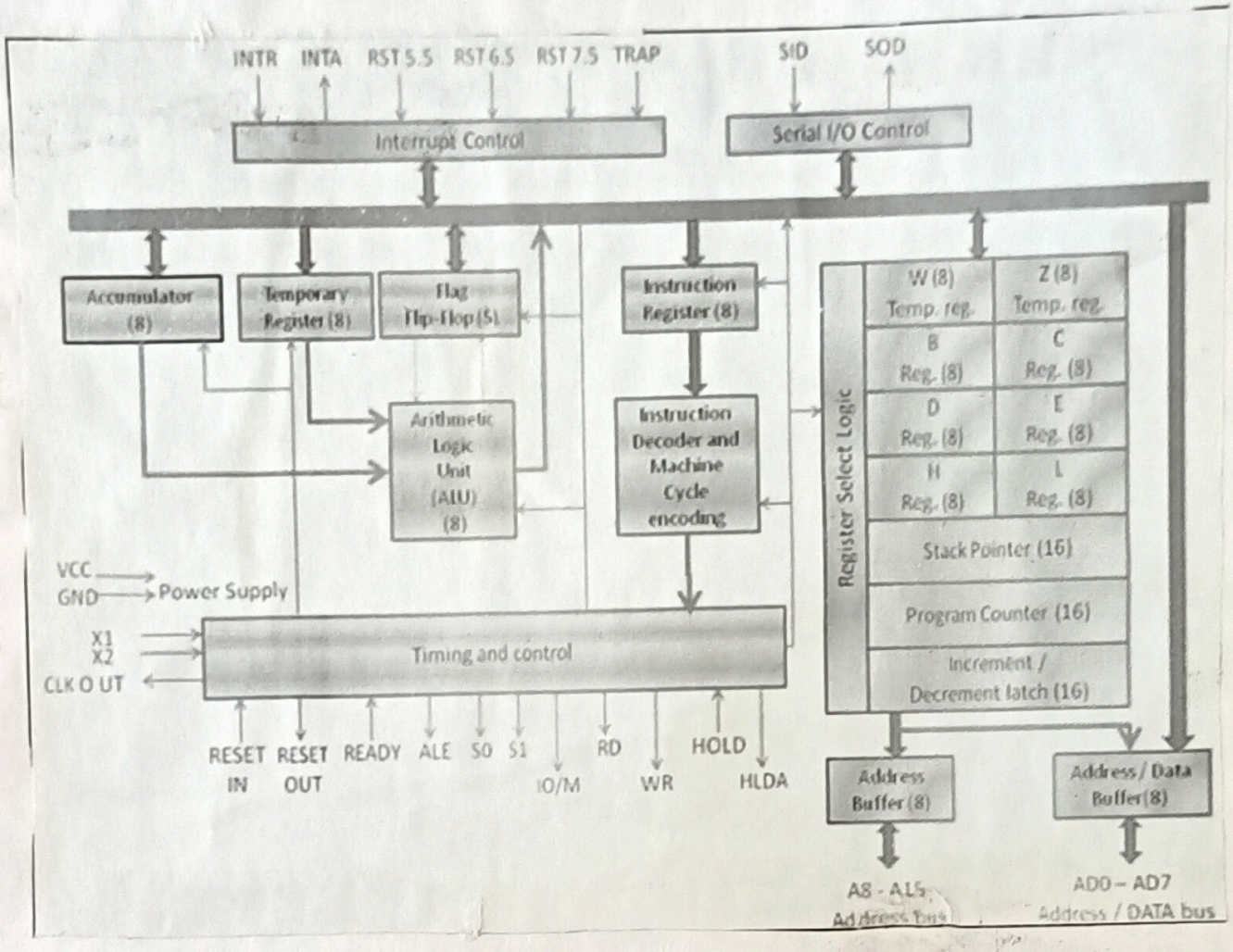
* Internal Architecture of 8085

The diagram shows as functional block diagram of 8085



The internal architecture of 8085 contains ALU, Control unit (CU) & set of registers inside it.

* ALU : Arithmatic Logic Unit

It is responsible for performing arithmatic operations like addition, subtraction, multiplication & division.

It is also responsible for performing logical operations like AND, OR, X-OR, Compliment, rotate and shift.

These operations are performed by arithmatic circuits like half adder, full adder, half subtractor, full subtractor. These circuits are implemented by logic gates & gates are implemented by using transistors.

8085 has 8 bit ALU which can operate upon 8 bit & 16 bit numbers.

* **Control Unit (CU):** It acts as a brain of the processor.

CU is responsible for generating necessary control signals at appropriate time.

The examples of control signals generated by control unit are - memory read signal ($\overline{MEMR}$) memory write signal ($\overline{MEMW}$) IO Read ($\overline{IOR}$), IO Write ($\overline{IOW}$) or Interrupt Acknowledge ($\overline{INTA}$).

* **Register File:** 8085 has or it contains a set of internal registers. These registers are used for temp. storage of data. It means these registers are acting as internal memory of 8085. It is always better to have more no. of internal registers so that we get more internal memory so that we have to depend less on external memory. Also it is better to ~~large~~ have registers of larger size.

The various registers inside 8085 can be classified according to their size use & availability to user.

Classification of Register Inside 8085

A] **Classification According to size**

i) 8 bit - examples are A, B, C, D, E, H, L
ii) 16 bit - examples are Program Counter (PC)
Stack Pointer (SP)

B] **Classification According to their Use**

i) **General Purpose Registers (GPR)**

If the register is used only for tempory storage of data, then it is known as GPR.
examples are - B, C, D, E, H & L

ii) **Special function Registers (SFR)**

IF the register is used for only a specific function, then it is known as SFR
ex. - Program Counter (PC), Stack Pointer (SP)

c] Classification of Registers based on their availability to User

i] **Programmer Visible Registers** : If the register is available to user while writing the programs then it is known as programer visible register.
ex - A, B, C, D, E, H, L

ii] **Programmer Invisible Registers** : If the register is not available to user while writing the programs, but it is used internally by the processor (8085), then it is known as Programmer Invisible Registers
ex - W, Z.

* Detailed Explaination (Register Organization) of Registers

* **Accumulator (A)** : 8085 Contains a 8 bit register known as accumulator. It is denoted as 'A'. While performing any arithmatic operation (eg. addition of two 8 bit nos), out of two operands (nos), one operand or number is always stored in A. After performing the desired operation, the result generated is stored by into Accumulator. It is a by default process.

* **General Purpose Registers**

8085 provides six general purpose register having size of 8 bit each.
These registers are named as B, C, D, E, H & L.
They are used for temporary storage of 8 bit data.
These register can paired together to store 16 bit numbers. But 8085 allows only limited pairing
Three possible register pairs are BC DE & HL
In these pairs we can store a 16 bit no. as size of pair is 16 bit.

W & Z Registers:

These two registers i.e. W & Z are of 8 bit each.
They are known as Programmer Invisible Registers.
These two registers are not available to the user while writing the programs, but they are used internally by 8085 itself

Program Counter (PC):

It is a 16 bit Special Function Register (SFR).
It always contains 16 bit memory address.
It contains a specific address. i.e it contains 16 bit memory address of the next memory location which is to be executed.
Thus, PC is used to keep track of execution.

Stack Pointer (SP):

It is a 16 bit Special function Register (SFR)
It always contains 16 bit memory address of stack memory.
Stack pointer (SP) always points towards ~~stack~~ top of the stack memory.

Flag Register:

8085 has a 8 bit flag register.
It contains 5 flags. (flip flops)
Each flag requires 1 bit from flag register. Hence intotal 5 bits from flag Register (8 bits) are used & the remaining three bits are not used or unused.
Flag Register gives us the status of the result generated by ALU.
The result generated by ALU, after performing the desired operation is stored in Accumulator (A) as well as it is given to flag register so that the respective flags are updated (i.e. they are set or resetted).

8085 has 5 flags & these are

i) sign (S)
ii) Zero (Z)
iii) Carry (CY)
iv) ~~Auxiliary~~ Auxillary Carry (AC)
v) Parity (P)

The format of flag register is as shown -

S	Z	-	AC	-	P	-	CY

Sign (S): If the result of arithmatic operation is negative (-), then this flag is set to One, otherwise it is resetted or zero.

Zero (Z): If the result of arithmatic operation is zero then this flag is set to One, otherwise it is resetted or becomes '0'

Carry (CY): While performing addition of two 8 bit nos. if carry is generated from MSB (Most significant Bit), CY flag is set to One, otherwise ~~resetted~~ (0)

**There is no seperate flag for borrow (subtraction) The same flag is used subtraction operation also. While performing subtraction of two 8 bit nos. if borrow is generated from MSB, then also CY flag is set to one, otherwise resetted (0)

Auxillary Carry (AC):

This flag is used in case of BCD operations.
BCD - Binary Coded Decimal.

While performing addition of two BCD nos. if carry is generated from D_3 bit to D_4 bit (i.e lower nibble to upper nibble), this flag is set (1), otherwise it is resetted (0)

Parity (P)

If the 8 bit result generated in accumulator (A) contains even no. of one's then this flag is set to one, otherwise it is resetted or becomes zero (0)

Tempory Register

It is an 8 bit register. It is used to hold (store) the second operand (no) temporarily.

While performing any arithmatic or logical operation inside ALU, one no. is provided to ALU by Accumulator & the second no. is provided by Tempory Register.

Instruction Register (IR)

It is an 8bit register

It is used to hold Opcode (short code) of the instruction which is to be executed.

Apart from all these registers there are some other blocks present in architecture diagram.

Instruction Decoder & Machine Cycle Encoding Unit

Opcode of the instruction which is to be executed is first stored into Instruction Register (IR) & then it is given to this block.

This block will decode the instruction so that 8085 will come to know exact meaning of the instruction (i.e what to do with this instruction) & then it will calculate the amount of time required for execution of this instruction. The time required for execution of inst. depends upon the clock frequency

Serial I/O Control

This block will control the serial communication facility in 8085. SOD pin will be used to send out the data from 8085 in bit by bit manner & SID is used to receive the data in bit by bit manner.

Interrupt Control :

This block will control the five H/W interrupts of 8085 i.e TRAP, RST 7.5, RST 6.5, RST 5.5 & INTR. Also if INTR interrupt is received, it will generate acknowledge signal $\overline{INTA}$

All the blocks are connected together by using the internal bus as shown in the architecture diagram.

Concept of memory, memory locations, memory address

In any microprocessor including 8085 the size of main memory (RAM) is decided by number of address lines present in that processor. The formula used is

$$2^N = \text{Size of Main memory (RAM)}$$

where N = No. of address line.

In 8085, there are 16 address lines, hence size of main memory (RAM) will be 2^{16} = 64 Kbytes.

In computers 1 Kbyte = 2^{10} bytes
= 1024 bytes.

Hence in 8085, 64 K bytes = 64 X 1 Kbytes
= 64 X 1024 bytes
= 65,536 bytes.

Thus in 8085 RAM i.e main memory size is 64 Kilo bytes or 65,536 bytes.

The main memory (RAM) of 8085 has 65,536 memory locations.

Each location is capable of storing 8 bit data or 1 byte i.e storage capacity of each location is 8 bit or 1 byte

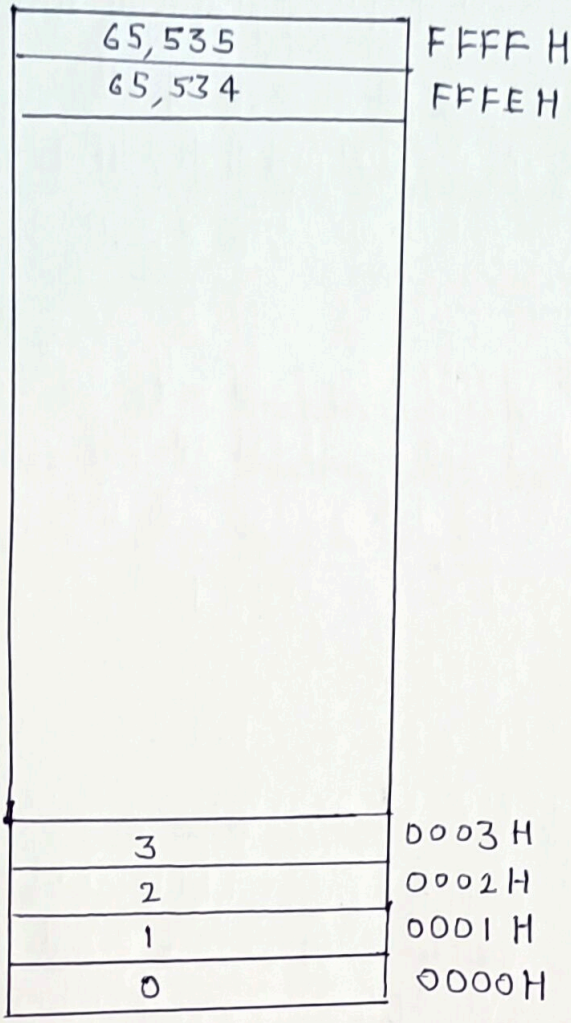
In total there are 65,536 locations. Hence each memory location is identified by unique address & known as memory address.

The memory address of each memory location is 16 bit as there are 16 address lines in 8085.

* With above dissecussion we have to remember two things

1] Each memory location can store only 8 bit (1 byte) data

2] Each location is identified by 16 bit unique address known as memory address.



* In above diagram 64K memory is divided into 65,536 locations. The first mem. location is numbered as '0' hence last will be 65,535
* Each location is capable of holding 8 bit data or 1 byte
* Each location will 16 bit memory address.

The address of 0th Location = 0000 0000 0000 0000
All 16 address lines will have value zero.
The address of 1st location = 0000 0000 0000 0001
The address of 2nd location = 0000 0000 0000 0010
3rd location = 0000 0000 0000 0011

⋮

The address of last or 65536th location = 1111 1111 1111 1111

But it is quite difficult to remember these 16 bit binary addresses. Hence this 16 bit address is divided into 4 groups each of bits. When 4 binary bits are combined together

it gives its hexadecimal Equivalent.

For ex. add. of 0th location = 0000 0000 0000 0000

When these 16 bits are grouped into groups of 4 bit each we get

0000	0000	0000	0000
0	0	0	0

Thus address of 0th location will be 0000H in hex

Address of 1st location will be 0001H in hex

0000	0000	0000	0001
0	0	0	1

Similarly,

The add of last location (65,536) will be FFFFH in hex

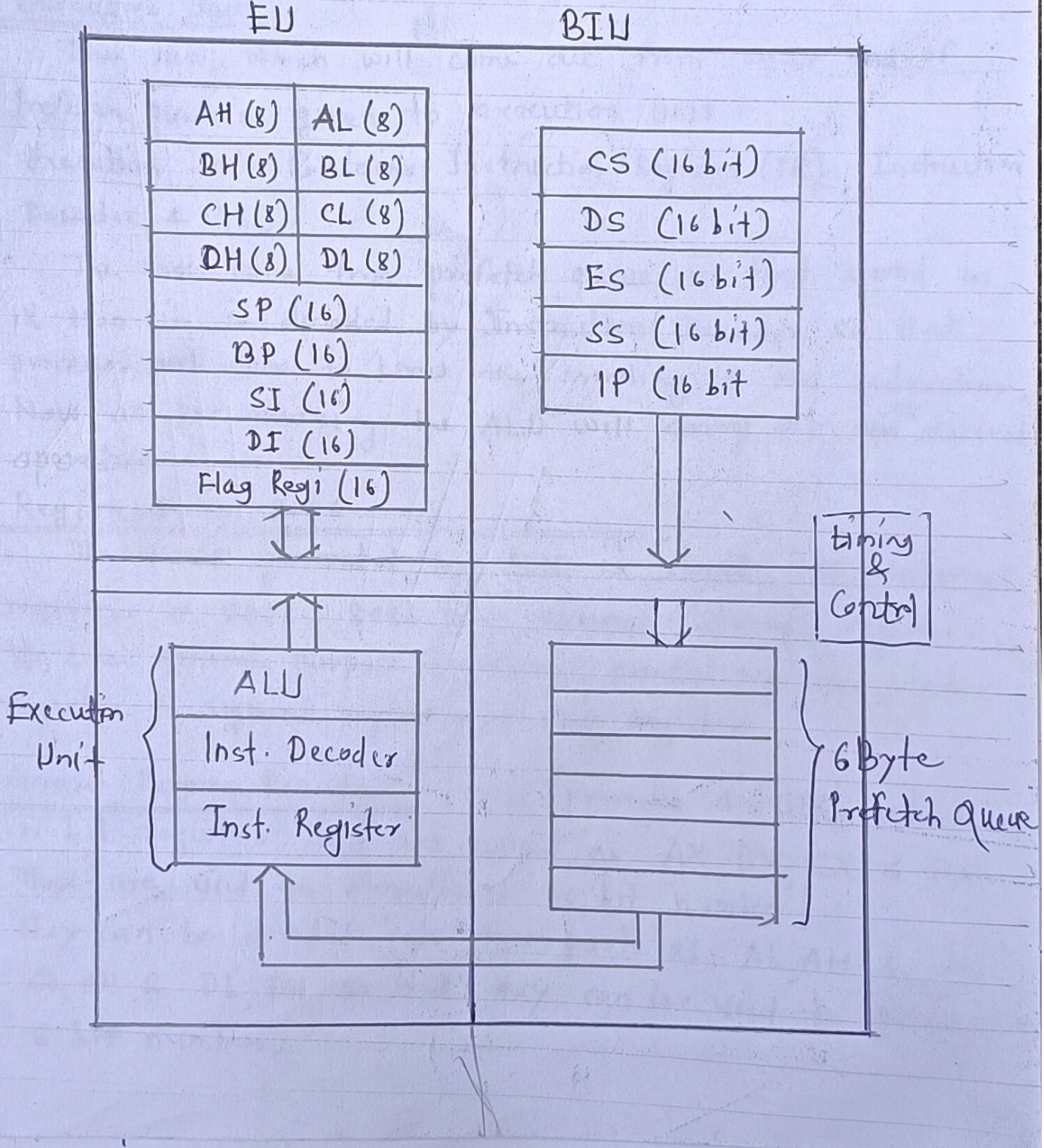
1111	1111	1111	1111
F	F	F	F

* For simplicity memory addresses are always expressed in Hex form.

* In this way the starting address of 8085 memory is 0000H & the ending or last address is FFFFH.

Architecture of 8086 Processor.

8086 is successor of 8085. 8086 is the first processor in which first time concept of prefetching (inst. pipeline) is used. Also first time main memory is divided into number of smaller parts called segments. The internal architecture of 8086 is as shown in the diagram. The diagram is divided into two parts 1) Execution Unit (EU)
2) Bus Interface Unit (BIU)



BIU: It is responsible for all the communication with outside world (memory or I/O devices).

The BIU prefetches the instructions from main memory by using system bus.

Prefetch Queue: The instructions prefetched by BIU will enter into prefetch queue. In 8086 the length of prefetch queue is 6 bytes (i.e it can store 6 bytes / 6 instructions) The queue follows first in first out (FIFO) policy.

Execution Unit:

The inst. which will come out from other end of Prefetch queue is given to execution unit.

Execution Unit Contains Instruction Register (IR), Instruction Decoder & ALU.

The inst. came from prefetch queue is first stored in IR then it is decoded by Instruction Decoder so that processor come to know the meaning of the instruction. Now as per meaning the ALU will carry out the desired operation.

Registers in 8086

The result generated by ALU is stored into internal registers of 8086. 8086 have various internal registers. like general purpose registers, pointer registers, index register & segment registers, & flag register

General Purpose Registers: 8086 provides 4 GPR's which are 16 bit registers. They are named as AX, BX, CX & DX They are used to store/hold 16 bit numbers. They can be divided into two parts as AL, AH, BL, BH CL, CH & DL, DH so that they can be used to store 8 bit numbers.

Pointer Registers

8086 Provides two pointer registers 1) SP 2) BP

1) Stack Pointer SP (16 bit) : It is a 16 bit SFR which always points towards top of the stack memory
2) Base Pointer BP (16 bit) : It is a 16 bit SFR which always points towards normal memory.

Index Registers

8086 provides two 16 bit index register SI & DI

1) Source Index (SI) - 16 bit
2) Destination Index (DI) - 16 bit

These two index registers are used in case of string or array related operations for auto increment/decrement

Flag Register

8086 provides 16 bit flag register
8086 has 9 flags. Each flag requires One bit from FR
The 9 flags are

1) Sign (S)
2) ~~Carry~~ Zero (Z)
3) Carry (CY)
4) Auxillary Carry (AC)
5) Parity (P)
6) Overflow (OV)
7) Direction flag (DF)
8) Interrupt flag (IF)
9) Trap flag (TF)

The first 5 flags are same as 8085 & they are known as status flags & the remaining 4 flags are known as control flags.

Segment Registers:

8086 provides 4 segment registers each of 16 bit
They are named as CS, DS, ES & SS
They are used to give starting address of the segme

Instruction Pointer

8086 provides a 16 bit Instruction Pointer which gives offset value (distance to be travelled within the segment.

Segmentation

* Dividing the larger size of memory into number of smaller modules is called as segmentation.
* 8086 has 20 Address lines

∴ Size of Main Memory = 2^{20} = 1M

This 1 MB memory is divided into 16 equal parts or segments.

Each segment is of size 64 KB

Out of 16 segments only 4 segments are active at a time. The active segments are named as —

1] Data Segment
2] Code Segment
3] Extra Segment
4] Stack Segment

Control Unit Design:

Generally a digital system is seperated into two parts 1) Data processing Unit
2) Control Unit.

* The data processing unit is collection of functional units capable of performing certain operations on data.
* While the control unit issues control signals to the data processing part to perform operations on data.
* These control signals selects the functions to be performed at specific times & route the data through appropriate functional units.

We can say that the data processing unit is logically reconfigured by the control unit & it is intimately involved in the sequencing & timing of the data processing unit.

Control Unit Design Methodology:

There are two approches in control unit design
1] Hardwired Control Unit Design
2] Micro-programmed Control Unit Design

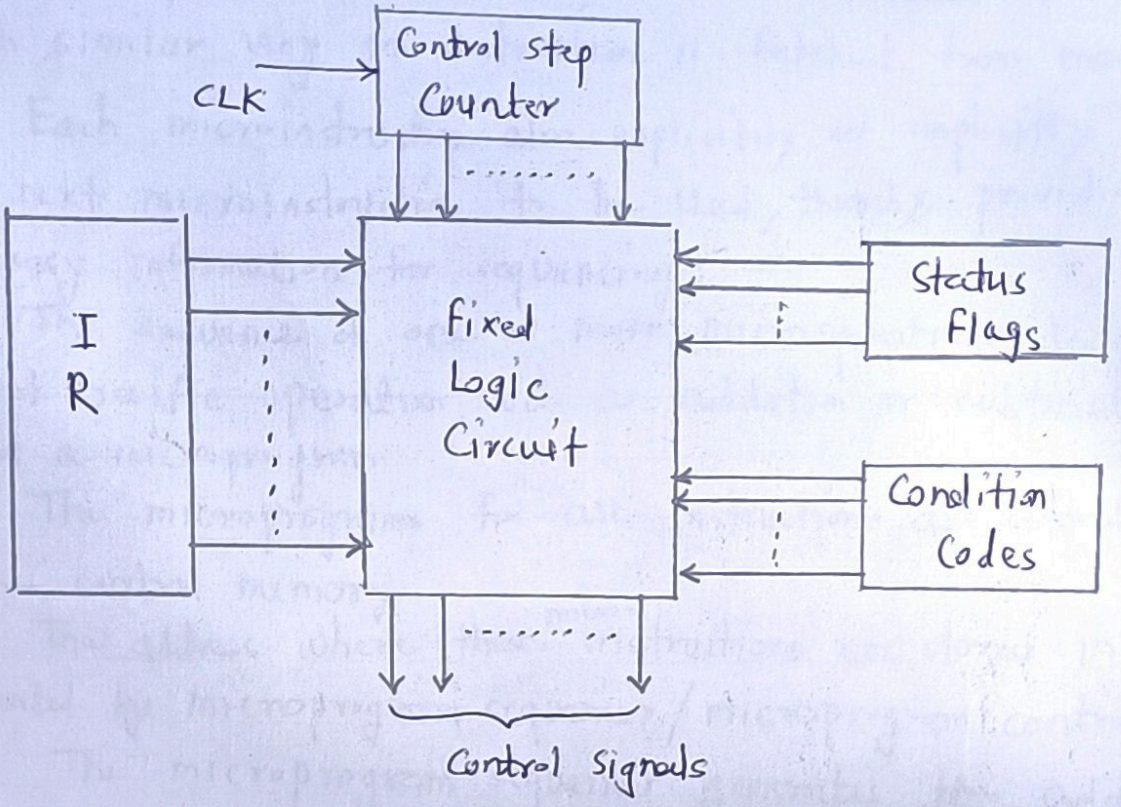
Hardwired Control Unit Design:

In the hardwired implementation the control units use fixed logic circuits to interpret instructions and generate control signals from them.

The hardwired control unit design involves various trade-offs between the amount of hardwired used, the speed of operation and the cost of the design process itself. In hardwired control unit design there are diffent methods

1] State Table method
2] Delay Element method
3] Sequence counter method
4] Programmable Logic Array (PLA) method.

The fig. shows block diagram of a typical hardwired control unit



* Microprogrammed Control Unit

In this approche the control unit uses microprograms to interpret and execute instructions.

* Every instruction in a CPU is implemented by a sequence of one or more sets of concurrent micro-operations.
* Each micro-operation is associated with a specific set of control lines which, when activated, causes the microoperation to take place.
* Since the number of instructions & control lines is often in hundreds the complexity of h/w control unit is very high & hence it is costly & difficult to design.
* furthermore h/w control unit is relatively inflexible because it is difficult to change the design, if one wishes to correct design error or modify the inst. set.

Microprogramming is a method of control unit design in which the control signal selection and sequencing information is stored in a ROM or RAM called as control memory.

The control signals to be activated at any time are specified by microinstruction, which is fetched from CM in much similar way an instruction is fetched from main memory

Each microinstruction also explicitly or implicitly specifies the next microinstruction to be used, thereby providing the necessary information for sequencing.

The sequence of one or more microoperations designed to control specific operation, such as addition or subtraction is called a microprogram.

The microprograms for all instructions are stored in in the control memory.

The address where these micro-instructions are stored in CM is generated by microprogram sequencer / microprogram controller

The microprogram sequencer generates the address for microinstruction according to the instruction stored in the IR

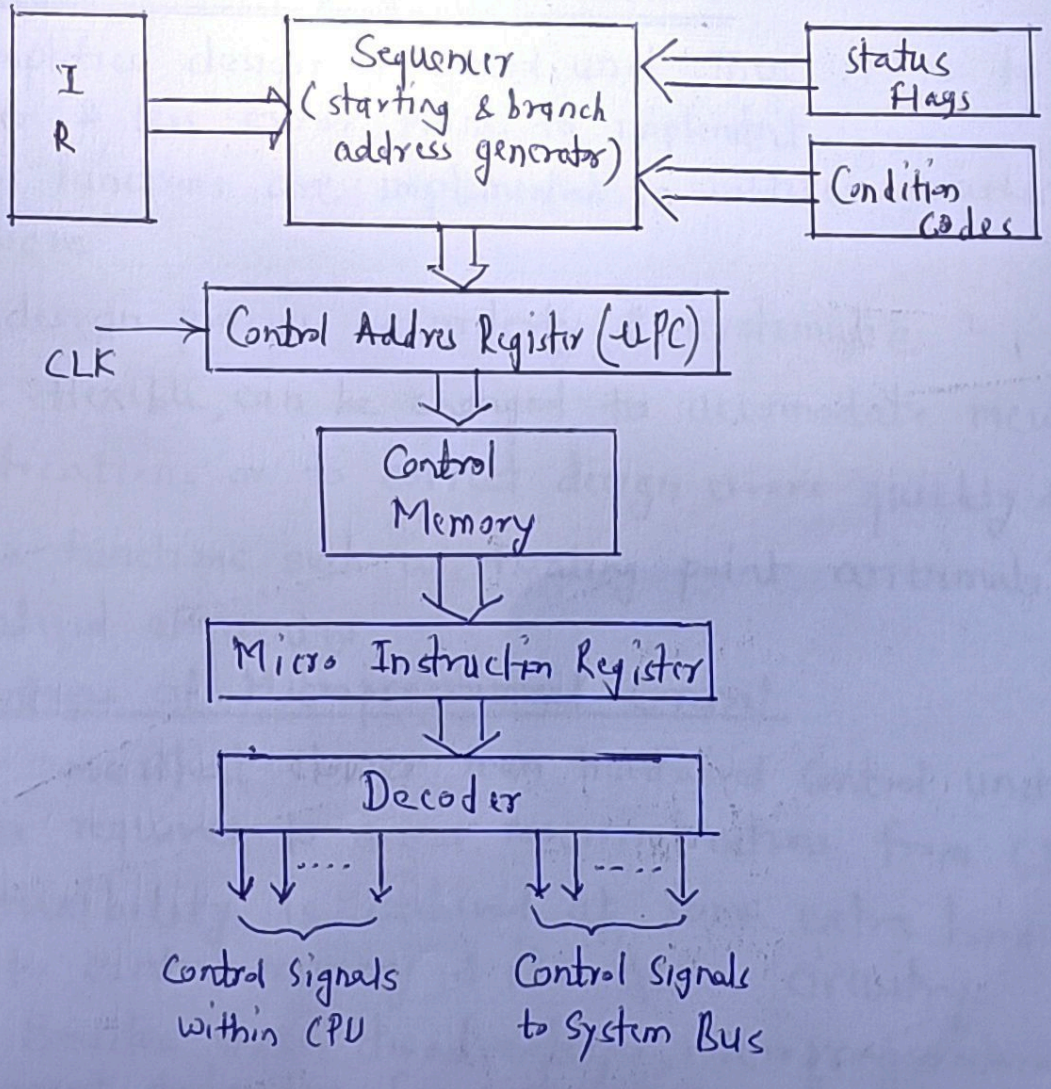


Fig shows the organisation of microprogrammed CU. Consist of control memory, control address register, micro struction register and microprogram sequencer.

This CU works as follows

1] The control address register holds the address of the next microinstruction to read.

2] When address is available in control address register, the sequencer issues READ Command to the control memory.

3] After issue of READ command, the word from addressed location is read into microinstruction register.

4] Now the content of microinstruction register generates cont signals & next addres information for the sequencer

5] The sequencer loads a new address into the control address register based on the next address information.

Advantages of Microprogrammed Control

1] It simplifies design of control unit. Thus it is both cheaper & less error prone to implement.

2] Control functions are implemented in software rather than hardware

3] The design process is orderly & systematic

4] More flexible, can be changed to accomodate new syste specifications or to correct design errors quickly & cheap

5] Complex functions such as floating point arithmetic can be realised efficiently.

Disadvantages of Microprogrammed Control

1] It is somewhat slower than hardwired control unit, becau time is required to access microinstructions from CM.

2] The flexibility is acchived at some extra hardware co due to control memory & its access circuitry

Besides these disadvantages, microprogramming is the dominant technique for implementing control units.

Difference between Hardwired CU & Microprogrammed CU

Hardwired Control	Microprogrammed Control
1] It is faster	1] It is slower
2] It is not flexible, to accomodate new system specifications or new instructions	2] It is more flexible, to accomodate new system specifications or new instructions
3] It is difficult to handle large/complex instruction sets	3] It is easier to handle large or complex instruction sets.
4] Design process is somewhat complicated	4] Design process is orderly & systematic
5] Very difficult to support operating system and diagnostic features.	5] Very easy to support operating system & diagnostic features.
6] It is used mostly in RISC microprocessors	6] It is used in mainframes & some microprocessors
7] ROM is not used	7] ROM of size 2K-10K by 20-400 bit microinst. is used.
8] It uses least chip area	8] It uses more chip area.

I/O Module & It's functions

A computer communicates with outside world by means of input output (IO) system.

The main function of I/O system is to transfer the information between CPU or memory and the outside world.

The important point to be noted here is I/O devices (peripherals) can not be connected directly to the system bus because —

1] A variety of peripheral devices with different methods of operation are available. So it would be impractical to incorporate the necessary logic within the CPU to control a range of devices.

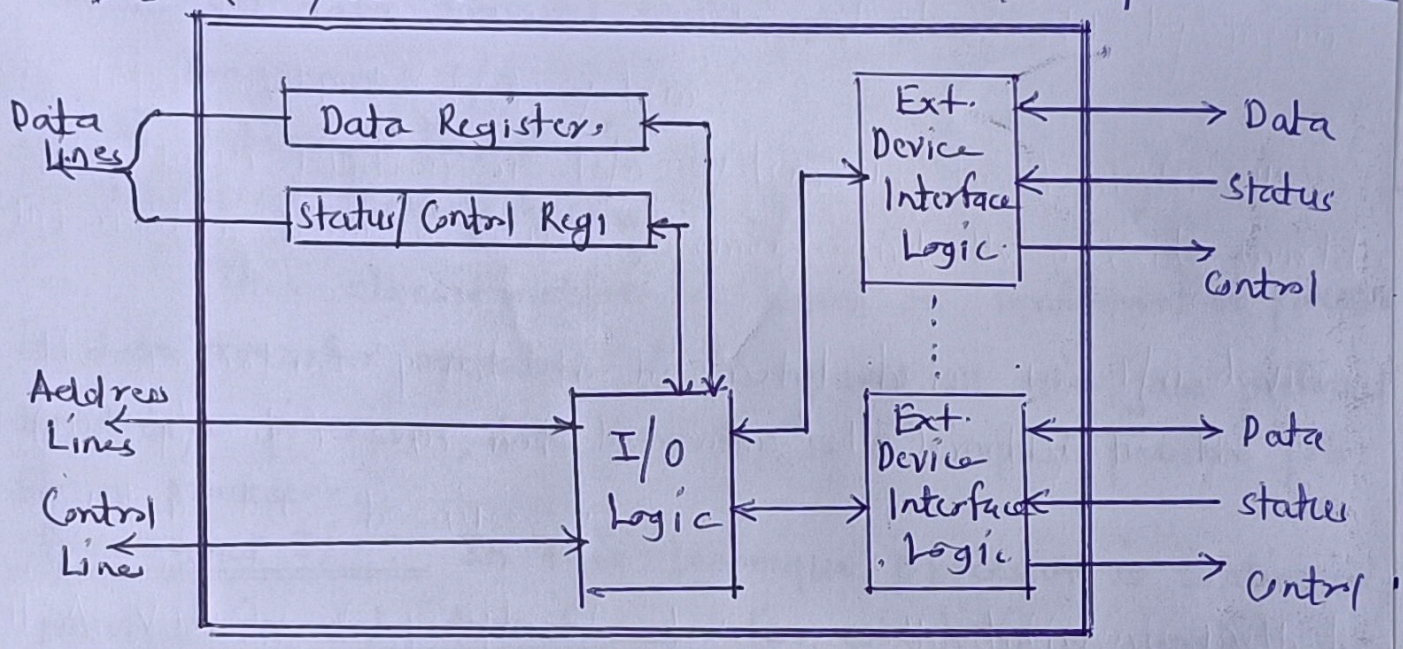
2] The data transfer rate of peripherals is often much more slower than that of the CPU or memory. So it is impractical to use a high speed system bus to communicate directly with the peripherals.

3] Generally, the peripherals used in a computer system have different ~~methods~~ data formats and word lengths than that of CPU used in it.

4] Generally CPU & Memory they are in IC form i.e (semiconductor technology) & most of the peripherals are electromechanical device & hence they differ in speed of operation.

So to overcome all these difficulties, it is necessary to use an I/O Module in between CPU and I/O device.

Block diagram of I/O Module.



Functions of an I/O Module

① **Control & Timing**: The I/O module co-ordinates the flow of traffic between internal resources and external devices by providing necessary timing & control requirements.

② **Processor Communication**: I/O module communicates with µp It accepts commands from processor. It exchanges data with processor. It also performs the status reporting job.

③ **Device Communication**: I/O Module also communicates with devices. This communication involves commands, status information & data

④ **Data Buffering**: It also performs data buffering job Data taken out memory is buffered into I/O module or data which is to be stored in memory is buffered into I/O module.

⑤ **Error Detection**: I/O module is also responsible for detection of errors & subsequently reporting errors to the processors. (eg paper jam, bad disk track)

I/O Data Transfer

Various I/O data transfer techniques are

i] Programmed I/O
ii] Interrupt driven I/O
iii] Direct Memory Access

This classification is based on involvement of processor in data transfer operation. As compared all other jobs performed by today's processor, data transfer is a simplest possible job for a processor.

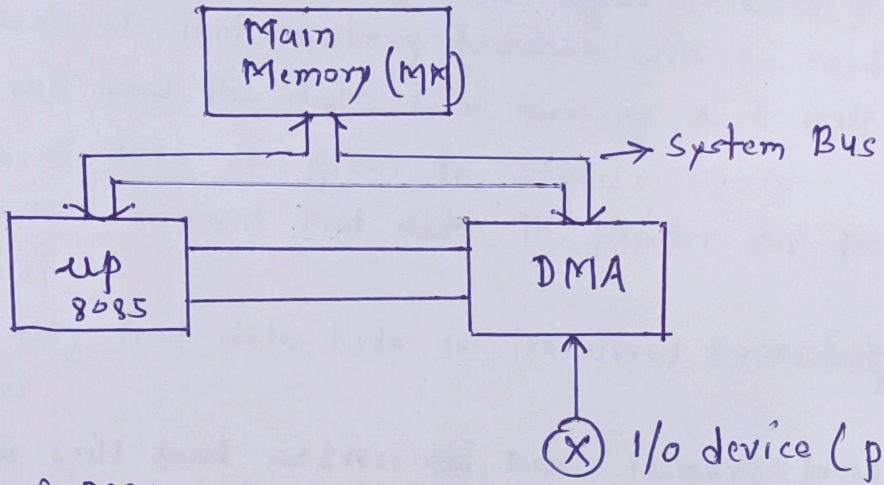
Programmed I/O: In this technique processor is continuously involved in data transfer operation which is a simplest job for any processor. Processor will not perform any other job while doing data transfer. It means valuable time of processor is utilised for doing simplest possible job. Data transfer is done by using IN & OUT instructions

Interrupt driven I/O: In this technique processor is not continuously involved but it is partially involved in data transfer operation. Whenever any I/O device connected to processor wants to perform data transfer, it will interrupt the processor. The processor will stop the current activity & it give go to do the data transfer & when the data transfer is over it will rejoin the original activity which he was doing initially. Thus, in this method processor is not continuously involved in data transfer, but it goes or performs data transfer only when it is interrupted & once it is over, it goes back to rejoin the original activity which it was performing

Thus processor is partially involved in data transfer & hence this method is better than programmed I/O method.

* DMA : Direct Memory Access

In this method processor is not at all involved in data transfer operation. It is required only at the begining (for getting control of the system bus) & at the end of data transfer operation (to transfer back control of the system bus)



Operation of DMA

- I/O Device (printer) is not directly connected to 8085 but it is connected to DMA
- DMA, 8085 (up) & Main memory (MM) are connected to each other by using system bus.
- When such a system is made ON up (8085) is bus Master & DMA is slave. 8085 (up) is master means system bus controlled or owned by 8085 (up)

Steps involved in data transfer from Memory to I/O device (printer)

1] Printer will make request to DMA for data transfer

2] As DMA is slave, it will forward that request to 8085 (up) by activating HOLD signal.

HOLD: It is the request made by DMA controller to 8085 (up) for getting control of the system bus

3) Upon received HOLD signal 8085 (up) will generate HLDA (hold acknowledge) signal & it is given to DMA

4) When this HLDA is received by DMA, it becomes master & 8085 (up) becomes slave [Now the control of bus is with DMA]

6] DMA will inform the requesting device (printer) about acceptance of its request

7] Now DMA will address of the first memory which is to be read to memory. by using system bus

8] Now DMA will send Memory Read Signal ($\overline{MEMR}$) to memory so that contents of that memory location will be read.

9] Now, he will read the data from memory & it will be placed on data bus & then it given to DMA.

10] Now DMA will forward that data to printer for printing by activating $\overline{IOW}$ signal.

11] In this way first data byte is transfered successfully from memory to printer.

11] Now DMA will send address of next memory location & it will generate $\overline{MEMR}$ signal to read the data

12] The data will be brought into DMA & then given to printer for printing by activating $\overline{IOW}$ signal.

13] This process is repeated till all the data bytes from memory gets transferred.

14] When all the bytes are over, the DMA will deactivate HOLD signal so that 8085 becomes master & DMA becomes slave.

When DMA is doing data transfer by using system bus, at that 8085 will not keep quite or remain idle but it will perform all other jobs which are not requiring the system bus

* Thus in DMA operation, DMA is performing data transfer at the same time processing is also doing some other jobs simultaneously

Thus concurrent operations are possible & hence DMA is the best method of doing the data transfer.

DMA is most preferred method for data transfer of large data files.

Interrupt structure of 8085

Interrupt: It is the disturbance to the normal execution of program in any processor/computer.

Classification of Interrupts in 8085

1] Hardware Interrupts and software interrupts
2] Maskable Interrupts and Non-maskable interrupts
3] Single Level Interrupt system & Multilevel interrupts
4] Vectored, Non-vectored & Auto Vectored Interrupts

* 8085 provide 5 Hardware & 8 Software Interrupts

Hardware Interrupt: The interrupt generated ~~from~~ by an external device connected to processor from outside is called as hardware interrupt.

8085 has 5 Hardware interrupts & these are

1. TRAP
2. RST 7.5
3. RST 6.5
4. RST 5.5
5. INTR

These interrupts will be received on Pin No. 6, 7, 8, 9 & 10 of 8085 respectively. The details of these interrupts are –

Type of Interrupt	Priority	Maskable/ Non Maskable	Type of Triggering	Vector Address
TRAP	1st (highest)	NM	+ Ve Edge	0024 H
RST 7.5	2nd	M	+ Ve Edge & level	003C H
RST 6.5	3rd	M	Level	0034 H
RST 5.5	4th	M	Level	002C H
INTR	last or least	M	Level	—

NM – Non-maskable
M – Maskable.

Software Interrupts

Interrupts which are generated by execution of any interrupt related instruction are called as software interrupts. They are generated internally.

There are 8 software interrupts in 8085 & these are.

Interrupt	Vector Address
RST 0	0000H
RST 1	0008H
RST 2	0010H
RST 3	0018H
RST 4	0020H
RST 5	0028H
RST 6	0030H
RST 7	0038H

Difference Between Hardware & Software Interrupts

Hardware Interrupts	Software Interrupts
1] It is an asychronous event which may or may not occur with respect to time	1] It is an synchronous event which always occurs with respect to time
2] It is generated by external devices (peripherals) connected to microprocessor	2] It is generatted internally by execution of any interrupt related instruction.
3] They have lower priority	3] They have highest priority
4] They are used for interfacing peripheral Devices	4] They are used for program debugging.
5] up may or may not generates interrupt acknowledge signal	5] up never generates interrupt acknowledge signal.
6] Ex. TRAP, RST 7.5, RST 6.5 RST 5.5 & INTR	6] Ex. RST 0, RST 1, RST 2, RST 3 RST 4, RST 5, RST 6, RST 7

Maskable Interrupts: Interrupts which can be avoided or made pending are known as maskable interrupts.

* Generally hardware interrupts are maskable.

Non-Maskable Interrupts: Interrupts which can not be avoided or made pending are known as non-maskable interrupts.

All software interrupts are non-maskable.

Difference Between Maskable & Non-maskable Interrupts

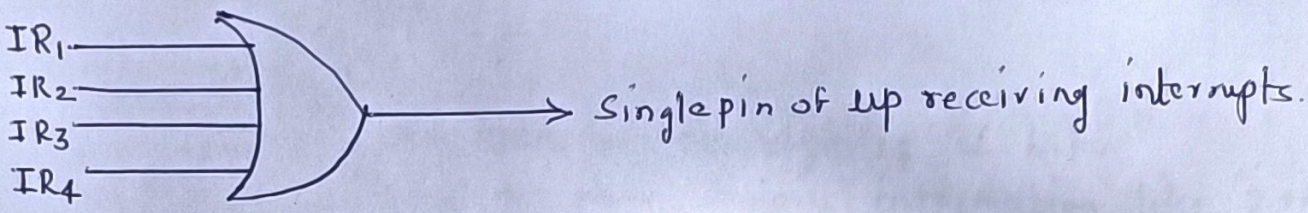
Maskable Interrupts	Non-Maskable Interrupts
1] Interrupts which can be avoided or made pending are known as maskable Interrupts	1] Interrupts which can not be avoided or made pending are known as non-maskable interrupts
2] They have lower priority	2] They have highest priority
3] They are used for interfacing peripheral devices	3] They are used for emergency situations like power failure, smoke detection
4] Generally ~~they are level triggered~~ h/w interrupts are maskable	4] Software interrupts are always ~~They are always edge triggered interrupts~~ maskable
5] up may or may not generates interrupt acknowledge signal	5] up never generates interrupt acknowledge signal.
6] Ex. RST 7.5, RST 6.5, RST 5.5 INTR	6] Ex. TRAP, RST 0, RST 1, RST 2 RST 3, RST 4, RST 5, RST 6 RST 7

* **Single Level & Multi Level Interrupt System**

Single Level Interrupt System

If only one pin is provided in the processor to receive the interrupts, then it is known as single level interrupt system.

If multiple interrupts are coming to such processor then in that case these interrupts are logically ORed together & the output of OR gate is connected to that single pin on up.



Multilevel Interrupt System

If more than one or multiple pins are provided on processor to receive the interrupts then it is called as multilevel interrupt system.

e.g. In 8085, five seperate pins (Pin No 6–10) are provided to receive 5 interrupts (TRAP, RST 7.5, RST 6.5, RST 5.5 & INTR) Hence it is a multilevel interrupt system.

✱ Steps Involved in providing seervice to a interrupt :–

When processor is executing its main program & in between any interrupt comes then –

① up will finish execution of current instruction (i.e instruction in hand)
② Before going to provide seervice to interrupt, up will store add. of the next instruction from main program (called as Return Address) on stack memory by performing PUSH Operation.
③ Now the program control is transfered from main program to Interrupt Seervice Routine (ISR).
④ up will start executing instructions from ISR & the last inst of ISR will be RET
⑤ When RET is executed, the program control is transferred back to main program & the return address is obtained back from stack memory by performing POP operation.
⑥ This return address is loaded into PC so that up will start executing remaining instruction from main program.

data & hence they

* Memory :

It is a storage medium or device.
It is used to store information.
The information can be data, programs, intermediate results or final results.

* Unit of Memory :

The smallest possible unit of memory is 1 bit.
The memory is used to store binary information like '0' or '1'
Group of four binary bits is called as Nibble
eg. 1100
Group of eight binary bits is called as 'byte'
eg. 11001100
The storage capacity of memory is always expressed in byte, Kilo byte, Mega bytes, Giga bytes etc.

1 KBytes = 2^{10} = 1024 bytes
1 MBytes = 2^{20} = 1,05,536 bytes
1 GByte = 2^{30} Bytes.

* Classification of Memory :

① Internal Memory & External Memory
② Primary (main) memory & Secondary Memory
③ Based on technology used — Magnetic Memory
Optical Memory
Semiconductor memory
④ Based on what type facity is available in memory
RAM
ROM
EPROM
EEPROM

Internal Memory : If the memory is present within the CPU (processor) it is called as internal memory
eg. Every processor is having a set of registers (register file) inside it. These registers are used for temporary storage of data & hence th

are called as internal memory of computer

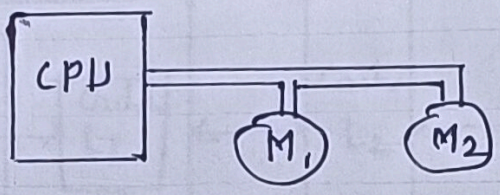
Apart from this now a day all the processor's (CPU) have inbuilt or internal cache memory.

As it is inbuilt or internally available hence it is called as internal memory or internal Cache.

The internal cache is also called as level 1 or L_1 Cache

External Memory:

If the memory device is connected externally to CPU (processor), then is called as external ~~pro~~ memory



As shown M_1 & M_2 are external memories. They are connected to CPU are using external bus.

Primary or Main Memory:

Primary memory is also known as Main Memory of computer (RAM)

It is used to store programs which are active in use. Basically it is used to store user programs. i.e. programs written by we people.

The have higher speed but their storage capacity is less as compared to secondary memory.

Secondary Memory

They are connected externally to CPU but they have huge storage capacity & they are slower in speed as compared to main memory (RAM)

The storage capacity of secondary memory is always in GB's of TB's

for eg. Hard disk.

This memory is used to store operating system or system programs & large data file which are not frequently used.

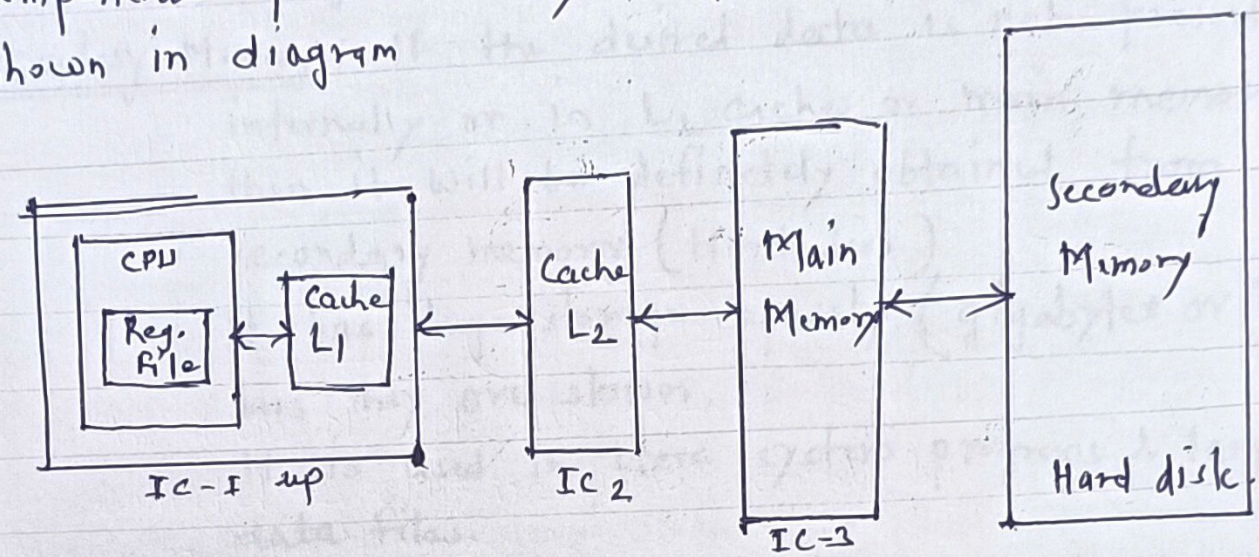
ype of Technology

Magnetic – Floppy, Hard disk, magnetic tape

Optical – CD

Semiconductor – RAM, ROM (in IC form)

Memory Hirarchy in Computer : The various memory components present in any computer system are as shown in diagram



* Internal Registers : They are present within CPU. They are used to store the data temporarily.
It is always better to have more no. of registers with larger in size.
They are the closest memory elements to CPU.

* Internal Cache (L_1) : It is present within or inside CPU
If the desired data is not present in L_1 cache processor will search it in internal cache (L_1)
The search from internal registers or L_1 cache is much more faster as it is internal communication.

* External Cache (L_2) : It is a high speed SRAM placed between CPU & main memory. If desired data is not available in internal registers or L_1 cache then CPU will try to get it from external cache or L_2 Cache.

Main memory: If the desired data in not available internally or in external Cache (L_2) the CPU will get it from external main memory.

Main memory (RAM) have generally larger storage capacity & it is used to store the programs which are active in use.

Secondary Memory: If the desired data is not present internally or in L_2 cache or main memory then it will be definetely obtained from secondary memory (Hard disk)

It has huge storage capacity (gigabytes or Tera bytes) bus they are slower.

It is used to store system programs & large data files.

Memory Characteristics / Properties to be considered while selecting a memory device.

① Cost of the memory:

Cost of the memory is not only the cost of storage medium but it includes the cost of accessing mechanism (Read/write heads)

For eg. in case of floppy, the cost of memory is cost of floppy disk + cost of floppy drive.

OR

In case of CD, the cost of memory is cost of CD + cost of CD drive.

The cost of memory is always expressed in dollars/bit

$$C = \frac{C}{S} \text{ dollars/bit}$$

C = cost of the complete memory system
S = storage capacity (total no. of bits)
c = cost of the memory.

It is always preferable to have lower cost of the memo

* Performance of Memory : It is measured in terms of sp

The speed of memory is measured in terms of "access tim

Read Access Time : The time required to read specific amount of information from memory is called Read Acce Time.

Less is the access time faster is the memory

* Write Access Time: The time required to write specific amount of information in memory is called Write Acces time.

Less is the access time faster is the memory

It is not necessary that read access time & write access time must be same. It depends on the type of storage medium (magnetic/optical) as well as type of accessing mechanism.

* Accessing Modes : It is the order or sequence in which information is accessed or stored in memory

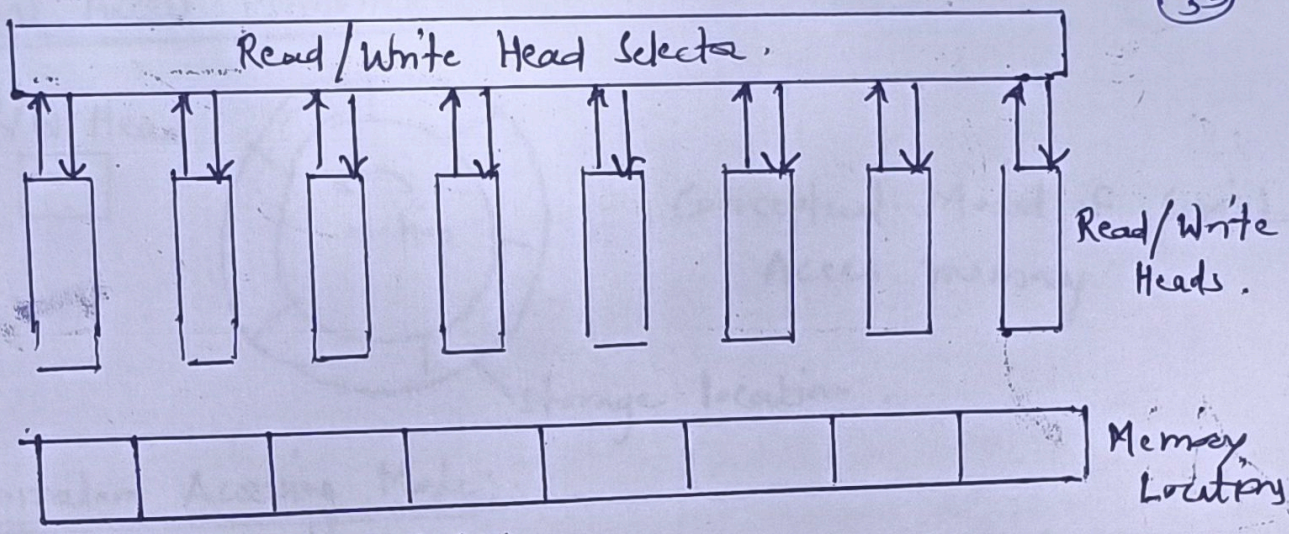
Random Accessing Mode:

If the information or locations can be accessed in any order and access time is independant of the location being accessed, then it is called as Random Accessing mode.

Semiconductor memories (memories in the form of IC) i.e. RAM, ~~ROM~~, they are random Access memories.

Each storage location of RAM can be accessed independantly. for this purpose a seperate access mechanism i.e. Read-Write head is required for each location.

Since every location has its own access mechanism the memory will be costlier but it will be faster.



Conceptual Model of RAM.

Serial or Sequential Access Memory

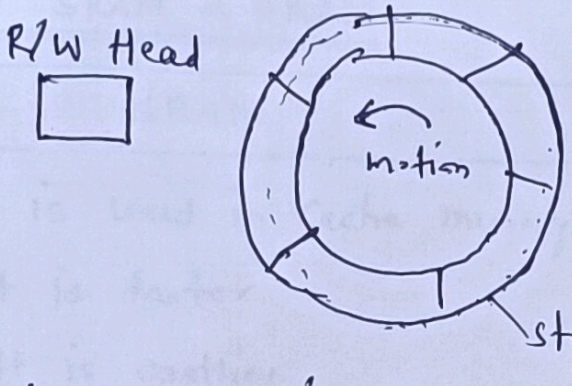
Memories whose locations can be accessed only in certain predetermined sequence are called as serial or sequential access memories.

Magnetic disk, magnetic tape & optical memories are examples of serial access memories.

In serial access memories the access mechanism i.e (Read Write Heads) is shared by storage locations & must be assigned to different locations at different time by moving the stored information or the read write Heads or both.

In most of serial memories the storage locations are moving or rotating around a closed path or track. A particular location can be accessed only when it passed the fixed R/W head. Hence the time required to access a particular location depends on its position relative to the R/W head.

Serial access memories are much more slower in speed & also their cost is less as they are using a single R/W head.



Conceptual Model of Serial Access memory

* Semiradom Accessing Mode:

Memory device like magnetic hard disk & CD-ROM contain many rotating tracks.

If each track has its own R/W head then tracks can be accessed randomly, but access within the track is serial. In such cases the accessing mode is semirandom

* Data Transfer Rate or Bandwidth of memory

The maximum amount of information that can be transferred to or from the memory per unit time is called data transfer rate or bandwidth.

It is measured in bits per second or words per second.

Kbyte, Mbytes or Gbytes these are the units used.

* Reliability:

It indicates how safe is data stored in memory. Reliability is measured in terms of mean time before failure (MTBF)

Memories with no moving parts have higher reliability than memories having moving parts.

Even in memories where there are no moving parts, the reliabity problem arises. pesticularly when very high storage densility or data transfer rates are used.

Error detecting & error correcting codes can increase the reliability of the memory.

Memory Retention:

① SRAM & DRAM

SRAM	DRAM
1) It is used in Cache memory	1) It is used in main memory
2) It is faster	2) It is slower
3) It is costlier	3) It is cheaper
4) Storage Capacity is less	4) storage capacity is larger
5) Physical size is larger.	5) Physical size is smaller.
6) Power consumption is more	6) Power consumption is less.
7) flips flops are used to store information	7) Capacitors are used to store information.
8) Refreshing circuitary is not required	8) Refreshing circuitary is required.

② Volatile & Non volatile memories:

Magnetic Tapes

Historically, **magnetic tape** was the first kind of secondary memory. A computer tape drive is analogous to a home tape recorder: a 2400-ft-long tape is wound from the feed reel past a recording head to the take-up reel. By varying the current in the recording head, the computer can write information on the tape in the form of little magnetized spots.

Figure 2-15 shows how information is organized on a magnetic tape. On a computer with 8-bit bytes, each frame contains 1 byte, plus an extra, redundant bit, called a parity bit, to improve reliability. Typical recording density is 1600 frames (bytes) per inch (denoted as 1600 bpi), which means that a frame takes up less than 1/1000 of an inch. Other common densities are 800 bpi and 6250 bpi. After a tape drive has finished writing a **physical record** (a sequence of frames), it leaves a gap on the tape while slowing down. If the program writes short physical records on the tape, most of the space will be wasted in the gaps. By writing physical records that are much longer than the gap, the tape utilization can be kept high.

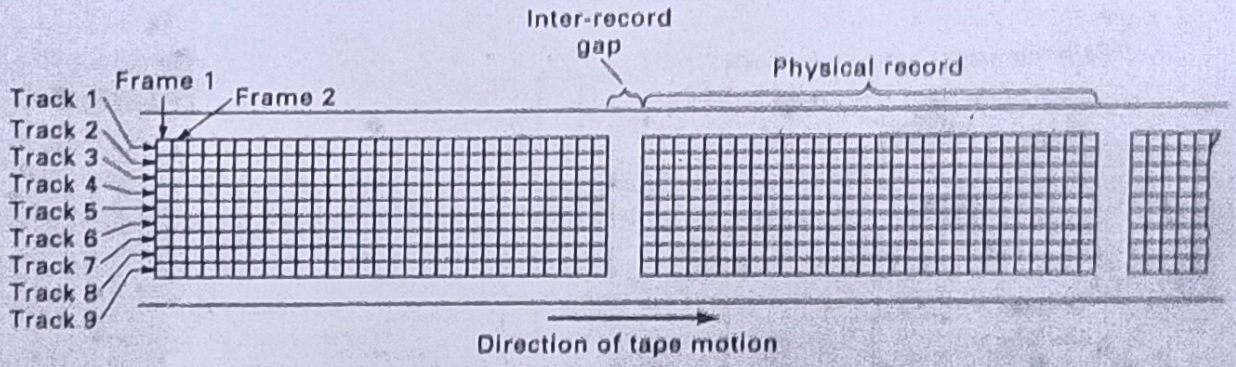


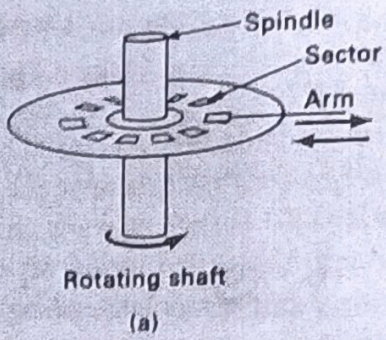
Fig. 2-15. Information on a magnetic tape is recorded as a sequence of rectangular bit matrices.

Magnetic tapes are sequential-access devices. If the tape is positioned at the beginning, to read physical record n, it is first necessary to read physical records 1 through $n - 1$, one at a time. If the information desired is near the end of the tape, the program will have to read almost the entire tape, which may take several minutes. Forcing a CPU that can execute millions of instructions per second to wait 200 sec while a tape is advanced is wasteful. Tapes are most appropriate when the data must be accessed sequentially.

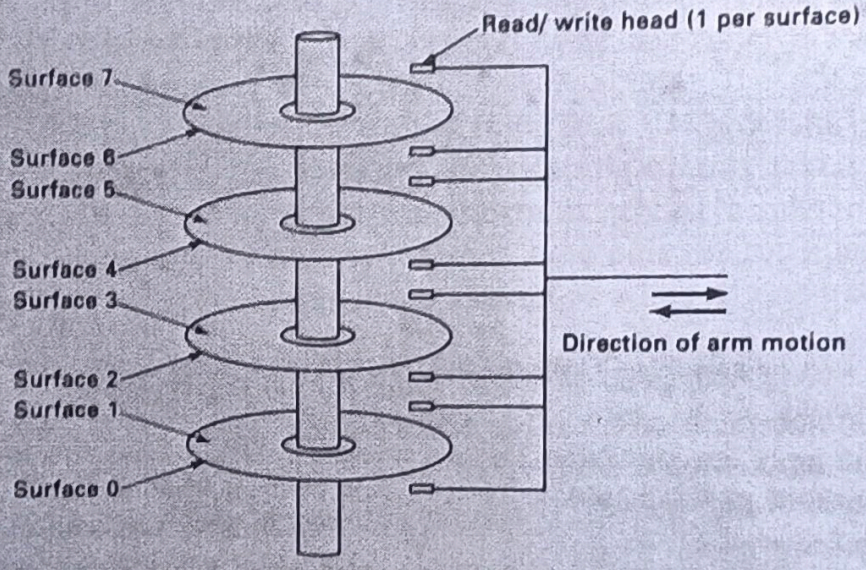
Magnetic Disks

A **disk** is a piece of metal, ranging from about 5 to 10 inches in diameter, to which a magnetizable coating has been applied at the factory, generally on both sides (see Fig. 2-16). Information is recorded on a number of concentric circles, called **tracks**. Disks typically have between 40 and a few hundred tracks per

surface. Each disk drive has a movable head that can be moved closer to, or farther from, the center. The head is wide enough to read or write information from exactly one track. A disk drive often has several disks stacked vertically about an inch apart. In such a configuration the arm will have one head next to each surface, all of which move in and out together. The radial position of the heads (distance from the spindle) is called the **cylinder**. A disk drive with n platters will have $2n$ heads, hence $2n$ tracks per cylinder.



(a)



(b)

Fig. 2-16. (a) A disk with one platter. (b) A disk with four platters.

Tracks are divided into **sectors**, normally between 10 and 100 sectors per track. A sector consists of a certain number of bytes, usually 512.

To specify a transfer, the program must provide the following information: the cylinder and head, which together specify a unique track, the sector number where the information starts, the number of words to be transmitted, the main memory address where the information comes from or goes to, and whether information is to be read from the disk into memory or written from memory to the disk.

Instruction Pipelining

Instruction Pipelining: Prefetching the next instruction in advance while execution of current instruction is going on is called as instruction pipelining.

The execution of any instruction is not carried out in a single stage, but requires no. of stages.

The various stages of instruction pipe are

1] Fetch the Instruction [FI]
2] Decode the Instruction [DI]
3] Fetch the Operand [FO] (It is optional)
4] Execute the Instruction [EX]
5] Write Back or Store the Result [WB]

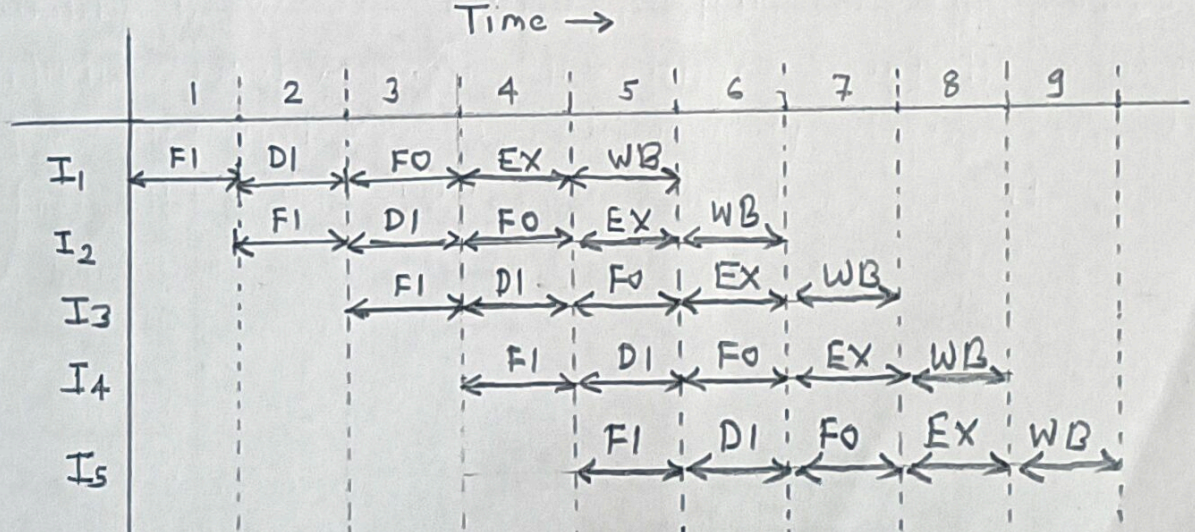
Effect of Instruction pipelining on performance of Computer

A] Performance Without Pipeline

Consider a program of 5 lines which is to be executed. Assume that each line requires equal amount of time i.e 5 seconds for execution. Thus for execution of this 5 line program without pipeline total time required for execution will be $5 \times 5 = 25$ seconds.

B] Performance with Pipeline:

Consider the same program of 5 lines & each line is requiring 5 seconds for execution. Each line requires 5 seconds & execution

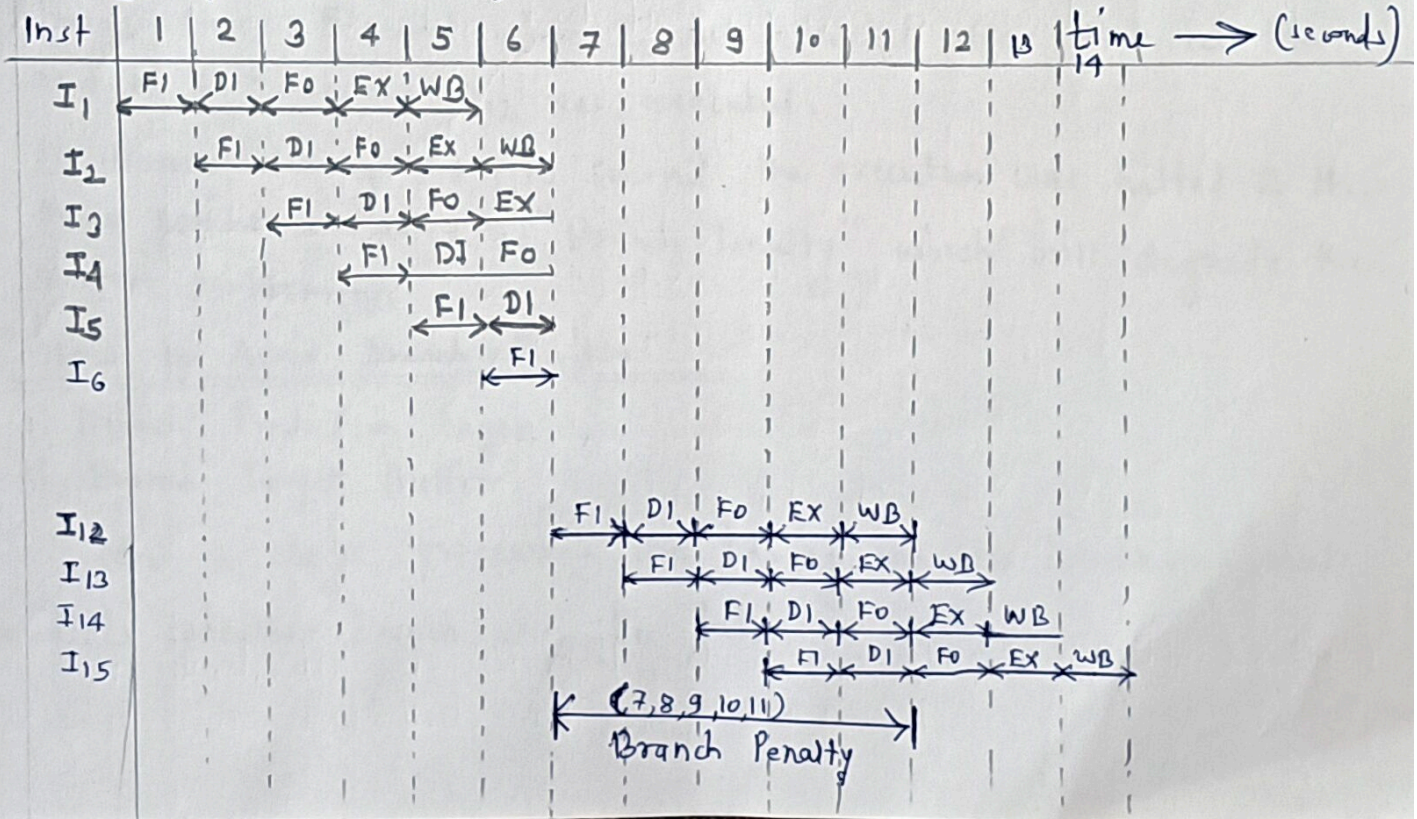


of Each line is carried out in 5 stages (FI, DI, FO, EX, WB) & it is assumed that each stage requires 1 seconds.

During 1st Second, Instruction 1 (I_1) will be taken inside pipeline
During 2nd Second Instruction 2 (I_2) will taken inside & I_1 will go to next stage i.e DI.
During 3rd Second I_1 will move FO (3) stage, I_2 will move to DI (2) stage, at the same time I_3 will be accepted in & it will continue
At the end of 5th second the first inst. (I_1) will get completely executed. At the end of 6th second I_2 will get completely executed & At the end of 9th second I_5 means last line of the program will get executed.
Thus the entire program of 5 lines will get executed in only 9 secs which was earlier taking 25 seconds to execute (without pipelining)
Thus by using instruction pipelining we are saving 25 − 9 = 16 Secs. which is huge amount of time & thus perfomance will improved tremendously by using concept of Instruction Pipelining.

Effect of Branch Instruction on Performance of Inst. Pipeline

No doubt perfomance is improved by use of instruction pipelining, but if there is a branch instruction in between in the program then the perfomance is degraded as shown below. Consider the similar situation. Each line requires 5 seconds. [within one line each stage require 1 seconds] Now the program is of 15 lines & I_5 line is Branch inst. (Jump at line No. 12)



As shown in the diagram

1) During 1st second I_1 will be taken inside
2) During 2nd second I_1 will go to DI stage & I_2 will enter inside
3) This process will continue and at the end of 5th second, the first line I_1 will get completely executed.
4) During 6th second (if we see vertically in colmn 6), I_2 is completely executed, I_3 is in EX stage, I_4 is in FO stage, I_5 is DI stage & I_6 is just taken inside.
5) I_5 is in DI stage & when I_5 decoded, it meaning is jump at instruction No. 12 (I_{12}) by bypassing instructions in between.
7) Thus I_3, I_4, I_5, I_6 will remain pending & processor will start executing instructions from I_{12} onwards.
8) I_{12} will get executed at the end of 11th second & I_{15} the last line I_{15} will get executed in 14th second.
9) Normally in pipeline system we have to just wait for first 5 seconds (pipeline gets full) & after that for every second there is execution of one instruction continuously.
10) But In this case the I_2 was the instruction which was completely executed at the end of 6th sec. & after that I_3, I_4, I_5, I_6 were kept pending & instead of that I_{12} was taken inside, then I_{13}, I_{14}, I_{15}
11) Finally I_{12} was completely executed at the end of 11th seconds.
12) Thus after 6th second where I_2 was executed, after that at the end of 11th second I_{12} was executed.

It means during 7, 8, 9, 10 second the execution was halted & this time period is called as "Branch Penalty" which will degrade the system performance.

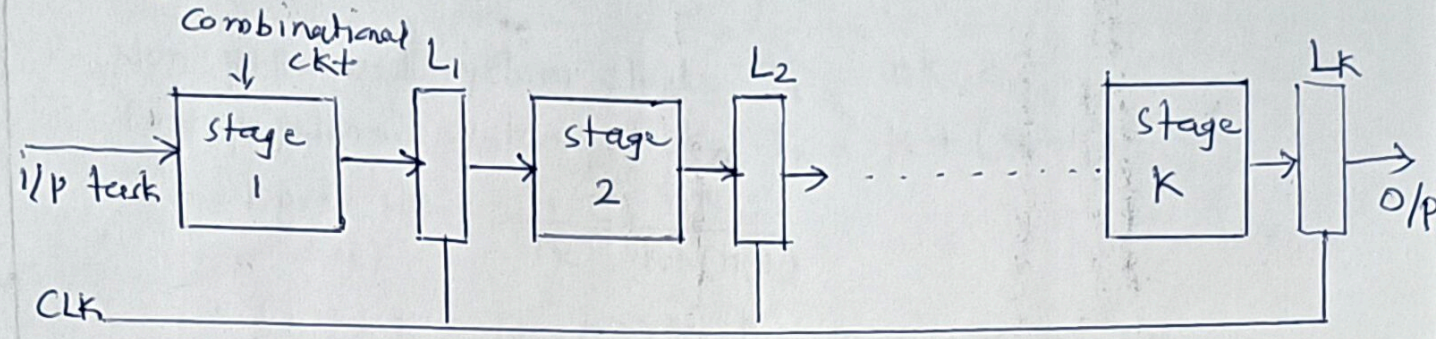
How to Avoid Branch Penalty

1] Branch Prediction Logic
2] Branch Target Buffer

Now a days processors are having inbuilt hardware that predicts wheather branch will be taken or not.

Performance of a linear pipeline

* Linear pipeline is designed by using a number of pipeline stages & each of these pipeline stage is purely a combinational circuit.
* Two successive stages in a pipeline are seperated by using a high speed latch.



* Result generated by stage 1 is copied to the Latch & it goes for processing the next job

* Performance of a linear pipeline is measured in terms of
  1) Clock frequency / clock period
  2) speed up
  3) Efficiency of pipeline
  4) Throughput of pipeline

Clock frequency / Clock Period:

The clock frequency of linear pipeline is

$$T = \max_{i=1}^{K} \{t_i\} + t_L$$

where t_L = Latch period.

While defining clock frequency of pipeline maximum time delay is taken into account as per the equation

Reciprocal of clock period gives clock frequency of pipeline

$$\text{Clock frequency of pipeline} = \frac{1}{\text{Time Period } (T) \text{ of pipeline}}$$

Speed up:

Ratio of time required for 'n' tasks on a non pipelined system to the time required on a pipelined system is called speed up.

No. of tasks = n
No of pipeline stages = K

∴ Non pipelined system clocks $T_1 = nK$
∴ for pipelined system clocks $T_K = K + (n-1)$

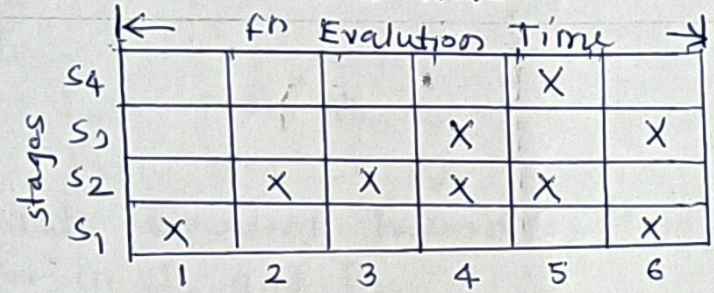
∴ Speed up $(S_K) = \frac{T_1}{T_K} = \frac{nK}{K+(n-1)}$

$S_K \to K$ for $n \gg K$ for maximum speed up.

Efficiency:

It is the fraction of busy time in the total time

$$\eta_{(E)} = \frac{\text{Busy Time}}{\text{Total Time}}$$



$$\eta_{(E)} = \frac{9}{24}$$

OR

Efficiency is speed up in the pipeline system due to 1 stage

$$\eta_{(E)} = \frac{S_K}{K} = \frac{nK}{K+(n-1)} \times \frac{1}{K} = \frac{n}{K+(n-1)}$$

$\eta_{(E)} \to 1$ as $n \to \infty$

Throughput:

It is the average number of results generated per unit time.

If No. of tasks are = n
No. of stage are = K

∴ No. of clocks required = K+(n-1)
∴ Total time = [K+(n-1)]T
∴ Throughput = $\frac{n}{[K+(n-1)T]}$

∴ Throughput = $\frac{\eta_{(E)}}{T}$

Pipeline Hazards

There are two types of pipeline hazards
1] Data dependant Hazards
2] Control Hazards.

Data Dependant Hazards

These hazards are dw to inter instruction data dependency Broadly, data dependent hazards are caused by resource usage conflicts among successive instructions

Normally such conflict is due to common register between two instructions.

In case of data dependent hazards unless the earlier instruction goes beyound the possible resource conflict stage, the next or subsequent instruction is not allowed in the processing stage.

It means 2nd instruction can be fetched as well as decoded but execution is delayed untill first instruction gets totally executed

In 8086 Processor

I_1 : MOV AX, BX
I_2 : MOV CX, AX

Here data dependent hazard is triggered due to use of AX register in I_1 and I_2

In case of data dependent hazards, linear operation of pipeline is sometimes suspended.

There are three types of data dependent hazards
1] Read After Write (RAW)
2] Write After Read (WAR)
3] Write After Write (WAW)

* RAW Hazard

In 8086 Processor

I_1 : ADD AX, BX : Write
I_2 : SUB CX, AX : Read

with respect to AX register data dependent hazard is triggered in RAW

A RAW Hazard may occur when instruction 'J' attempts to read some data object which has been modified by previous instruction 'I'

D(I) defines the domain of Instruction I i.e. a set of resource data objects that may affect the execution of Inst 'I'

R(I) defines the range of instruction I i.e a set of resource data objects that can be modified by execution of Instruction 'I'

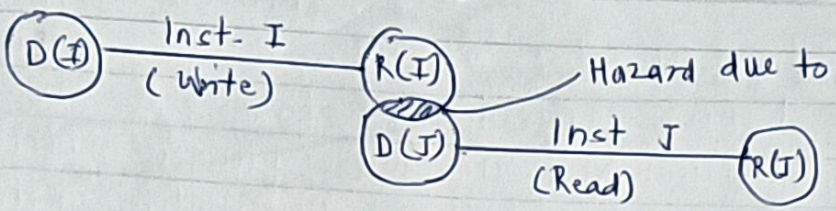


Illustration of RAW hazard condition

D(I) & D(J) define domain of instruction I & J respectively. (Domain is input)

R(I) & R(J) defines range of instruction I & J respectively (Range is output)

* WAR Hazard :-

In 8086 system

I_1 : ADD AX, BX : Read

I_2 : SUB BX, CX : Write

With respect to BX Register data dependency hazard is triggered in WAR

A WAR hazard can occur if instruction 'J' attempt to modify some data object which is read by instruction 'I'

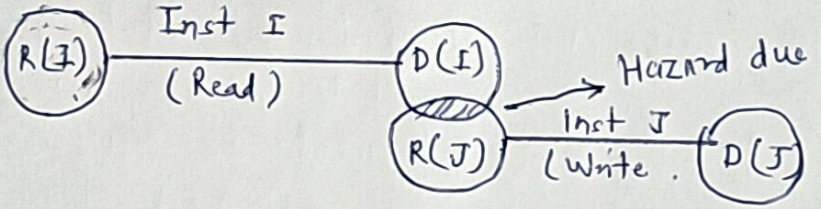


Illustration of WAR Hazard Condition

* WAW Hazard :-

In 8086 system

I_1 : MOV AX, BX : Write

MOV AX, CX : Write

With respect to AX register data dependent hazard is triggered in WAW

In WAW hazard can occur when both I & J instruction modify the same data object.

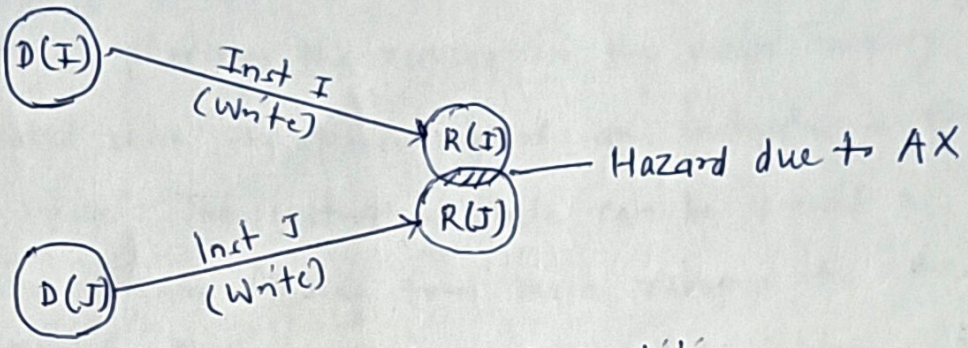


Illustration of WAW Hazard Condition

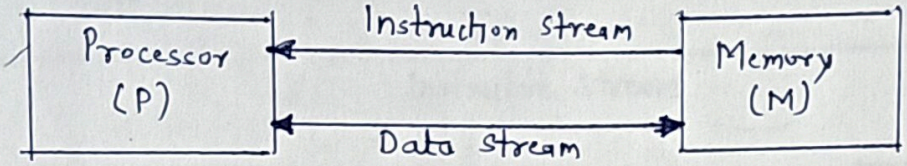
Control Hazards :

Control hazards are caused due to execution of

i] Branch Instruction
ii] Subroutine CALL instruction or
iii] In processing of Interrupts.

FLYNN's Classification of Parallel Processing Systems.

A typical central processing unit (CPU) operates by fetching instructions & operands from main memory, executing the the instructions & placing the results in the main memory. The steps associated with the processing of an instructiom form an "Instruction cycle". The instructim cycle can be viewed as forming an instructim stream flowing from main memory to the processor while the operands form another stream i.e data stream flowing to or from the processor as shown.



Instruction & Data Streams in a simple computer.

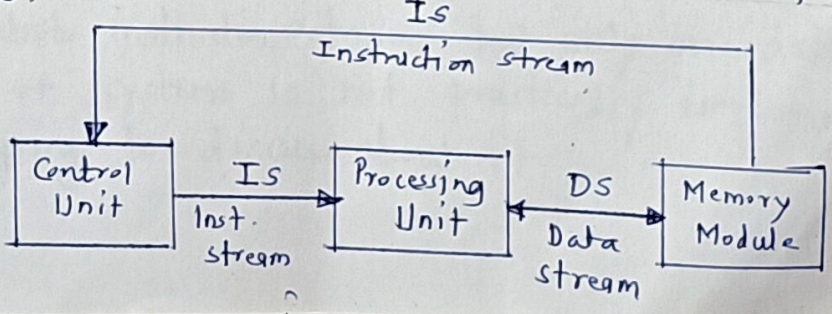
Based on the number of instructions streams & number of data data streams present Michael Flynn's has divided computers into 4 groups.

1] Single Instruction Stream - single data stream [SISD]
2] Single Instructim Stream - Multiple Data Streams [SIMD]
3] Multiple Instruction Stream - Single Data Stream [MISD]
4] Multiple Instruction Stream - Multiple Data Stream [MIMD]

(1) SISD

In these systems there is only one instructim stream & only one data stream.

Most conventional machines with only one CPU containing a single arithmatic & Logic unit (ALU) capable of only of scalar arithmatic fall into this category. SISD computers & sequential computers are thus synonymous.

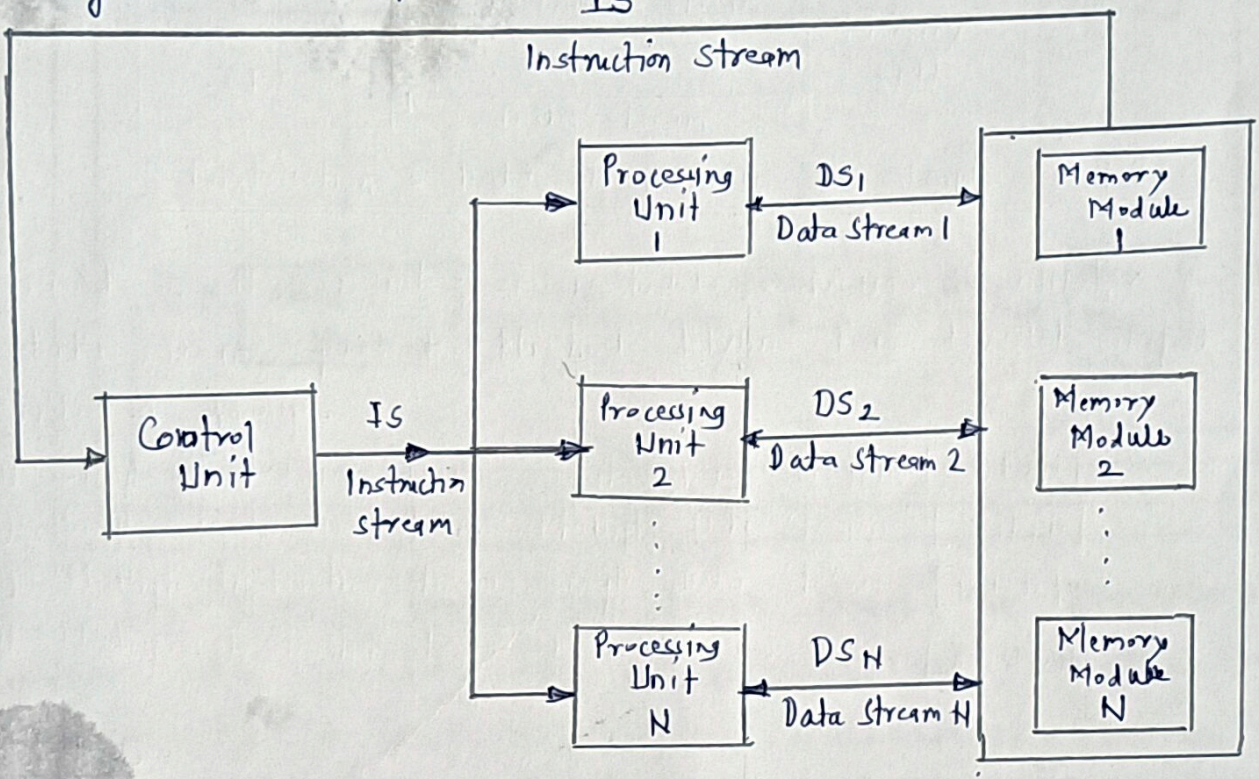


In SISD computers instructions are executed sequentially but may overlap in their execution stages (Pipelining). They may have more than one functional unit but all the functional units are controlled by a single Control Unit.

SIMD

They have single Instruction stream but Multiple Data streams. This category of computers corresponds to "Array Processors" They have multiple processing / execution units but only one control unit.

Therefore, all processing / execution units are supervised by the single control unit.



Here, all the processing units are receiving the same instruction from the control unit.

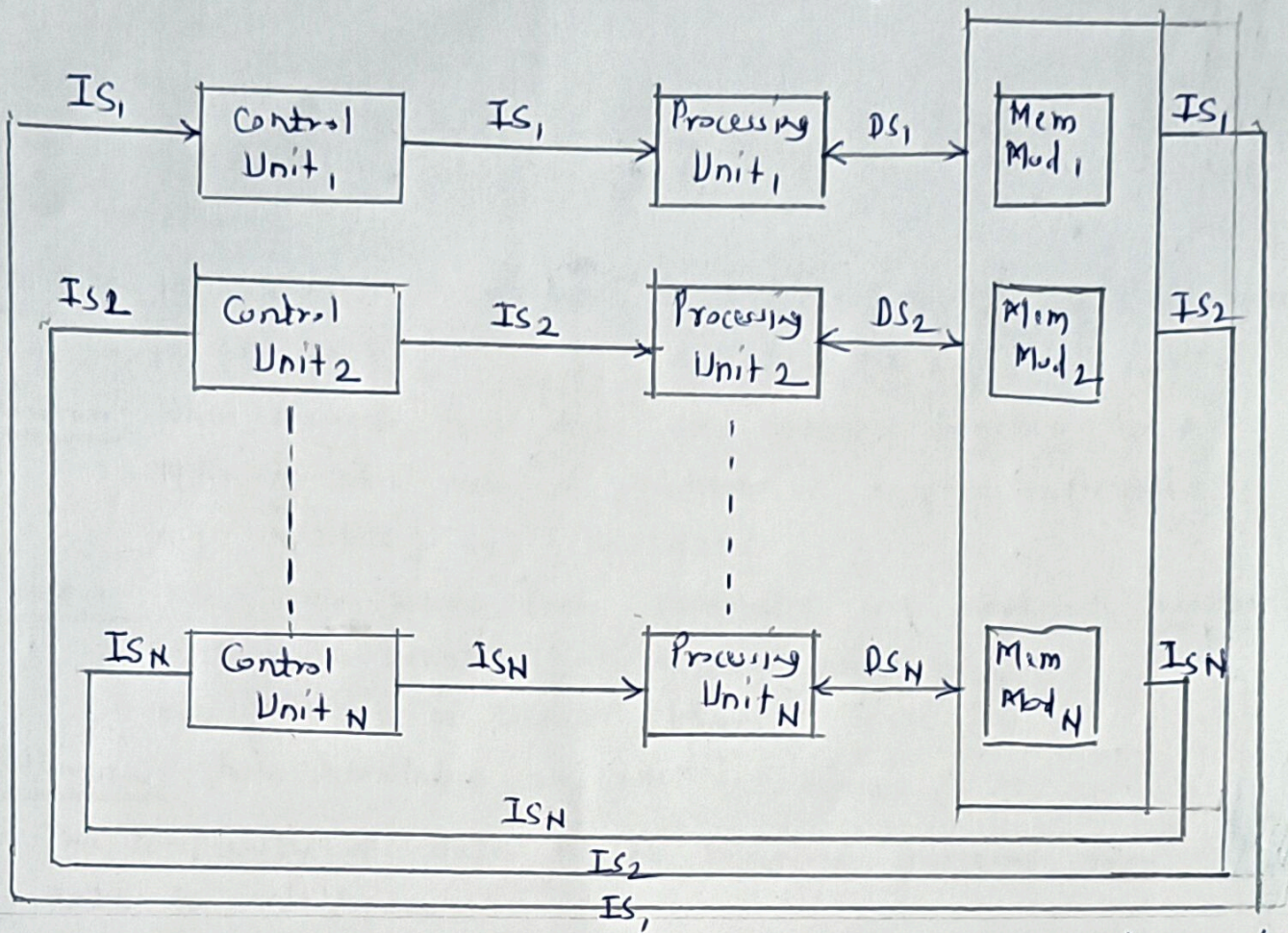
But each processing unit will operate on different data sets from distinct data streams.

MISD

They have multiple instruction streams but only one data streams This category of systems is not practically implemented & we are not going to discuss about it.

MIMD :

They have multiple instruction streams & multiple data streams. Most multiprocessor system & multi-computer system can be classified in this category.



In MIMD, there are more than one processing unit having the ability to execute several programs simultaneously. MIMD Computer implies interactions among the multiple processing units because all memory streams are derived from the same data space shared by all the processors.

Parallel Processing & its Applications

Parallel Processing: It is an efficient form of information processing which emplies exploitation of concurrent events in computing process.

Parrellel Processing is applicable to –

i) Data
ii) Information
iii) Knowledge
iv) Intelligence.

Data: It is collection of alphabets, newmericals or special characters
e.g. { P, N, E } { S, 4, H, 2, R, *, O, I }

Information: When elements from 'data' are arranged together by applying some rules of grammer it becomes information.
e.g. { PEN } or { SHRL420 }

Knowledge: When some words from information are combind together to make a seentence, it becomes information.
e.g. PEN is of BLACK COlour.

Intedligence: from knowledge we get intelligence.

The complexity of data to be processed increases from data to intelligence.

Applications of Parallel Processing:

(A) Modelling & Simulation

i) Weather Modelling
ii) Ocenography & Astrophysics
iii) Socioeconomics & Goverment Use

1) Weather modelling:
It is used for i) Short range forecast of weather
ii) Long Range hazard predictions (floods / T-sunami / Earth quake)

In India weather modelling is carried out by using 'PARAM' series of supercomputers designed at C-DAC (Center for Development of Advanced Computing). A series of super computers models like PARAM 8000, PARAM 9000, PARAM 10000 is developed by C-DAC.

[II] Oceanography & Astrophysics

It is used for climate Prediction over ocean & more specifically for 4 application listed below.

i] Climate Predictive Analysis

ii] Fishery Management

iii] Ocean Resource Exploitation

iv] Coastal Dynamics & Tides

The computers or systems which are used for implementing these applications are "Cyber-205" & "ILLIAC-IV" array processors.

[III] Socioeconomics & Govt. Use

It is used for

i] Econometrics

ii] Social Engineering

iii] Govt. Census / Public Opinin Polls

iv] Crime Control & Investigation

v] Modelling of National / World Economy.

These applications are implemented using CDC Scientific computers.

CDC → Control Data Corporation.

(B) Engineering Design & Automation

It is used in

① Finite Element Analysis: FEA is used for the design of huge structures like dams, bridges, ships, supersonic jet engins etc.

* The designing of such things needs a sequence of differential equations to be solved
* On a traditional / conventional machine to solve those differential equations it takes longer time. Hence parallel structures like array processor or Vector processor are used to reduce the processing time.
- systems which are implementing such type of applications are
  CDC Star-100 Vector Processor
  Cyber-205 Vector Processor.

② Computational Aerodynamics

It is used for designing of aerodynamic structures like JET engines.

System Design for Computational Aerodynamics i.e (NASF) Numerical Aerodynamic Simulation Facility is used.

It is developed by Burrough's Corporation & CDC

It is used for simulation of complete aircraft design.

③ AI & Automation:

It is used for

1] Image Processing
2] Pattern Recognition
3] Computer Vision
4] Speech Understanding
5] Machine Inference
6] CAD - Computer Aided Design
7] CAM - Computer Aided Manufacturing
8] office automation
9] Design of intelligent Robots
10] Design of Expert Systems.

In all the applications stated above a system should be developed with maximum LIPS (Logical Inferences Per Second)

One such system is CRAY-1 from CRAY Research Laboratory. To carry out these applications the processing power required is 100 MLIPS to 1 GLIPS

④ Remote Sensing Applications

* Remotely Sensed Earth resource data (via Satellite) is analysed by using parallel structures like an array processor MPP (Massively Parallel Processor)
* This analysis is helpful for forestry, Agricultural use & for water resource exploitation.

⑤ Medical, Military & Research Applications

It is used in

1] Computer Assisted Tomography
2] Genetic Engineering
3] Weapon Research & Defence

CISC & RISC Computers

* CISC Architecture:

* The Intel 80X86 and Motorola's 680X0 processors are the most commonly used CISC processors.
* They provide complex instruction sets having powerful instructions for direct implementation of high level language operations and program sequencing control features
* Execution of such instructions is quite complex and it is usually implemented by microprogrammed control.
* CISC architecture provides only few on chip registers & therefore most of the instructions have to refer main memory.
* This increases the time required for execution of instructions.
* As CISC architecture provides large number of instructions, they also provide many complex addressing modes & hence resulting into complex control unit.

* RISC Architecture:

* RISC is an acronym for "Reduced Instruction Set Computer"
* Today's processor such as Intel i860, SPARC, MIPS R3000 are examples of RISC architecture
* In most of RISC processors they use hardwired control unit.
* In RISC architecture it provides very few instructions & most of the instructions are 32 bit instructims.
* As there are less number of instructions, there are only few addressing modes.
* As the instructims are less & and they are simple in nature, this architecture requires simple decoding circuitary.
* It provides large number of on chip registers & hence the processor need not refer main memory for most of the instructions.

Thus because of simple instructions & less number of inst. & large number of internal registers RISC architecture is much more popular now a days.

Difference between CISC & RISC Computers

RISC	CISC
i) Simple instructions requiring only one cycle for executions	i) Complex instructions requiring multiple cycles for execution
ii) Very few Instructions	ii) Large number of instructions
iii) Fixed instruction formats	iii) Variable format instructions
iv) Verf few instructions refer main memory.	iv) Most of the instructions refer main memory
v) Instructions are executed by hardware	v) Instructions are executed by microprogram.
vi) Multiple register sets are available	vi) A single register set is used (only few registers)
vii) Very few addressing modes are used.	vii) Many addressing modes are used.
viii) Decoding circuitary is simple	viii) Decoding unit is more complex
ix) Highly pipelined.	ix) Not pipelined or less pipelined.
x) Complexity is in the compilier	x) Complexity is in microprogram

Concurrent Accesses to Memory

Memory Contention

A memory module can handle only one access request at a time. Hence when several processors request the same memory module, it causes memory contention.

This contention is resolved by an arbiter permitting the requesters to proceed one at a time. However the processors that wait to complete their memory cycle waste their time in busy waiting. This results in loss of performance

Communication Contention:

When several requesting processors can not complete their access due to the limitation of the interconnection network, the contention occurs and such contention is called communication contention.

Hot Spot Contention:

When several processors repeatedly access the same memory location, it gives rise to hot spot contention.

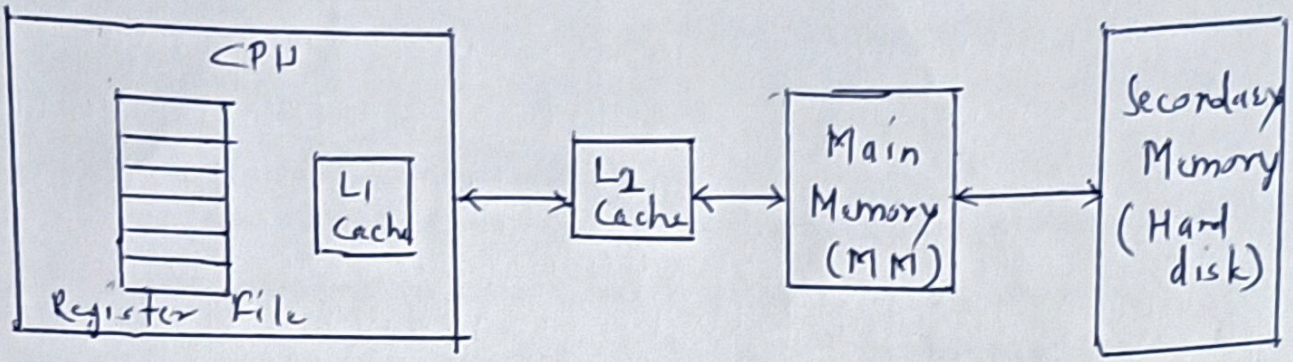
This is because it creats hot spot in either the memory module or the interconnection network that get overloaded with the requests and creat a bottleneck for the system performance.

Techniques for Reducing Contention

1] Local Memories
2] Better Interconnection network
3] Cache memory
4] Memory allocation.

Hierarchical Memory Organisation in Computer System :

Inside the computer system there are variety of memory elements. All these elements are organised or placed in hierarchical manner as shown in the diagram



1] Register file / Internal Register

Each processor contains a set of internal registers known as register file. These register used for temporary storage & hence they act as internal memory of computer.
It is always better to have large number of registers with larger size.
They are the closest memory element for microprocessor. i.e when up needs something he will approch first to internal registers.

2] Internal Cache / L1 Cache

Apart of from internal registers now a days each processor has inbuilt or internal cache which is known as L_1 Cache.

3] External Cache / L_2 Cache : Apart from internal cache there is external Cache which is a high speed SRAM placed between up & main memory

4] Main Memory

It is made up with DRAM technology & size of main memory is decided by number of add. lines.

5 Secondary Memory : It is much more larger in size & generally made up with magnetic technology i.e hard disk

Cache Memory

It is a high speed static RAM (SRAM) placed between Processor & the main memory.

Contents of Cache memory

The cache memory contains frequently used instructions and frequently used data items from main memory.

The instructions and data items which are required again & again by the processor are copied into cache memory.

Size of the Cache Memory

As compared to main memory size (storage capacity) of cache memory is much more smaller than memory. Also the cache memory is made up with SRAM technology while main memory is made up with DRAM technology.

Difference between SRAM & DRAM

SRAM	DRAM
1) It is used in Cache memory	1) It is used in main memory
2) It is faster	2) It is slower
3) It is costlier	3) It is cheaper
4) storage capacity is smaller	4) storage capacity is more
5) Physically size is much more larger	5) Physical size is smaller.
6) It consumes more power.	6) It consumes less power
7) Flip flop is used for information storage	7) Capacitors are used for information storage
8) Refreshing circuitary is not required	8) Refreshing circuitary is required.

Principle of Operation of Cache Memory

It is observed that same information from concecutive memory location is required again & again in order to executive program loop. Most of the program that run on PC concist of such loop. Characteristics of such programs can be explained with a term called "Locality of Reference" It has two terms —

1] Temporal Locality
2] Spatial Locality

Classification of Cache Memory

1] Internal & External Cache
2] Unified & split up cache

External Cache: If cache memory is connected externally (outside) to processor then it is called as external or Level 2 (L_2) cache.

Internal Cache: If cache memory is present within the processor (inbuilt) then it is called as internal or inbuilt or Level 1 (L_1) cache

Unified Cache: If the cache memory is used to store both instructions as well as data items, then it is known as unified cache memory.

Split up Cache: If two seperate cache memories are used, one for storing only instructions & one for storing only data then it is known as split up cache.

I Cache: The cache memory which contains or stores only instructions is called Instruction or I Cache

D Cache: The cache memory which contains or stores only data is called Data or D Cache.

* 80386 is the first processor in which cache memory is used. In this processor cache was connected externally (L_2) & it was unified cache.
* 80486: is the first processor in which both external (L_2) as well as internal (L_1) cache is used. Both the cache memories were unified.
* Pentium: is the first processor in which both external (L_2) and internal (L_1) cache is used. But both the cache memories were in split up form.

Performance of Cache Memory:

The performance of cache memory is measured in terms of "Hit Ratio"

$$\text{Hit Ratio} = \frac{\text{No. of Hits}}{\text{No. of Hits} + \text{No. of Miss}}$$

OR

$$\text{Hit Ratio} = \frac{\text{No. of Hits}}{\text{Total No. of Memory references made by CPU}}$$

The maximum possible value of hit ratio = 1
Hit Ratio is generally expressed in percentage & max. possible value is 100%. But practically 100% hit ratio is never acchived. It is always better to have higher/larger value of Hit Ratio

Hit : If processor makes refereftce of a desired thing & if it is found in cache memory, then it is called as Cache Hit

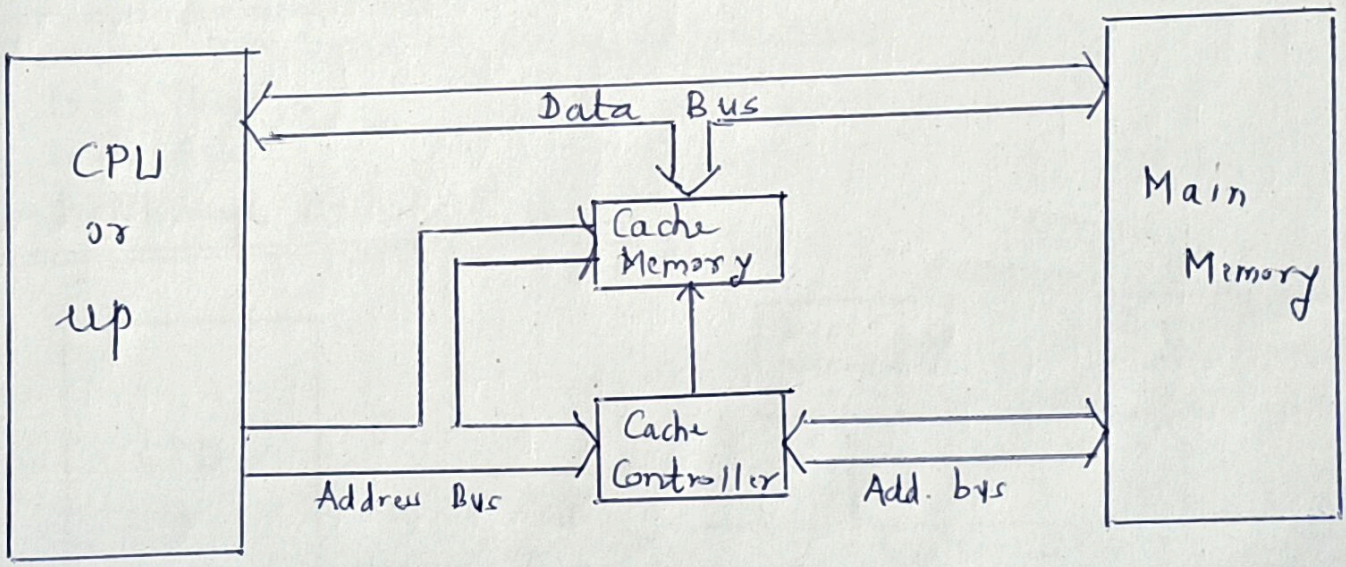
Miss : If processor makes reference of a desired thing & if it is not found in cache then in that case processor has to go to main memory, it is called as Cache Miss.

Example: Processor need 100 things, 80 times he got the desired things in cache & only 20 times he went to main memory.

∴ No of hits = 80
∴ No. of miss = 20

$$\therefore \text{Hit Ratio} = \frac{80}{80+20} = \frac{80}{100} = 0.8 \text{ or } 80\%$$

Working of Cache Memory



* Cache memory is always connected between CPU & main memory
* Cache memory always comes with Cache Controller
* When Microprocessor or CPU will start read operation, the Cache controller will check whether the desired information is present in cache or not.
* If the desired information is present in cache then it is given to CPU by using data bus & it is called as Read hit

If the cache controller determines that the desired thing is not present in cache then it is obtained from main memory (misses) The desired information from main memory is given to CPU by using data bus and also copied into cache memory.

Factors on which Hit Ratio / Performance Depends.

1] Cache memory size
2] Cache Architecture
3] Cache memory Organisation / memory mapping
4] Cache update / Write policy

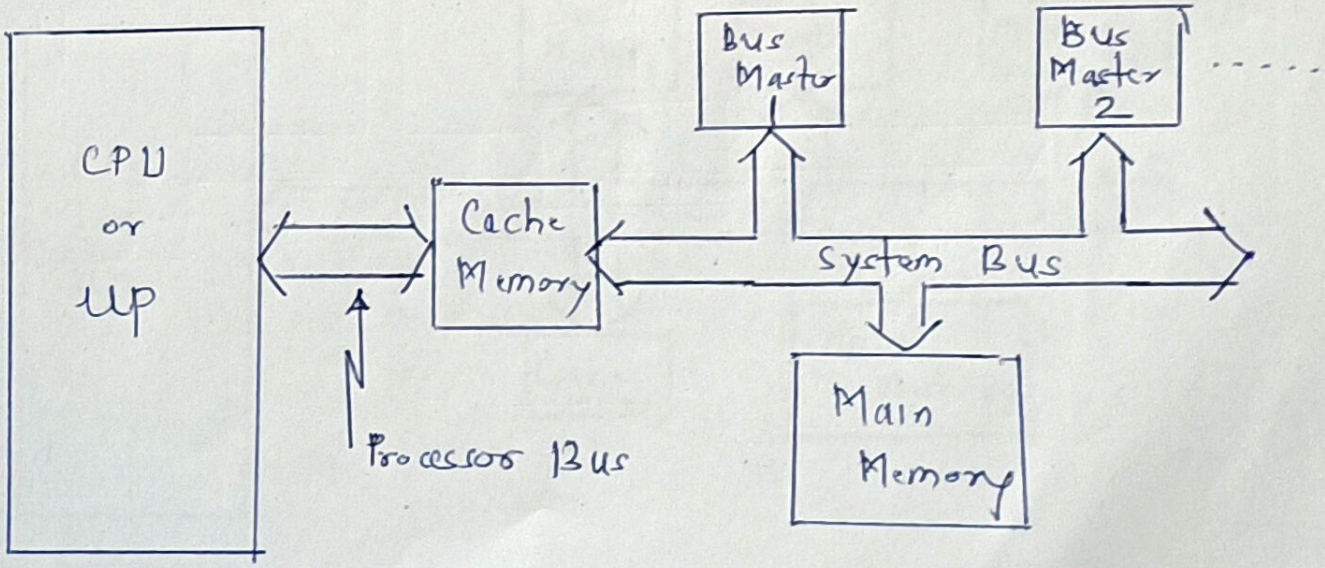
Cache Memory Size:

Hit Ratio depends on size (storage capacity) of cache memory Larger is the size of cache more will be the hit ratio. Because when cache size will increases the chances of getting the desired thing will increase. That means no. of hits will increase & ultimately hit ratio will increase. But we can increase the size of cache memory much more because of the physical size & cost. Hence a suitable combination of smaller size of Cache & larger size of main memory is to be used.

Cache Architecture

There are two types of Cache Architectures
1] Look Through
2] Look Aside

Look Through Architecture



* In this architecture two seperate buses are used. i.e processor bus & system bus.
* When CPU needs something, he will go the cache memory by using Processor bus & if the requested information is present in cache memory it will be immediately given to CPU by using data bus
* When CPU is accessing cache memory it uses processor bus & and at that time system bus is totally free.
* As shown in diagram all other bus masters are connected to main memory by using system bus & finally main memory is connected to Cache memory.
* When CPU is accessing Cache memory by using Processor bus at that time any other bus master can access Main memory. Thus concurrent operations are possible in this architecture which will improve the performance.

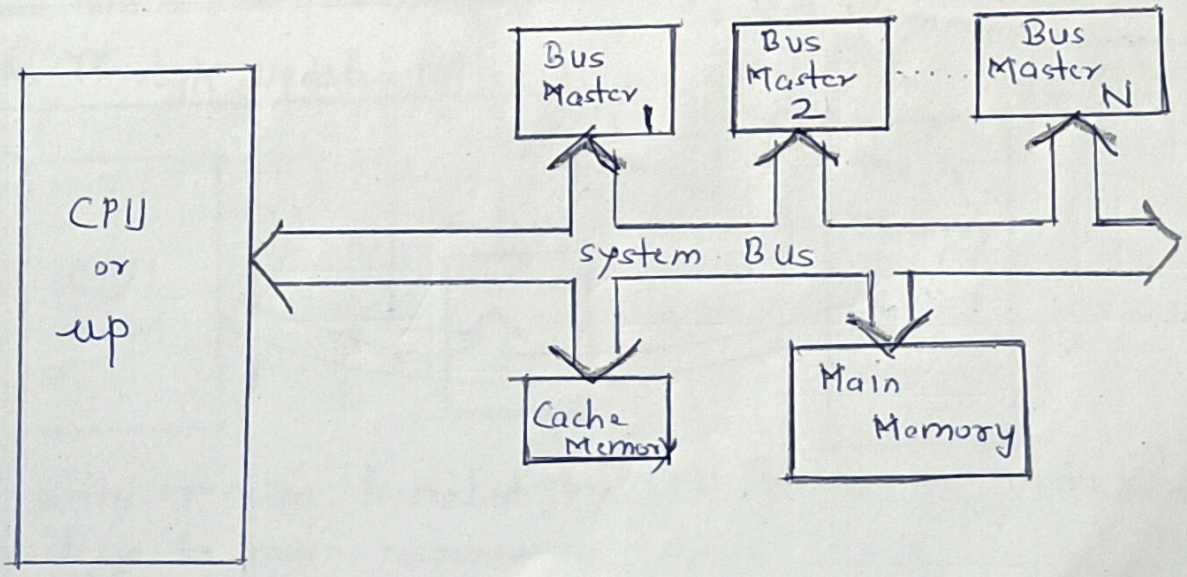
Advantages

1] Concurrent operations are possible

Disadvantages

1) Because of two seperate buses design is complex & costlier
2] In case of cache miss this method exhibits Look up Penalty i.e extra time component to check cache memory as well as main memory (one by one) in case of cache miss.

Look Aside Cache Architecture



In this method Cache memory main memory & all other bus masters are connected to CPU by using a common bus i.e system bus.

The cache controller (not shown in fig) monitors each memory request to see if cache memory contains copy of required information. The cache controller sits aside for this monitoring & hence the name is given as look-aside architecture.

When processor needs something, he will check in cache mem & if it is found in cache (cache hit), he will get it from cache. If it is not found in cache (cache miss), it will be obtained from main memory.

In case of Cache hit the main memory will start the access which is actually not required. Therefore there is unnecessary memory precharge cycle. This means that any other bus master needing access to memory must wait until the precharge cycle has completed.

Advantage

1] Simple design & hence less costly

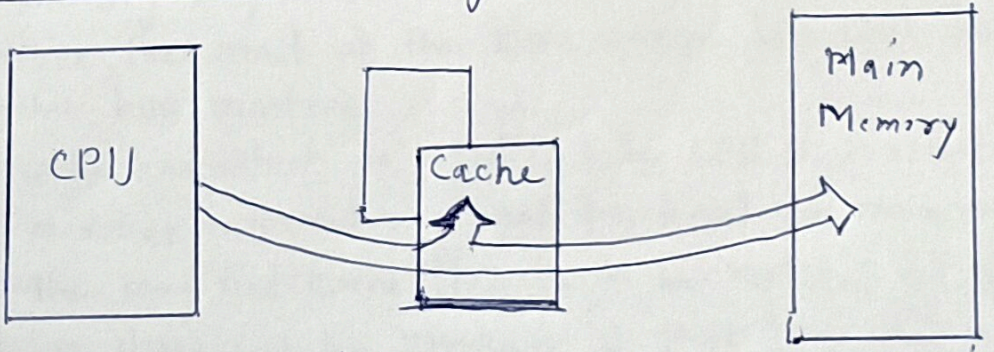
2] No Look up penalty i.e better response time in case of cache miss

Disadvantage

No concurrent operations are possible as all bus masters and up share a common system bus.

Types of update/Write Policies

1] Write Through Update Policy

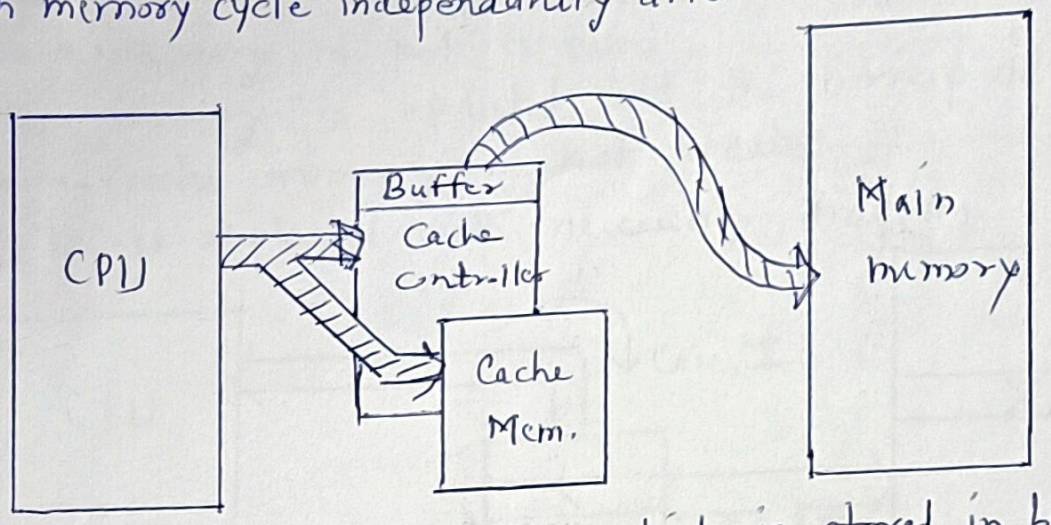


Each write operation initiated by CPU passes the information immediately to main memory as shown.

In short, main memory is immediately updated, therefore it

ontains always fresh & valid data
This implementation is very simple & effective but results in poor performance since each write operation must access slower main memory.

* Buffered Write Through / Posted Write
* When a write operation occurs in buffered write through method the cache controller stores the entire write operation in buffer while putting it into cache
* Therefore, there is no wait state. The controller completes the main memory cycle independantly after some time



* The updated information in cache which is stored in buffer is transferred to main main memory through system bus when any other bus master is not using it (Assuming look through Archi)
* The other bus masters connected to system bus are not allowed to access those main memory locations whose modifications are stored in buffer.

* Write Back Policy:
* This method improves overall system performance by updating the main memory only when it is necessary to update
* Therefore, for most of the time system bus will be free for use by other bus masters.
* The cache line which is modified by CPU is marked as modified/dirty
* This modified information is not transferred to main mem. immediately
* When the modified cache line is to be replaced by main memory line, which is demanded by processor at that time, that modication in cache line is transfered to corresponding location of main memory.
* This policy may encounter cache inconsistancy problem when main memory location is updated by one of the bus master which is already modified in cache by CPU.

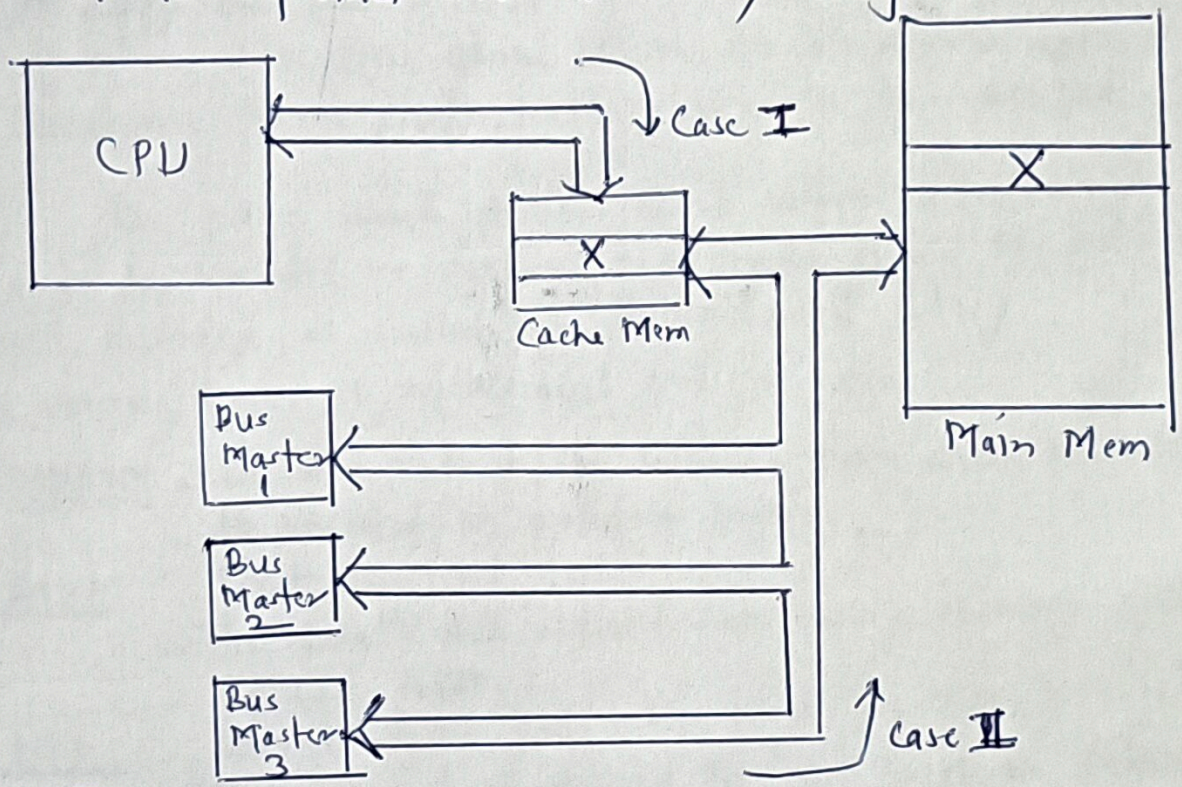
Cache Consistency / Coherency

If we want proper operation of cache subsystem, then contents of cache and main memory must be exact duplicate. There are several instances in which locations stored in cache memory are update but the corresponding location of main is memory is not updated & it contains old or stale information.

There are two possible conditions for this cache inconsistancy problem

1) Cache memory is updated but the corresponding location from main memory is not updated.
2) Main memory is updated but the corresponding location from cache memory is not updated.

It is explained with necessary diagram



Case I

Line X from Cache memory is updated by CPU while the same line from MM is not updated. This line X from MM migh be reffered by any other bus master connected to system bus (This information is old or stale)

Case II

Line X from main memory is updated by one of the bus master connected to system bus (MM is updated), while the same location which is mapped into CM is not accordingly updated (cache contains old or stale information)

In either of these 2 cases either MM or CM is updated while its duplicate contains state information. This problem is called inconsister

Replacement Policies used in Cache memory

* Whenever CPU needs something from memory, it will first check for into cache memory.
* If it is found in cache memory (cache hit), it will be directly provided to CPU
* But if it is not found in cache memory (cache miss), then it is obtained from main memory.
* The desired data from main memory is not only given to CPU but at the same time it is copied into cache mem.
* If there is space available in cache memory, then this data is copied into vacant locations of cache memory.

* But if cache memory is full i.e. no space is available to copy the data from main memory, in that case question will arise about where to store the data or which locations from cache memory are to be vacated.

The policy ~~which~~ that decides about which cache memory locations are to be vacated or replaced to provide space for data from main memory, is called as Replacement Policy

The various types of replacement policies are

1] RANDOM : In this method the locations from cache memory will be vacated in random manner.

2] FIFO : First In First Out
The data which entered into cache memory will be replaced first

3] LIFO : Last in first Out (Also known as FILO)
The data which entered in last will be taken out first

4] LRU : Least Recently Used
The data from cache memory which is least recently used or data which is not used for long time from cache memory will be replaced first because chances of using that in future will be very very less

LRU is the best Replacement Policy.

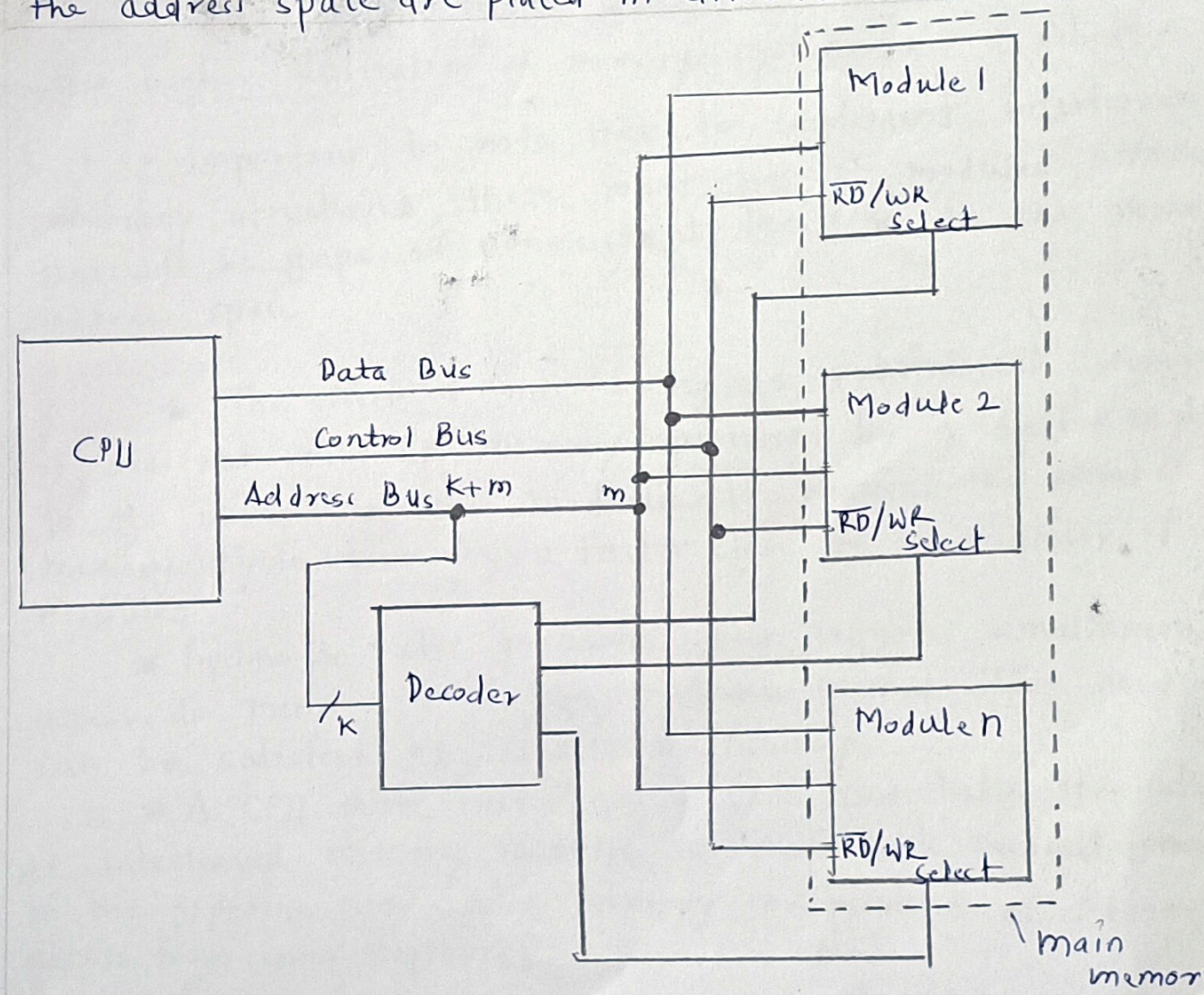
Interleaved Memory / Memory Interleaving

* We know that performance of memory is measured in terms of Access Time. Less is the access time faster is the memory.

* But more access time (slower memories) is one of the biggest problem in improving overall system performance

* One way to reduce the access time is to use cache memory

An alternative technique to reduce memory access time is memory interleaving. In this technique, the main memory is divided into a number of memory modules and the addresses are arranged such that the successive words in the address space are placed in different modules as shown



* Most of the time CPU accesses consecutive memory locations. In such situations address will be to different memory modules.

* Since these modules can be accessed in parallel, the average access time of fetching word from the main memory can be reduced.

* The lower order 'k' bit of memory address are generally used to select a module and the higher order 'm' bits are used to access a perticular location within the selected module.

* In this way consecutive addresses are located in successive modules. Thus any component of the system that generates requests for access to consecutive memory locations can keep several modules busy at any one time.

* This results in both faster access to a block of data and higher utilization of memory system as a whole.

* It is important to note that to implement interleaved memory structured, there must be 2^k modules, otherwise there will be gaps of nonexistant locations in the memory address space

* The effect of interleaving is substantial. However it does not speed up memory operation by a factor equal to the number of the module. It reduces the effect memory cycle time by a factor close to the number of modules.

* Pipeline & vector processors often require simultaneous access to memory from two or more sources. This need can be satisfied by interleaved memory.

* A CPU with inst. pipeline can also take the advantage of interleaved memory modules so that each segment (stage) in the pipeline can access memory, independent of memory access from other segments.