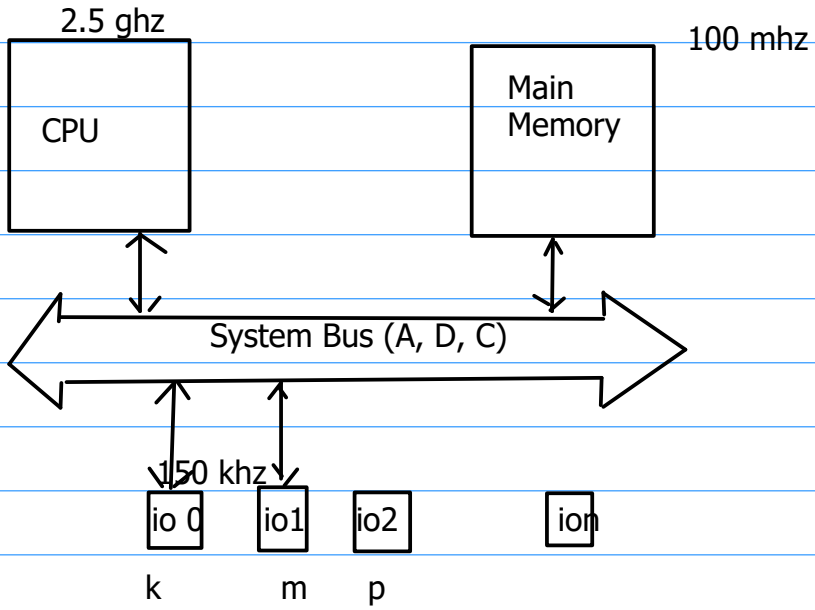
Final Exam

Main Memory concepts -> SRAM, DRAM
Cache Memory concepts
Mass Memory -> Harddisk
Virtual Memory

Processor (CPU)
Performance, speed up, amdahls law
architecture of cpu

Pipelining in processor

Input output organization



Disadvantage:

1. cpu need to wait more time

2. performance de

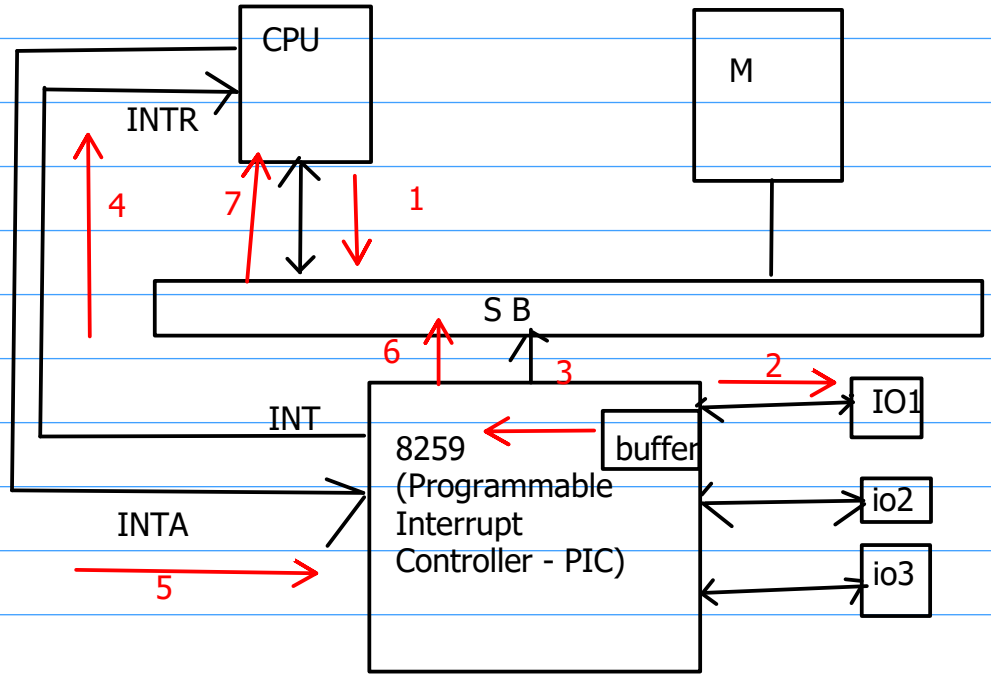
The method that is used to transfer information between internal storage and external I/o devices is known as IO devices.

Data transfer to and from the peripherals can be done in 3 ways:
1. Programmed I/O
2. Interrupt-initiated I/O
3. Direct Memory Access (DMA)

INSTRUCTIONS:

LDA IO -4

Interrupt driven IO



3 DMA