Darbhanga College of Engineering

Computer Science & Engineering

Subject:- Computer Organization &Architecture
4th Semester CSE (Question Paper)

Module 1

1. What is Computer Architecture ?
Ans: Computer architecture is a specification describing how hardware and software technologies interact to create a computer platform or system. When we think of the word architecture, we think of building a house or a building. Keeping that same principle in mind, computer architecture involves building a computer and all that goes into a computer system. Computer architecture consists of three main categories.

- System design – This includes all the hardware parts, such as CPU, data processors, multiprocessors, memory controllers and direct memory access. This part is the actual computer system.
- Instruction set architecture – The includes the CPU's functions and capabilities, the CPU's programming language, data formats, processor register types and instructions used by computer programmers. This part is the software that makes it run, such as Windows or Photoshop or similar programs.
- Microarchitecture – This defines the data processing and storage element or data paths and how they should be implemented into the instruction set architecture. These might include DVD storage devices or similar devices.

2. What are the three categories of Computer Architecture ?
Ans: . Computer architecture consists of three main categories.

- System design – This includes all the hardware parts, such as CPU, data processors, multiprocessors, memory controllers and direct memory access. This part is the actual computer system.
- Instruction set architecture – The includes the CPU's functions and capabilities, the CPU's programming language, data formats, processor register types and instructions used by computer programmers. This part is the software that makes it run, such as Windows or Photoshop or similar programs.
- Microarchitecture – This defines the data processing and storage element or data paths and how they should be implemented into the instruction set architecture. These might include DVD storage devices or similar devices.

3. What are the common Components of a Microprocessor?
Ans: **Some of the common components of a microprocessor are:**

- **Control Unit**.
- **I/O** Units.
- **Arithmetic Logic Unit** (ALU)
- Registers.
- Cache.

4. How Computer Architecture different from a Computer Organization?
Ans: **Computer Architecture**

Computer Architecture is a blueprint for design and implementation of a computer system. It provides the functional details and behaviour of a computer system and comes before computer organization. Computer architecture deals with 'What to do?'

Computer Organization

Computer Organization is how operational parts of a computer system are linked together. It implements the provided computer architecture. Computer organization deals with 'How to do?'

Following are some of the important differences between Computer Architecture and Computer Organization.

Sr. No.	Key	Computer Architecture	Computer Organization
1	Purpose	Computer architecture explains what a computer should do.	Computer organization explains how a computer works.
2	Target	Computer architecture provides functional behavior of computer system.	Computer organization provides structural relationships between parts of computer system.
3	Design	Computer architecture deals with high level design.	Computer organization deals with low level design.
4	Actors	Actors in Computer architecture are hardware parts.	Actor in computer organizaton is performance.
5	Order	Computer architecture is designed first.	Computer organization is started after finalizing computer architecture.

5. What are the various Interrupts in a Microprocessor system?

   Ans: Interrupts are the signals generated by the external devices to request the microprocessor to perform a task. There are 5 interrupt signals, i.e. TRAP, RST 7.5, RST 6.5, RST 5.5, and INTR.

   Interrupt are classified into following groups based on their parameter −

- **Vector interrupt** − In this type of interrupt, the interrupt address is known to the processor. **For example:** RST7.5, RST6.5, RST5.5, TRAP.

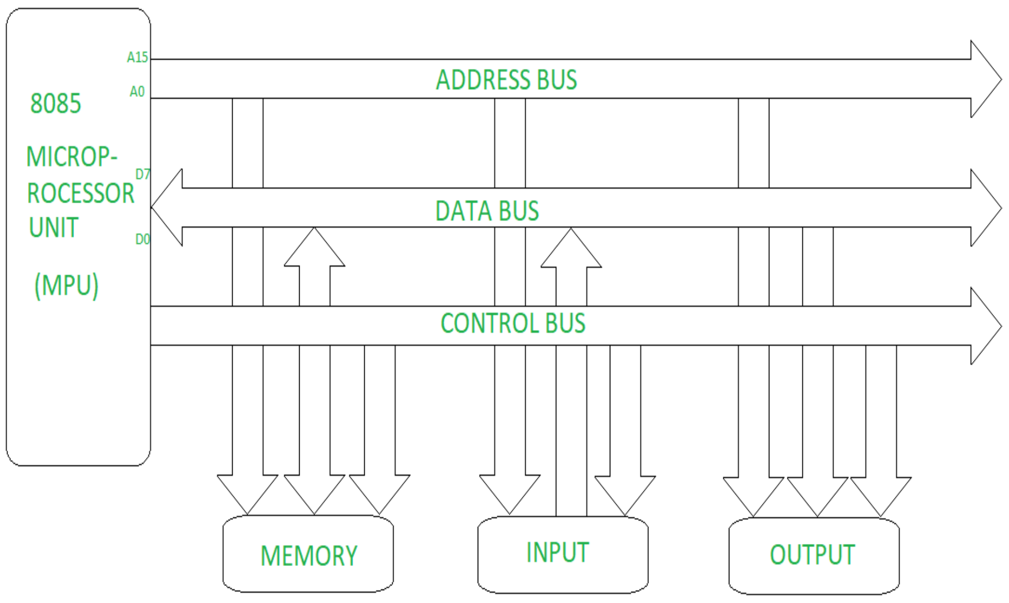
- **Non-Vector interrupt** − In this type of interrupt, the interrupt address is not known to the processor so, the interrupt address needs to be sent externally by the device to perform interrupts. **For example:** INTR.
- **Maskable interrupt** − In this type of interrupt, we can disable the interrupt by writing some instructions into the program. **For example:** RST7.5, RST6.5, RST5.5.
- **Non-Maskable interrupt** − In this type of interrupt, we cannot disable the interrupt by writing some instructions into the program. **For example:** TRAP.
- **Software interrupt** − In this type of interrupt, the programmer has to add the instructions into the program to execute the interrupt. There are 8 software interrupts in 8085, i.e. RST0, RST1, RST2, RST3, RST4, RST5, RST6, and RST7.
- **Hardware interrupt** − There are 5 interrupt pins in 8085 used as hardware interrupts, i.e. TRAP, RST7.5, RST6.5, RST5.5, INTA.

  Note − NTA is not an interrupt, it is used by the microprocessor for sending acknowledgement. TRAP has the highest priority, then RST7.5 and so on.

6. List out the types of bus.

   Ans: Bus is a group of conducting wires which carries information, all the peripherals are connected to microprocessor through Bus.

   Diagram to represent bus organization system of 8085 Microprocessor.



Bus organization system of 8085 Microprocessor

There are three types of buses.

1. **Address** **bus** –

   It is a group of conducting wires which carries address only.Address bus is unidirectional because data flow in one direction, from microprocessor to memory or from microprocessor to Input/output devices (That is, Out of Microprocessor).

   Length of Address Bus of 8085 microprocessor is 16 Bit (That is, Four Hexadecimal Digits), ranging from 0000 H to FFFF H, (H denotes Hexadecimal). The microprocessor 8085 can transfer maximum 16 bit address which means it can address 65, 536 different memory location.

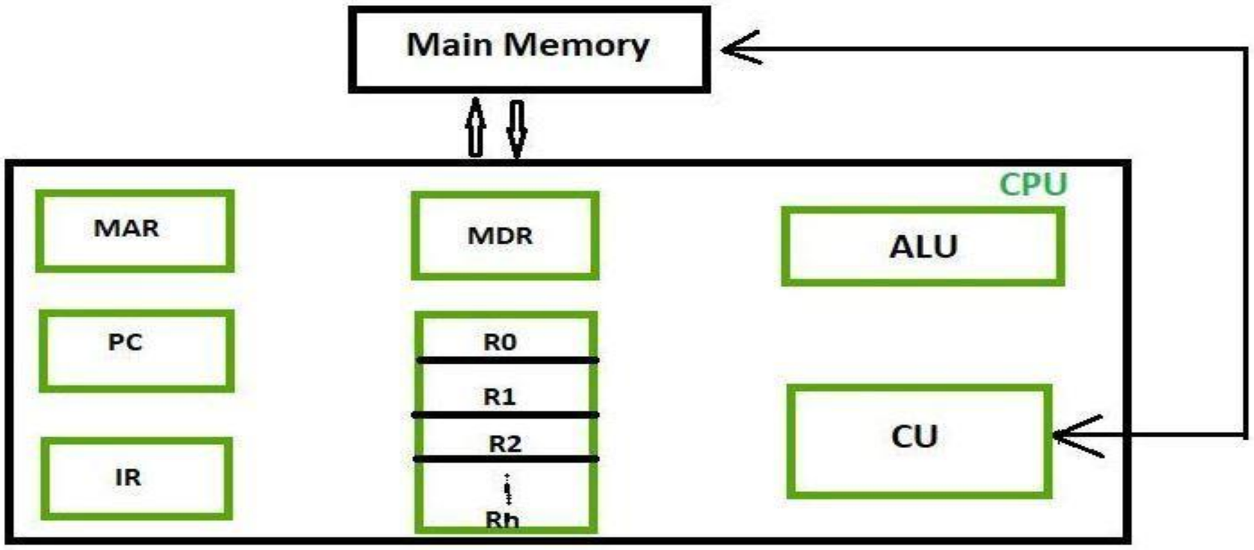
   The Length of the address bus determines the amount of memory a system can address.Such as a system with a 32-bit address bus can address 2^32 memory locations.If each memory location holds one byte, the addressable memory space is 4 GB.However, the actual amount of memory that can be accessed is usually much less than this theoretical limit due to chipset and motherboard limitations.

2. **Data bus –**
   It is a group of conducting wires which carries Data only.Data bus is bidirectional because data flow in both directions, from microprocessor to memory or Input/Output devices and from memory or Input/Output devices to microprocessor.
   Length of Data Bus of 8085 microprocessor is 8 Bit (That is, two Hexadecimal Digits), ranging from 00 H to FF H. (H denotes Hexadecimal).
   When it is write operation, the processor will put the data (to be written) on the data bus, when it is read operation, the memory controller will get the data from specific memory block and put it into the data bus.
   The width of the data bus is directly related to the largest number that the bus can carry, such as an 8 bit bus can represent 2 to the power of 8 unique values, this equates to the number 0 to 255.A 16 bit bus can carry 0 to 65535.
3. **Control bus –**
   It is a group of conducting wires, which is used to generate timing and control signals to control all the associated peripherals, microprocessor uses control bus to process data, that is what to do with selected memory location. Some control signals are:
   - Memory read
   - Memory write
   - I/O read
   - I/O Write
   - Opcode fetch

   If one line of control bus may be the read/write line.If the wire is low (no electricity flowing) then the memory is read, if the wire is high (electricity is flowing) then the memory is written.

7. Define Program Counter.
   Ans: A **program counter** is a register in a computer processor that contains the address (location) of the instruction being executed at the current time.

8. Why we have the types of Registers?
   Ans: In Computer Architecture, the Registers are very fast computer memory which are used to execute programs and operations efficiently. This does by giving access to commonly used values, i.e., the values which are in the point of operation/execution at that time. So, for this purpose, there are several different classes of CPU registers which works in coordination with the computer memory to run operations efficiently.



These are classified as given below.

- **Accumulator:**
  This is the most frequently used register used to store data taken from memory. It is in different numbers in different microprocessors.
- **Memory Address Registers (MAR):**
-
  It holds the address of the location to be accessed from memory. MAR and MDR (Memory Data Register) together facilitate the communication of the CPU and the main memory.
- **Memory Data Registers (MDR):**
-
  It contains data to be written into or to be read out from the addressed location.
- **General Purpose Registers:**
-
  These are numbered as R0, R1, R2….Rn, and used to store temporary data during any ongoing operation. Its content can be accessed by assembly programming.
- **Program Counter (PC):**
-
  Program Counter (PC) is used to keep the track of execution of the program. It contains the memory address of the next instruction to be fetched. PC points to the address of the next instruction to be fetched from the main memory when the previous instruction has been successfully completed. Program Counter (PC) also functions to count the number of instructions.
- **Instruction Register (IR):**
-
  It is the register which holds the instruction which is currently been executed.
  So, these are the different registers which are operating for a specific purpose.

9. What is Number Representation?

Ans: When working with any kind of digital electronics in which numbers are being represented, it is important to understand the different ways numbers are represented in these systems. Almost without fail, numbers are represented by two voltage levels which can represent a one or a zero (an interesting exception to this rule is the new memory device recently announced by Intel which uses one of four possible voltage levels, thereby increasing the amount of information that can be stored in a given space). The number system based on ones and zeroes is called the binary system (because there are only two possible digits). Before discussing the binary system, a review of the decimal (ten possible digits) system is in order, because many of the concepts of the binary system will be easier to understand when introduced alongside their decimal counterpart.

You should all have some familiarity with the decimal system. For instance, to represent the positive integer one hundred and twenty-five as a decimal number, we can write (with the postivie sign implied). The subscript 10 denotes the number as a base 10 (decimal) number.

12510 = 1*100 + 2*10 + 5*1 = 1*102 + 2*101 + 5*100

10. What is 2’s complement method and give few examples?

Ans: **Two's complement** is a mathematical operation on binary numbers, and is an **example** of a radix **complement**. It is used in computing as a **method** of signed number representation. The **two's complement** of an N-bit number is defined as its **complement** with respect to $\mathbf{2}^N$.

For **example**, **2's complement** of “01000” is “11000” (Note that we first find one's **complement** of 01000 as 10111). If there are all 1's (in one's **complement**), we add an extra 1 in the string. For **example**, **2's complement** of “000” is “1000” (1's **complement** of “000” is “111”).

11. Define Floating point representation of numbers.

Ans: The term **floating point** refers to the fact that a **number's** radix **point** (decimal **point**, or, more commonly in computers, binary **point**) can "**float**"; that is, it can be placed anywhere relative to the significant digits of the **number**.

In **computers**, **floating-point numbers are represented** in scientific notation of fraction (F) and exponent (E) with a radix of 2, in the form of F×2^E . Both E and F can be positive as well as negative. Modern **computers** adopt IEEE 754 standard for **representing floating-point numbers**.

12. Difference between Signed magnitude and 2’s complement?

Ans:

SIGNED MAGNITUDE METHOD	**2’S COMPLEMENT METHOD**
It is a method to denote fixed point signed numbers.	It is also used to denote fixed point signed numbers.
Number is divided into two parts.	Number is considered as a whole.
Sign bit is considered explicitly.	Sign bit is not considered explicitly.
Additional hardware is required for	No additional hardware is

SIGNED MAGNITUDE METHOD	2’S COMPLEMENT METHOD
resultant sign of arithmetic.	required in 2’s complement method.
Addition and subtraction are performed on separate hardware.	Addition and subtraction are performed by using adder only.
It has two different representation for 0. One is +0 and second is -0. (+0 : 0000 0000) & (-0 : 1000 0000)	0 has only one representation for -0 and +0 (+0 or -0 : 0000 0000).
It is non-weighted system.	It assigns negative weight to the sign bit.

13. How the negative numbers are stored in memory?

Ans: Suppose the following fragment of code, int a = -34; Now how will this be stored in memory. So here is the complete theory. Whenever a number with minus sign is encountered, the number (ignoring minus sign) is converted to its binary equivalent. Then the two’s complement of the number is calculated. That two’s complement is kept at place allocated in memory and the sign bit will be set to 1 because the binary being kept is of a negative number. Whenever it comes on accessing that value firstly the sign bit will be checked if the sign bit is 1 then the binary will be two’s complemented and converted to equivalent decimal number and will be represented with a minus sign.

Let us take an example:
Example –
int a = -2056;
Binary of 2056 will be calculated which is:

00000000000000000000100000001000 (32 bit representation, according of storage of int in C)
2’s complement of the above binary is:
11111111111111111111011111111000.

So finally the above binary will be stored at memory allocated for variable a.
When it comes on accessing the value of variable a, the above binary will be retrieved from the memory location, then its sign bit that is the left most bit will be checked as it is 1 so the binary number is of a negative number so it will be 2’s complemented and when it will be 2’s complemented will be get the binary of 2056 which is:

00000000000000000000100000001000

The above binary number will be converted to its decimal equivalent which is 2056 and as the sign bit was 1 so the decimal number which is being gained from the binary number will be represented with a minus sign. In our case -2056.

14. Why are negative numbers stored as 2's complement?

Ans: When doing addition/subtraction on binary numbers in other representations we need to apply different logics (circuits) to perform addition and subtraction. In 2s-complement representation, we represent a positive number as it is and negative number by its corresponding 2s-complement, so we can use the same circuit to perform addition and subtraction.

For example: to add 6+3 using 5 bit 2s-complement representation,

```
00110
+
00011
———
01001
```

To subtract 6-3, rewrite as 6+ (-3):

```
00110
+
11101 (2s-complement of 3)
——-
00011
```

15. What is Booth's Algorithm?

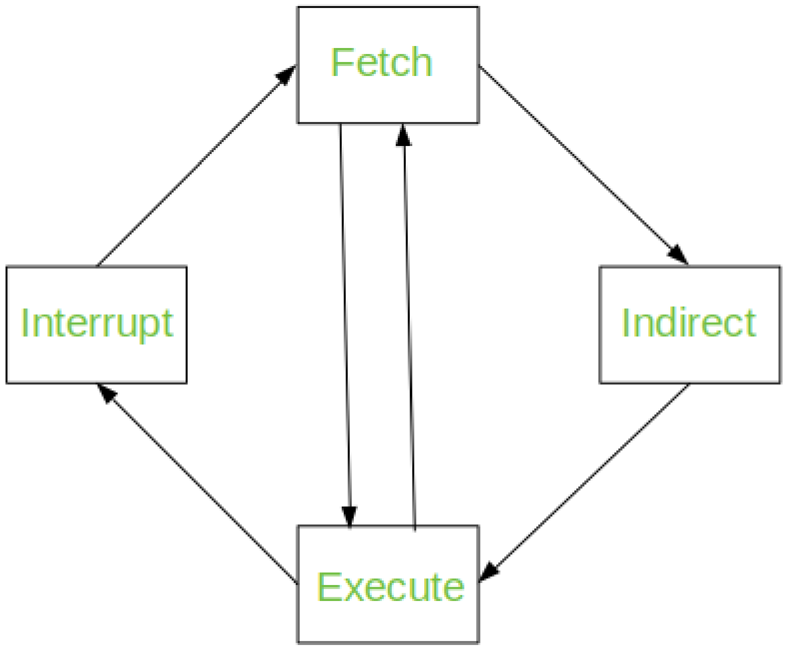
Ans: Booth algorithm gives a procedure for multiplying binary integers in signed 2's complement representation in efficient way, i.e., less number of additions/subtractions required. It operates on the fact that strings of 0's in the multiplier require no addition but just shifting and a string of 1's in the multiplier from bit weight 2^k to weight 2^m can be treated as 2^(k+1) to 2^m.

As in all multiplication schemes, booth algorithm requires examination of the multiplier bits and shifting of the partial product. Prior to the shifting, the multiplicand may be added to the partial product, subtracted from the partial product, or left unchanged according to following rules:

1. The multiplicand is subtracted from the partial product upon encountering the first least significant 1 in a string of 1's in the multiplier
2. The multiplicand is added to the partial product upon encountering the first 0 (provided that there was a previous '1') in a string of 0's in the multiplier.
3. The partial product does not change when the multiplier bit is identical to the previous multiplier bit.

16. What are the different Instruction Cycles?

Ans: **The Instruction Cycle –**
Each phase of Instruction Cycle can be decomposed into a sequence of elementary micro-operations. In the above examples, there is one sequence each for the *Fetch, Indirect, Execute and Interrupt Cycles*.



The Instruction Cycle

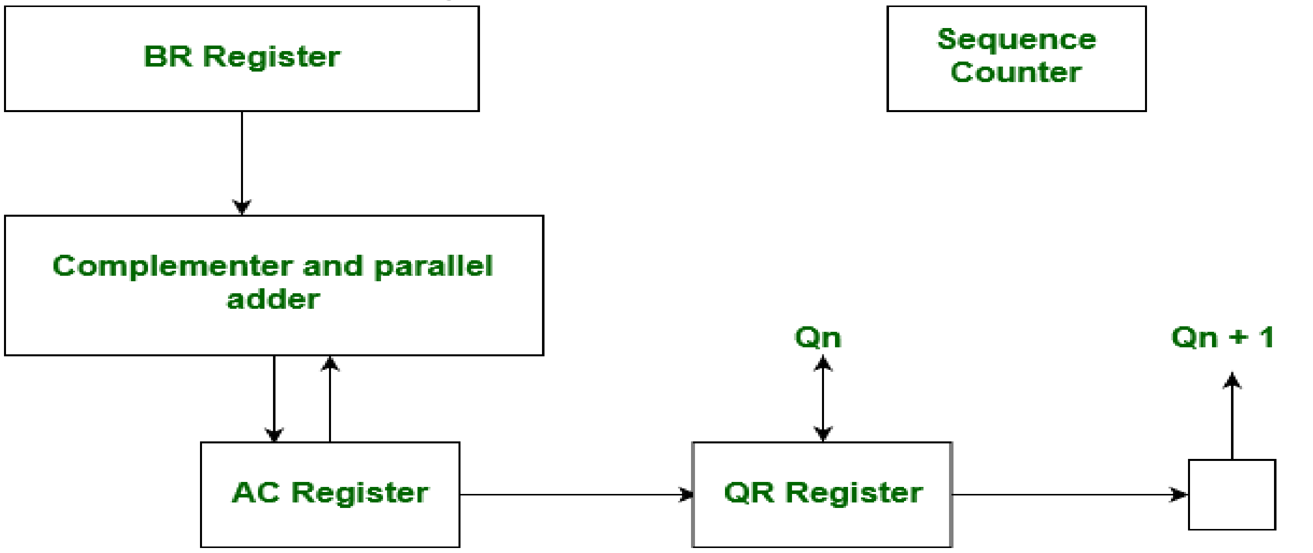
The *Indirect Cycle* is always followed by the *Execute Cycle*. The *Interrupt Cycle* is always followed by the *Fetch Cycle*. For both fetch and execute cycles, the next cycle depends on the state of the system.

17. What is Booth's Algorithm?
Ans: Booth algorithm gives a procedure for **multiplying binary integers** in signed 2's complement representation **in efficient way**, i.e., less number of additions/subtractions required. It operates on the fact that strings of 0's in the multiplier require no addition but just shifting and a string of 1's in the multiplier from bit weight 2^k to weight 2^m can be treated as 2^(k+1) to 2^m.
As in all multiplication schemes, booth algorithm requires examination **of the multiplier bits** and shifting of the partial product. Prior to the shifting, the multiplicand may be added to the partial product, subtracted from the partial product, or left unchanged according to following rules:

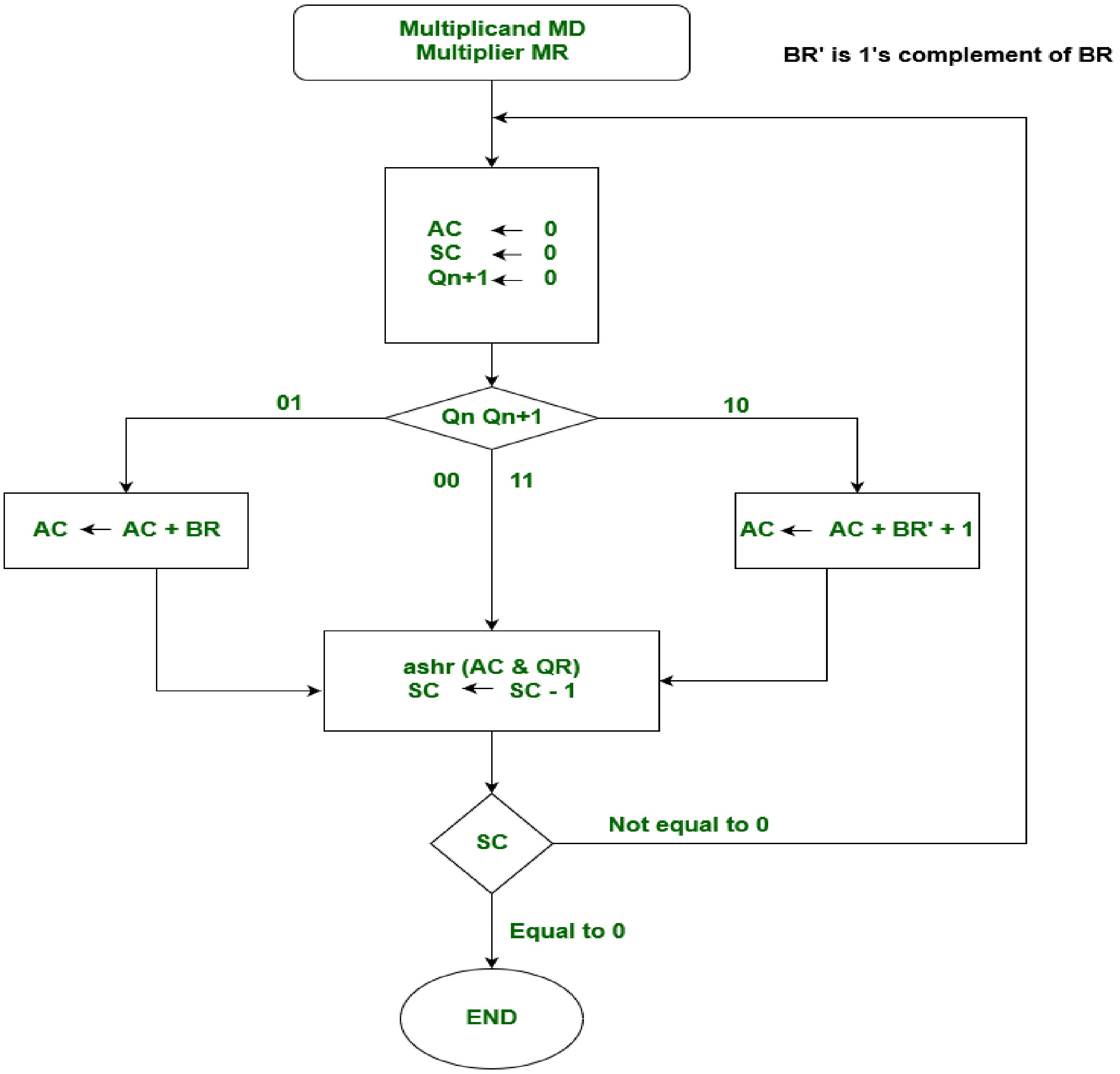
1. The multiplicand is subtracted from the partial product upon encountering the first least significant 1 in a string of 1's in the multiplier
2. The multiplicand is added to the partial product upon encountering the first 0 (provided that there was a previous '1') in a string of 0's in the multiplier.
3. The partial product does not change when the multiplier bit is identical to the previous multiplier bit.

Hardware Implementation of Booths Algorithm – The hardware implementation of the booth algorithm requires the register configuration shown in the figure below.

Booth's Algorithm Flowchart –



We name the register as A, B and Q, AC, BR and QR respectively. Qn designates the least significant bit of multiplier in the register QR. An extra flip-flop Qn+1is appended to QR to facilitate a double inspection of the multiplier. The flowchart for the booth algorithm is shown below.



AC and the appended bit Qn+1 are initially cleared to 0 and the sequence SC is set to a number n equal to the number of bits in the multiplier. The two bits of the multiplier in Qn and Qn+1are inspected. If the two bits are equal to 10, it means that the first 1 in a string has been encountered.

This requires subtraction of the multiplicand from the partial product in AC. If the 2 bits are equal to 01, it means that the first 0 in a string of 0's has been encountered. This requires the addition of the multiplicand to the partial product in AC.

When the two bits are equal, the partial product does not change. An overflow cannot occur because the addition and subtraction of the multiplicand follow each other. As a consequence, the 2 numbers that are added always have a opposite signs, a condition that excludes an overflow. The next step is to shift right the partial product and the multiplier (including Qn+1). This is an arithmetic shift right (ashr) operation which AC and QR ti the right and leaves the sign bit in AC unchanged. The sequence counter is decremented and the computational loop is repeated n times.

Example – A numerical example of booth's algorithm is shown below for n = 4. It shows the step by step multiplication of -5 and -7.

MD = -5 = 1011, MD = 1011, MD'+1 = 0101

MR = -7 = 1001

The explanation of first step is as follows: Qn+1

AC = 0000, MR = 1001, Qn+1 = 0, SC = 4

Qn Qn+1 = 10

So, we do AC + (MD)'+1, which gives AC = 0101

On right shifting AC and MR, we get

AC = 0010, MR = 1100 and Qn+1 = 1

OPERATION	AC	MR	QN+1	SC
	0000	1001	0	4
AC + MD' + 1	0101	1001	0	
ASHR	0010	1100	1	3
AC + MR	1101	1100	1	
ASHR	1110	1110	0	2
ASHR	1111	0111	0	1
AC + MD' + 1	0010	0011	1	0

Product is calculated as follows:

Product = AC MR

Product = 0010 0011 = 35

18. Define Micro-Operation.

Ans: **Micro-operation**. In computer central processing units, **micro-operations** (also known as a **micro**-ops or μops) are detailed low-level instructions used in some designs to implement complex machine instructions (sometimes termed macro-instructions in this context).

19. Define Instruction Set Architecture.

Ans: The instruction set, also called ISA (instruction set architecture), is part of a computer that pertains to programming, which is basically machine language. The instruction set provides commands to the processor, to tell it what it needs to do.

20. Define instruction.

Ans: Three Categories of Instructions: The instruction set is a collection of instructions each representing a CPU operation. In assembly language, the instructions are represented by the mnimonics and the operands (or their addresses) involved in the operation Data manipulation.

21. What is the types of instructions?

Ans: Three Categories of Instructions:

The instruction set is a collection of instructions each representing a CPU operation. In assembly language, the instructions are represented by the *mnimonics* and the operands (or their addresses) involved in the operation.

1. Data manipulation
   - Arithmetic manipulation:

     add, sub, mult, div, etc.

   - Logic and bit manipulation:

     and, or, nor, xor, etc.

   - Shift and rotation (to right or left):

     sll, srl, sra, rol, ror, etc.

     Note: Instruction srl (shift right logical) shifts a 0 into the vacated bit (the sign bit), while instruction sra (shift right arithmetic) repeats the sign bit (sign extension for signed 2's complement).

2. Data transfer
   - transfer data between MM and CPU:

     lw (load word), la (load address), lb (load byte),

     sw (store word), sb (store byte), etc.

   - transfer data between registers in RF:

     move, mfhi, mflo, mthi, mtlo,

3. Program control

- branch to instruction other than the one following the current one conditionally or unconditionally (based on comparison between two operands or between one operand and zero):

  b, beq, bne, bgt, blt, bge, ble, beqz, bnez, bgez, bgtz,

- jump to different segments of the program (functions, subroutines, etc.)

  j, jal, jalr, jr, etc.

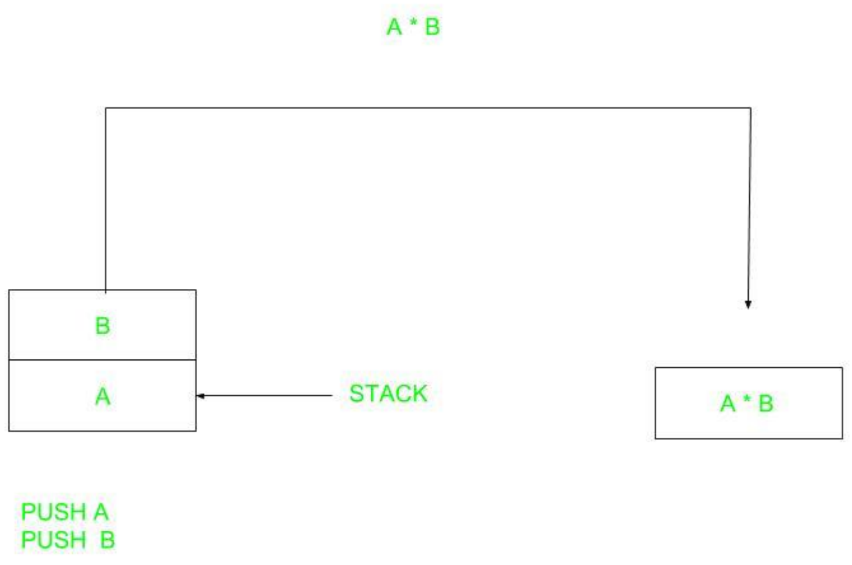
22. Who define the maximum length of each type of instruction?

Ans: n instruction format defines the different component of an instruction. The main components of an instruction are opcode (which instruction to be executed) and operands (data on which instruction to be executed). Here are the different terms related to instruction format:

1. Instruction set size – It tells the total number of instructions defined in the processor.
2. Opcode size – It is the number of bits occupied by the opcode which is calculated by taking log of instruction set size.
3. Operand size – It is the number of bits occupied by the operand.
4. Instruction size – It is calculated as sum of bits occupied by opcode and operands.

23. What are Instruction Formats (Zero, One, Two and Three Address Instruction)?

1. Ans: **Zero Address Instructions –**



A stack based computer do not use address field in instruction.To evaluate a expression first it is converted to revere Polish Notation i.e. Post fix Notation.

Expression: X = (A+B)*(C+D)

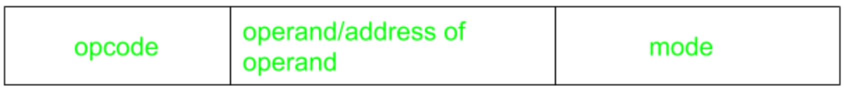
Postfixed : X = AB+CD+*

TOP means top of stack

M[X] is any memory location

PUSH	A	TOP = A
PUSH	B	TOP = B
ADD		TOP = A+B
PUSH	C	TOP = C
PUSH	D	TOP = D
ADD		TOP = C+D
MUL		TOP = (C+D)*(A+B)
POP	X	M[X] = TOP

2. **One Address Instructions –**
This use a implied ACCUMULATOR register for data manipulation.One operand is in accumulator and other is in register or memory location.Implied means that the CPU already know that one operand is in accumulator so there is no need to specify it.



Expression: X = (A+B)*(C+D)

AC is accumulator

M[] is any memory location

M[T] is temporary location

LOAD	A	AC = M[A]
ADD	B	AC = AC + M[B]
STORE	T	M[T] = AC
LOAD	C	AC = M[C]
ADD	D	AC = AC + M[D]
MUL	T	AC = AC * M[T]

STORE	X	M[X] = AC

3. **Two Address Instructions –**
   This is common in commercial computers.Here two address can be specified in the instruction.Unlike earlier in one address instruction the result was stored in accumulator here result cab be stored at different location rather than just accumulator, but require more number of bit to represent address.

opcode	Destination address	Source address	mode

   Here destination address can also contain operand.

   Expression: X = (A+B)*(C+D)

   R1, R2 are registers

   M[] is any memory location

MOV	R1, A	R1 = M[A]
ADD	R1, B	R1 = R1 + M[B]
MOV	R2, C	R2 = C
ADD	R2, D	R2 = R2 + D
MUL	R1, R2	R1 = R1 * R2
MOV	X, R1	M[X] = R1

4. **Three Address Instructions –**
   This has three address field to specify a register or a memory location. Program created are much short in size but number of bits per instruction increase. These instructions make creation of program much easier but it does not mean that program will run much faster because now instruction only contain more information but each micro operation (changing content of register, loading address in address bus etc.) will be performed in one cycle only.

opcode	Destination address	Source address	Source address	mode

   Expression: X = (A+B)*(C+D)

   R1, R2 are registers

   M[] is any memory location

ADD	R1, A, B	R1 = M[A] + M[B]

ADD	R2, C, D	R2 = M[C] + M[D]
MUL	X, R1, R2	M[X] = R1 * R2

24. Difference between 3-address instruction and 0-address instruction.

Ans: **Difference between Three-Address Instruction and Zero-Address Instruction :**

THREE-ADDRESS INSTRUCTION	**ZERO-ADDRESS INSTRUCTION**
It has four fields.	It has only one field.
It has one field for opcode and three fields for address.	It has one field for opcode and no fields for address.
It has long instruction length.	It has shorter instruction.
It is slower accessing location inside processor than memory.	It is faster accessing location inside processor than memory.
There is distinct address fields for destination and source.	There is no address field common for destination and source.
In 3-address format, destination address can not contain operand.	While in 0-address format, there is no field for operand.
In 3-address format, number of instructions are less.	While in 0-address format, number of instructions are more.
It may need three memory accesses for one instruction.	It does not need three memory accesses.

25. Difference between 3-address instruction and 1-address instruction.

Ans:

THREE-ADDRESS INSTRUCTION	ONE-ADDRESS INSTRUCTION
It has four fields.	It has only two fields.
It has one field for opcode and three fields for address.	It also has one field for opcode but there is only one field for address.
It has long instruction length.	It has shorter instruction.
There may be three memory accesses needed for an instruction.	There is a single memory access needed for an instruction.
It is slower accessing location inside processor than memory.	It is faster accessing location inside processor than memory.
It disadvantage i.e. three memory access is eliminated by two-address memory.	It eliminated two memory access.
There are three location for operand and result.	There is only one location for operand and result.

26. What is the difference between Hardwired and Micro-programmed Control Unit ?

Ans: **Difference between Hardwired and Microprogrammed Control Unit:**

ATTRIBUTES	HARDWIRED CONTROL UNIT	MICROPROGRAMMED CONTROL UNIT
1. Speed	Speed is fast	Speed is slow
2. Cost of Imlementation	More costlier.	Cheaper.
3. Flexibility	Not flexible to accommodate new system specification or new instruction redesign is required.	More flexible to accommodate new system specification or new instruction sets.
4. Ability to Handle Complex	Difficult to handle complex intruction sets.	Easier to handle complex intruction sets.

ATTRIBUTES	HARDWIRED CONTROL UNIT	MICROPROGRAMMED CONTROL UNIT
Instructions		
5. Decoding	Complex decoding and sequencing logic.	Easier decoding and sequencing logic.
6. Applications	RISC Microprocessor	CISC Microprocessor
7. Instruction set of Size	Small	Large
8. Control Memory	Absent	Present
9. Chip Area Required	Less	More
10. Occurrence	Occurrence of error is more	Occurrence of error is less

27. Write the steps are followed in addition and substraction of two floating point numbers.

Ans:
To understand floating point addition, first we see addition of real numbers in decimal as same logic is applied in both cases.

For example, we have to add **1.1 * 10$_3$** and **50.**

We cannot add these numbers directly. First, we need to align the exponent and then, we can add significand.
After aligning exponent, we get **50 = 0.05 * 10$_3$**
Now adding significand, **0.05 + 1.1 = 1.15**
So, finally we get **(1.1 * 10$_3$ + 50) = 1.15 * 10$_3$**
Here, notice that we shifted **50** and made it **0.05** to add these numbers.

Now let us take example of floating point number addition
We follow these steps to add two numbers:

1. Align the significand
2. Add the significands

3. Normalize the result

Let the two numbers be
x = 9.75
y = 0.5625

Converting them into 32-bit floating point representation,
9.75’s representation in 32-bit format = **0 10000010 00111000000000000000000**
0.5625’s representation in 32-bit format = **0 01111110 00100000000000000000000**

Now we get the difference of exponents to know how much shifting is required.
(**10000010 – 01111110**)$_2$ = (**4**)$_{10}$

Now, we shift the mantissa of lesser number right side by 4 units.
Mantissa of **0.5625 = 1.00100000000000000000000**
(note that 1 before decimal point is understood in 32-bit representation)
Shifting right by **4** units, we get **0.00010010000000000000000**
Mantissa of **9.75** = **1. 00111000000000000000000**

Adding mantissa of both
0. 00010010000000000000000
+ 1. 00111000000000000000000
———————————————————-
1. 01001010000000000000000
In final answer, we take exponent of bigger number
So, final answer consist of :
Sign bit = **0**
Exponent of bigger number = **10000010**
Mantissa = **01001010000000000000000**

32 bit representation of answer = **x + y** = **0 10000010 01001010000000000000000**

- **FLOATING POINT SUBTRACTION**

Subtraction is similar to addition with some differences like we subtract mantissa unlike addition and in sign bit we put the sign of greater number.

Let the two numbers be
x = 9.75
y = – 0.5625

Converting them into 32-bit floating point representation
9.75’s representation in 32-bit format = **0 10000010 00111000000000000000000**
– 0.5625’s representation in 32-bit format = **1 01111110 00100000000000000000000**

Now, we find the difference of exponents to know how much shifting is required.

(**10000010 – 01111110**)$_2$ = (**4**)$_{10}$
Now, we shift the mantissa of lesser number right side by 4 units.
Mantissa of **– 0.5625 = 1.00100000000000000000000**
(note that 1 before decimal point is understood in 32-bit representation)
Shifting right by **4** units, **0.00010010000000000000000**
Mantissa of **9.75**= **1. 00111000000000000000000**

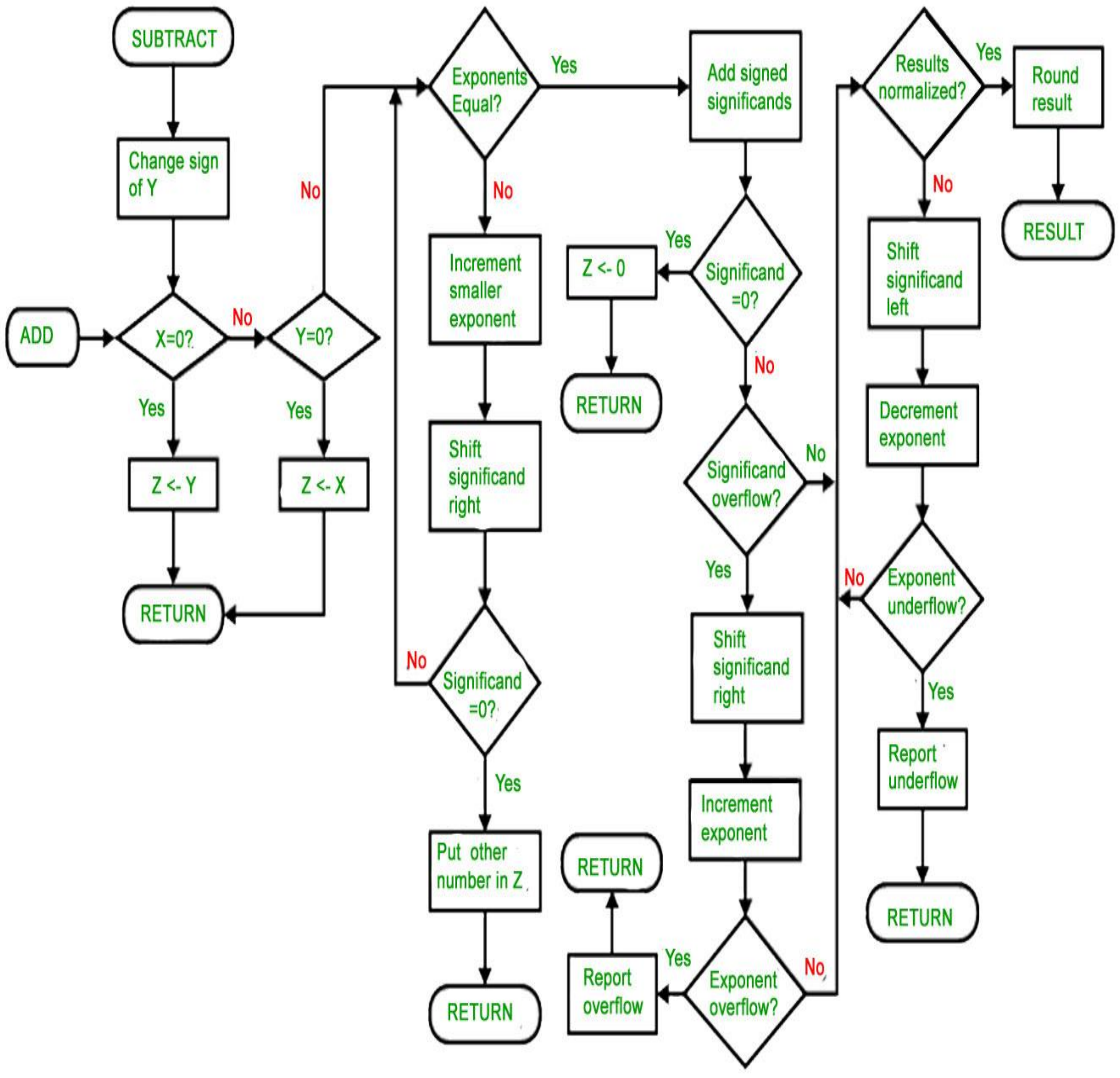
Subtracting mantissa of both
0. 00010010000000000000000

– 1. 00111000000000000000000

---

1. 00100110000000000000000

Sign bit of bigger number = **0**
So, finally the answer = **x – y = 0 10000010 00100110000000000000000**



Floating-Point Addition and Subtraction (Z ← X ± Y)

28. Discuss different addressing modes in brief.

Ans: **Addressing Modes**– The term addressing modes refers to the way in which the operand of an instruction is specified. The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually executed.

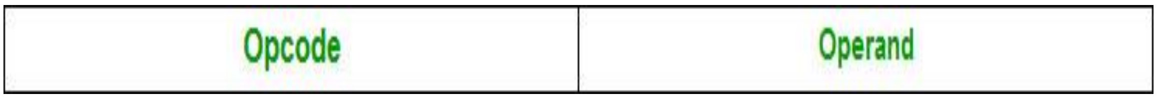
Addressing modes for 8086 instructions are divided into two categories:

1) Addressing modes for data

2) Addressing modes for branch

The 8086 memory addressing modes provide flexible access to memory, allowing you to easily access variables, arrays, records, pointers, and other complex data types. The key to good assembly language programming is the proper use of memory addressing modes.

An assembly language program instruction consists of two parts



The memory address of an operand consists of two components:

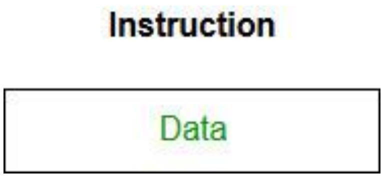
IMPORTANT TERMS

- **Starting address** of memory segment.
- **Effective address or Offset**: An offset is determined by adding any combination of three address elements: **displacement, base and index.**
  - **Displacement:** It is an 8 bit or 16 bit immediate value given in the instruction.
  - **Base**: Contents of base register, BX or BP.
  - **Index**: Content of index register SI or DI.

According to different ways of specifying an operand by 8086 microprocessor, different addressing modes are used by 8086.

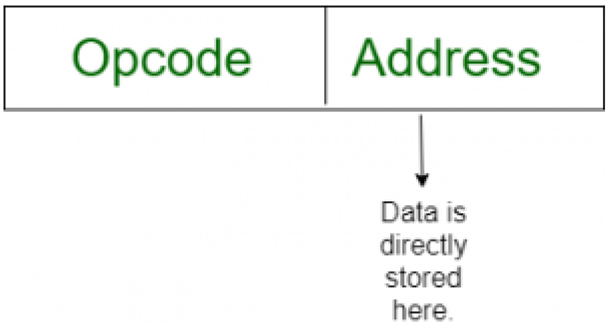
Addressing modes used by 8086 microprocessor are discussed below:

- **Implied mode:**: In implied addressing the operand is specified in the instruction itself. In this mode the data is 8 bits or 16 bits long and data is the part of instruction.Zero address instruction are designed with implied addressing mode.



Example: CLC (used to reset Carry flag to 0)

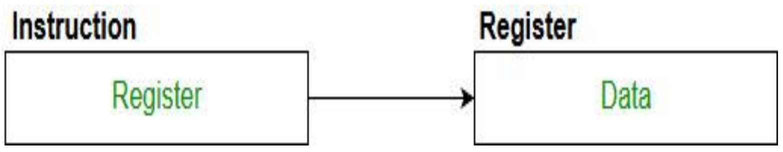
- **Immediate addressing mode (symbol #):**In this mode data is present in address field of instruction .Designed like one address instruction format. **Note:**Limitation in the immediate mode is that the range of constants are restricted by size of address field.



Example: MOV AL, 35H (move the data 35H into AL register)

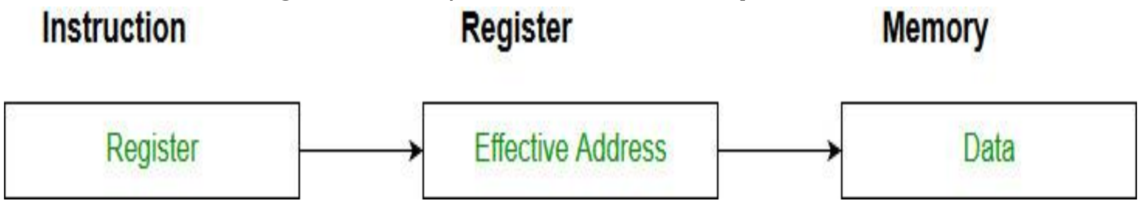
- **Register mode:** In register addressing the operand is placed in one of 8 bit or 16 bit general purpose registers. The data is in the register that is specified by the instruction.

Here one register reference is required to access the data.



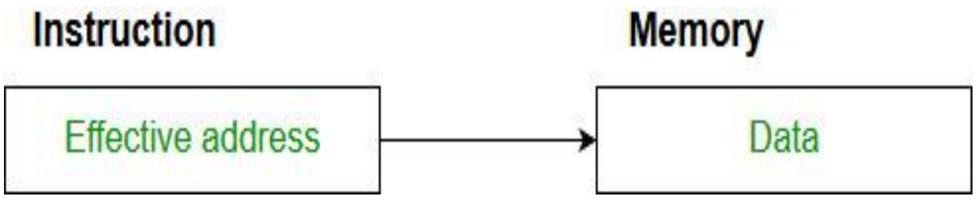
Example: MOV AX,CX (move the contents of CX register to AX register)

- **Register Indirect mode**: In this addressing the operand's offset is placed in any one of the registers BX,BP,SI,DI as specified in the instruction. The effective address of the data is in the base register or an index register that is specified by the instruction.
Here two register reference is required to access the data.



The 8086 CPUs let you access memory indirectly through a register using the register indirect addressing modes.

- MOV AX, [BX](move the contents of memory location s
addressed by the register BX to the register AX)
- **Auto Indexed (increment mode)**: Effective address of the operand is the contents of a register specified in the instruction. After accessing the operand, the contents of this register are automatically incremented to point to the next consecutive memory location.**(R1)+**.
Here one register reference,one memory reference and one ALU operation is required to access the data.
Example:
- Add R1, (R2)+ // OR
- R1 = R1 +M[R2]
R2 = R2 + d
Useful for stepping through arrays in a loop. R2 – start of array d – size of an element
- **Auto indexed (decrement mode)**: Effective address of the operand is the contents of a register specified in the instruction. Before accessing the operand, the contents of this register are automatically decremented to point to the previous consecutive memory location. –**(R1)**
Here one register reference,one memory reference and one ALU operation is required to access the data.
Example:
Add R1,-(R2) //OR
R2 = R2-*d*
R1 = R1 + M[R2]
Auto decrement mode is same as auto increment mode. Both can also be used to implement a stack as push and pop . Auto increment and Auto decrement modes are useful for implementing "Last-In-First-Out" data structures.
- **Direct addressing/ Absolute addressing Mode (symbol [ ]):** The operand's offset is given in the instruction as an 8 bit or 16 bit displacement element. In this addressing mode the 16 bit effective address of the data is the part of the instruction.
Here only one memory reference operation is required to access the data.



Example:ADD AL,[0301]   //add the contents of offset address 0301 to AL

- **Indirect addressing Mode (symbol @ or ()):**In this mode address field of instruction contains the address of effective address.Here two references are required.
  1st reference to get effective address.
  2nd reference to access the data.
  Based on the availability of Effective address, Indirect mode is of two kind:
  1. Register Indirect:In this mode effective address is in the register, and corresponding register name will be maintained in the address field of an instruction.
     Here one register reference,one memory reference is required to access the data.
  2. Memory Indirect:In this mode effective address is in the memory, and corresponding memory address will be maintained in the address field of an instruction.
     Here two memory reference is required to access the data.
- **Indexed addressing mode**: The operand's offset is the sum of the content of an index register SI or DI and an 8 bit or 16 bit displacement.
  Example:MOV AX, [SI +05]
- **Based Indexed Addressing:** The operand's offset is sum of the content of a base register BX or BP and an index register SI or DI.
  Example: ADD AX, [BX+SI]
  Based on Transfer of control, addressing modes are:
  - **PC relative addressing mode:** PC relative addressing mode is used to implement intra segment transfer of control, In this mode effective address is obtained by adding displacement to PC.
  - EA= PC + Address field value

  PC= PC + Relative value.
  - **Base register addressing mode:**Base register addressing mode is used to implement inter segment transfer of control.In this mode effective address is obtained by adding base register value to address field value.
  - EA= Base register + Address field value.
  - PC= Base register + Relative value.

  Note:
  1. PC relative nad based register both addressing modes are suitable for program relocation at runtime.
  2. Based register addressing mode is best suitable to write position independent codes.

29. What is the concept behind use of Booth Multiplier?

Ans: Booth's algorithm is a multiplication algorithm that multiplies two signed binary numbers in 2's compliment notation.
Booth used desk calculators that were faster at shifting than adding and created the algorithm to increase their speed. Booth's algorithm is of interest in the study of computer architecture. Here's the implememtation of the algorithm.

Examples:
```
Input : 0110, 0010
```

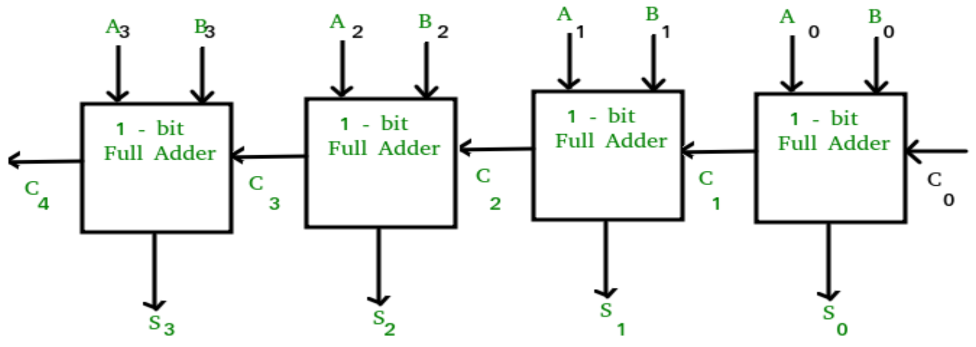
```
Output :  qn      q[n+1]                AC      QR     sc(step count)
                  initial         0000   0010        4
       0       0     rightShift      0000   0001        3
       1       0     A = A - BR      1010
                  rightShift      1101   0000        2
       0       1     A = A + BR      0011
                  rightShift      0001   1000        1
       0       0     rightShift      0000   1100        0


Result=1100
```

Algorithm :

Put multiplicand in BR and multiplier in QR
and then the algorithm works as per the following conditions :
1. If Qn and Qn+1 are same i.e. 00 or 11 perform arithematic shift by 1 bit.
2. If Qn Qn+1 = 10 do A= A + BR and perform arithematic shift by 1 bit.
3. If Qn Qn+1 = 01 do A= A – BR and perform arithematic shift by 1 bit.

30. How does the carry look-ahead adder work?

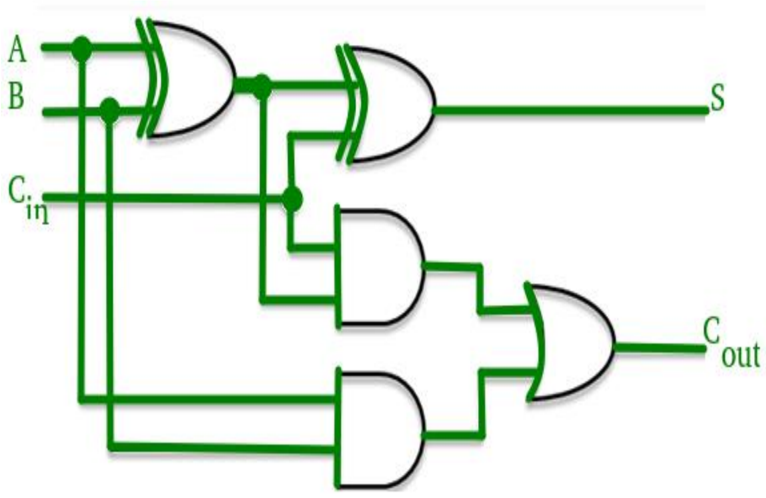
Ans: **Motivation behind Carry Look-Ahead Adder :**
In ripple carry adders, for each adder block, the two bits that are to be added are available instantly. However, each adder block waits for the carry to arrive from its previous block. So, it is not possible to generate the sum and carry of any block until the input carry is known. The block waits for the block to produce its carry. So there will be a considerable time delay which is carry propagation delay.

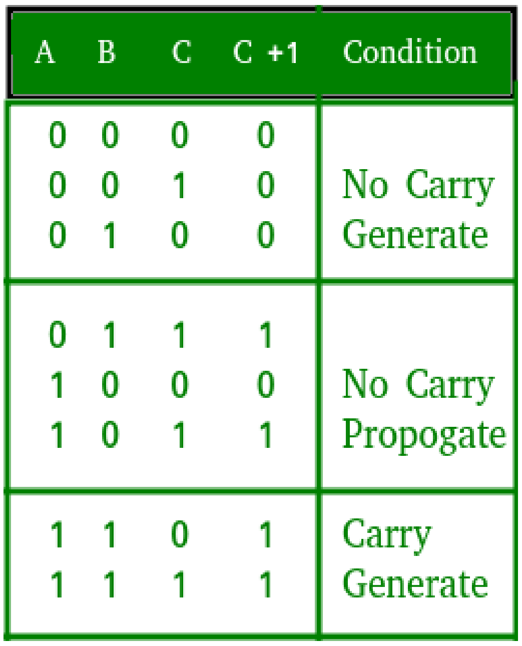


Consider the above 4-bit ripple carry adder. The sum is produced by the corresponding full adder as soon as the input signals are applied to it. But the carry input is not available on its final steady state value until carry is available at its steady state value. Similarly depends on and on . Therefore, though the carry must propagate to all the stages in order that output and carry settle their final steady-state value.

The propagation time is equal to the propagation delay of each adder block, multiplied by the number of adder blocks in the circuit. For example, if each full adder stage has a propagation delay of 20 nanoseconds, then will reach its final correct value after 60 (20 × 3) nanoseconds. The situation gets worse, if we extend the number of stages for adding more number of bits.

Carry Look-ahead Adder :
A carry look-ahead adder reduces the propagation delay by introducing more complex hardware. In this design, the ripple carry design is suitably transformed such that the carry logic over fixed groups of bits of the adder is reduced to two-level logic. Let us discuss the design in detail.





A	B	C	C +1	Condition
0	0	0	0	No Carry Generate
0	0	1	0	
0	1	0	0	
0	1	1	1	No Carry Propogate
1	0	0	0	
1	0	1	1	
1	1	0	1	Carry Generate
1	1	1	1	

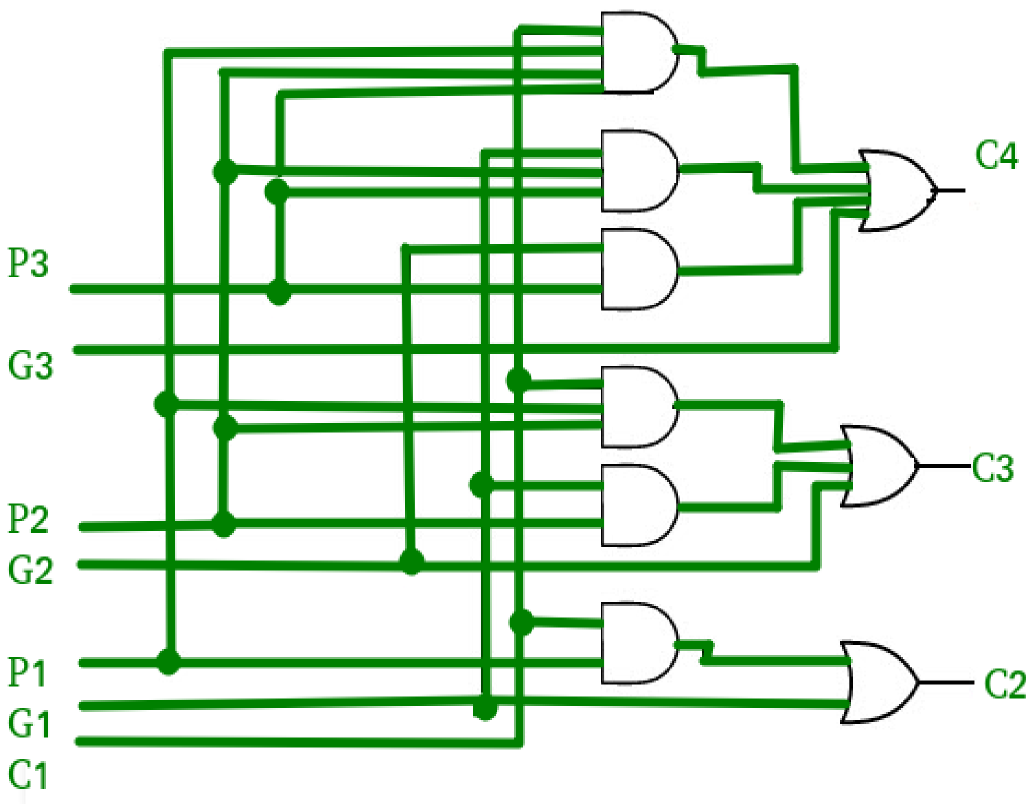
Consider the full adder circuit shown above with corresponding truth table. We define two variables as **‘carry generate’** and **‘carry propagate’** then,

The sum output and carry output can be expressed in terms of carry generate and carry propagate as where produces the carry when both , are 1 regardless of the input carry. is associated with the propagation of carry from to .

The carry output Boolean function of each stage in a 4 stage carry look-ahead adder can be expressed as

From the above Boolean equations we can observe that does not have to wait for and to propagate but actually is propagated at the same time as and . Since the Boolean expression for each carry output is the sum of products so these can be implemented with one level of AND gates followed by an OR gate.

The implementation of three Boolean functions for each carry output (, and) for a carry look-ahead carry generator shown in below figure.



Time Complexity Analysis :

We could think of a carry look-ahead adder as made up of two "parts"

1. The part that computes the carry for each bit.
2. The part that adds the input bits and the carry for each bit position.

The complexity arises from the part that generates the carry, not the circuit that adds the bits.

Now, for the generation of the carry bit, we need to perform a AND between (n+1) inputs. The complexity of the adder comes down to how we perform this AND operation. If we have AND gates, each with a fan-in (number of inputs accepted) of k, then we can find the AND of all the bits

in time. This is represented in asymptotic notation as .

Advantages and Disadvantages of Carry Look-Ahead Adder :
Advantages –

- The propagation delay is reduced.
- It provides the fastest addition logic.

Disadvantages –

- The Carry Look-ahead adder circuit gets complicated as the number of variables increase.
- The circuit is costlier as it involves more number of hardware.

31. Difference between Memory based and Register based Addressing Modes.

Ans: **Addressing modes** are the operations field specifies the operations which need to be performed. The operation must be executed on some data which is already stored in computer registers or in the memory. The way of choosing operands during program execution is dependent on addressing modes of the instruction. "The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually referenced. "Basically how we are interpreting the operand which is given in the instruction is known as addressing mode.
Addressing mode very much depend on the type of CPU organisation. There are three types of CPU organisation:

1. Single Accumulator organisation
2. General register organisation
3. Stack organisation

Addressing modes is used for one or both of the purpose. These can also be said as the **advantages** of using addressing mode:

1. To give programming versatility to the user by providing such facilities as pointers to memory, counter for loop control, indexing of data, and program relocation.
2. To reduce the number of bits in the addressing field of the instruction.

There are numbers of addressing modes available and it depends on the architecture and CPU organisation which of the addressing modes can be applied.

MEMORY BASED ADDRESSING MODES	REGISTER BASED ADDRESSING MODES
The operand is present in memory and its address is given in the instruction itself. This addressing mode is taking proper advantage of memory address, e.g., Direct addressing mode	An operand will be given in one of the register and register number will be provided in the instruction.With the register number present in instruction, operand is fetched, e.g., Register mode
The memory address specified in instruction may give the address where the effective address is stored in	The register contains the address of the operand. The effective address can be derived from the content of the register specified

MEMORY BASED ADDRESSING MODES	REGISTER BASED ADDRESSING MODES
the memory. In this case effective memory address is present in the memory address which is specified in the instruction, e.g., Indirect Addressing Mode	in the instruction. The content of the register might not be the effective address. This mode takes full advantage of registers, e.g., Register indirect mode
The content of base register is added to the address part of the instruction to obtain the effective address. A base register is assumed to hold a base address and the address field of the instruction gives displacement relative to the base address, e.g., Base Register Addressing Mode	If we are having a table of data and our program needs to access all the values one by one we need something which decrements the program counter/or any register which has base address. Though in this case register is basically decreased, it is register based addressing mode, e.g., In Auto decrements mode
The content of the index register is added to the address part that is given in the instruction to obtain the effective address. Index Mode is used to access an array whose elements are in successive memory locations, e.g., Indexed Addressing Mode	If we are having a table of data and our program needs to access all the values one by one we need something which increment the program counter/or any register which has base address, e.g., Auto increment mode
The content of program counter is added to the address part of the instruction in order to obtain the effective address. The address part of the instruction in this case is usually a signed number which can be either positive or negative, e.g., Relative addressing mode	Instructions generally used for initializing registers to a constant value is register based addressing mode,and this technique is very useful approach, e.g., Immediate mode.

Memory based addressing modes are mostly rely on Memory address and content present at some memory location. Register based addressing modes are mostly rely on Registers and content present at some register either it is data or some memory address.

Objective

32. The smallest integer that can be represented by an 8-bit number in 2's complement form is

a) -256 b) -128 c) -127 d) 0

Answer: (b)

33. The decimal value 0.5 in IEEE single precision floating point representation has

a) fraction bits of 000…000 and exponent value of 0
b) fraction bits of 000…000 and exponent value of −1
c) fraction bits of 100…000 and exponent value of 0
d) no exact representation

Answer: (b)

34. P is a 16-bit signed integer. The 2's complement representation of P is (F87B)16.The 2's complement representation of 8*P

a) (C3D8)16 b) (187B)16 c) (187B)10 d) (187B)8

Answer: (a)

35. (1217)8 is equivalent to

a) (1217)16 b) (028F)16 c)(2297)10 d) (0B17)16

Answer: (b)

36. Consider the equation (123)5 = (x8)y with x and y as unknown. The number of possible solutions is _____ .

a) 0 b) 1 c) 2 d) 3

Answer: (d)

37. The range of integers that can be represented by an n bit 2's complement number system is __________________.

Ans: $-(2^{(n-1)})$ to $+(2^{(n-1)}-1)$

38. The hexadecimal representation of (657)8 is

a) 1AF b) D78 c) D71 d) 32F

Ans: a)

Module 2

39. Explain the design of a simple hypothetical CPU.

Ans: It actually takes very little hardware to implement a simple CPU. Today in class we built a CPU with the very basics:

- An instruction fetch unit, which grabs the next instruction. The bits of the instruction are what activate different parts of the CPU circuit to make things happen.
- A register file, from which operands are read, and results are written.
- An arithmetic unit, which performs operations on the operands.

There are a few key tricks used throughout these designs:

- In a CPU, we need some way to keep operating on the same values. In a circuit, this means feeding the output of the circuit back around to its own input, recycling the same values over and over.
- Most CPUs operate on values of more than one bit. Using a single ***bus*** to represent multiple bits with a single line dramatically simplifies the circuit schematic compared to the physically more realistic approach of drawing separate wires for each bit. Logisim calls this "Data Bits"; most other tools call it a "bus".
- A ***multiplexor*** selects one input line from a set of inputs, based on "select line". For example, an 8-in mux has a 3 bit select input, which is a binary code indicating which of the 8 inputs to select as the output. Multiplexors are used:
    - In the register file, to select which register should provide input data for arithmetic. The register select lines are controlled by the input register portion of the instruction being executed.
    - In the arithmetic unit, to pick the output of one arithmetic circuit from those listed. The arithmetic select lines are controlled by the opcode portion of the instruction.

  Timing matters! In a real circuit timing is often the most important consideration for both performance and correctness. In the simulator, all the registers are edge-triggered. To avoid timing bugs, we fetch the next instruction on one edge of the clock, and write the results on the opposite edge of the clock, which prevents bugs where the register grabs a value as it changes. In a more complex design, we might need to keep a shift register to cycle between operating stages within a single instruction.

40. Difference between hardwired and microprogrammed design approaches.

Ans: To execute an instruction, there are two types of control units Hardwired Control unit and Micro-programmed control unit.

1. Hardwired control units are generally faster than microprogrammed designs. In hardwired control, we saw how all the control signals required inside the CPU can be generated using a state counter and a PLA circuit.
2. A microprogrammed control unit is a relatively simple logic circuit that is capable of (1) sequencing through microinstructions and (2) generating control signals to execute each microinstruction.

HARDWIRED CONTROL UNIT	**MICROPROGRAMMED CONTROL UNIT**
Hardwired control unit generates the control signals needed for the processor using logic circuits	Micrprogrammed control unit generates the control signals with the help of micro instructions stored in control memory
Hardwired control unit is faster when compared to microprogrammed control unit as the required control signals are generated with the help of hardwares	This is slower than the other as micro instructions are used for generating signals here
Difficult to modify as the control signals that need to be generated are hard wired	Easy to modify as the modification need to be done only at the instruction level
More costlier as everything has to be realized in terms of logic gates	Less costlier than hardwired control as only micro instructions are used for generating

HARDWIRED CONTROL UNIT	MICROPROGRAMMED CONTROL UNIT
	control signals
It cannot handle complex instructions as the circuit design for it becomes complex	It can handle complex instructions
Only limited number of instructions are used due to the hardware implementation	Control signals for many instructions can be generated
Used in computer that makes use of Reduced Instruction Set Computers(RISC)	Used in computer that makes use of Complex Instruction Set Computers(CISC)

41. What are the different types of semiconductor memory technologies?

Ans: There is a large variety of types of ROM and RAM that are available. Often the overall name for the memory technology includes the initials RAM or ROM and this gives a guide as to the overall type of format for the memory.
With technology moving forwards apace, not only are the established technologies moving forwards with SDRAM technology moving from DDR3 to DDR4 and then to DDR5, but Flash memory used in memory cards is also developing as are the other technologies.
In addition to this, new memory technologies are arriving on the scene and they are starting to make an impact in the market, enabling processor circuits to perform more effectively.
The different memory types or memory technologies are detailed below:

- ***DRAM:*** Dynamic RAM is a form of random access memory. DRAM uses a capacitor to store each bit of data, and the level of charge on each capacitor determines whether that bit is a logical 1 or 0.

  However these capacitors do not hold their charge indefinitely, and therefore the data needs to be refreshed periodically. As a result of this dynamic refreshing it gains its name of being a dynamic RAM. DRAM is the form of semiconductor memory that is often used in equipment including personal computers and workstations where it forms the main RAM for the computer. The semiconductor devices are normally available as integrated circuits for use in PCB assembly in the form of surface mount devices or less frequently now as leaded components.

- ***EEPROM:*** This is an Electrically Erasable Programmable Read Only Memory. Data can be written to these semiconductor devices and it can be erased using an electrical voltage. This is typically applied to an erase pin on the chip. Like other types of PROM, EEPROM retains the contents of the memory even when the power is turned off. Also like other types of ROM, EEPROM is not as fast as RAM.

- ***EPROM:*** This is an Erasable Programmable Read Only Memory. These semiconductor devices can be programmed and then erased at a later time. This is normally achieved by exposing the semiconductor device itself to ultraviolet light. To enable this to happen there is a circular window in the package of the EPROM to enable the light to reach the silicon of the device. When the PROM is in use, this window is normally covered by a label, especially when the data may need to be preserved for an extended period.

The PROM stores its data as a charge on a capacitor. There is a charge storage capacitor for each cell and this can be read repeatedly as required. However it is found that after many years the charge may leak away and the data may be lost.

Nevertheless, this type of semiconductor memory used to be widely used in applications where a form of ROM was required, but where the data needed to be changed periodically, as in a development environment, or where quantities were low.

- ***Flash memory:*** Flash memory may be considered as a development of EEPROM technology. Data can be written to it and it can be erased, although only in blocks, but data can be read on an individual cell basis.

  To erase and re-programme areas of the chip, programming voltages at levels that are available within electronic equipment are used. It is also non-volatile, and this makes it particularly useful. As a result Flash memory is widely used in many applications including USB memory sticks, compact Flash memory cards, SD memory cards and also now solid state hard drives for computers and many other applications.

- ***F-RAM:*** Ferroelectric RAM is a random-access memory technology that has many similarities to the standard DRAM technology. The major difference is that it incorporates a ferroelectric layer instead of the more usual dielectric layer and this provides its non-volatile capability. As it offers a non-volatile capability, F-RAM is a direct competitor to Flash.

- ***MRAM:*** This is Magneto-resistive RAM, or Magnetic RAM. It is a non-volatile RAM memory technology that uses magnetic charges to store data instead of electric charges.

  Unlike technologies including DRAM, which require a constant flow of electricity to maintain the integrity of the data, MRAM retains data even when the power is removed. An additional advantage is that it only requires low power for active operation. As a result this technology could become a major player in the electronics industry now that production processes have been developed to enable it to be produced.

- ***P-RAM / PCM:*** This type of semiconductor memory is known as Phase change Random Access Memory, P-RAM or just Phase Change memory, PCM. It is based around a phenomenon where a form of chalcogenide glass changes is state or phase between an amorphous state (high resistance) and a polycrystalline state (low resistance). It is possible to detect the state of an individual cell and hence use this for data storage. Currently this type of memory has not been widely commercialised, but it is expected to be a competitor for flash memory.

- ***PROM:*** This stands for Programmable Read Only Memory. It is a semiconductor memory which can only have data written to it once - the data written to it is permanent. These memories are bought in a blank format and they are programmed using a special PROM programmer.

  Typically a PROM will consist of an array of fuseable links some of which are "blown" during the programming process to provide the required data pattern.
- ***SDRAM:*** Synchronous DRAM. This form of semiconductor memory can run at faster speeds than conventional DRAM. It is synchronised to the clock of the processor and is capable of keeping two sets of memory addresses open simultaneously. By transferring data alternately from one set of

addresses, and then the other, SDRAM cuts down on the delays associated with non-synchronous RAM, which must close one address bank before opening the next.

Within the SDRAM family there are several types of memory technologies that are seen. These are referred to by the letters DDR - Double Data Rate. DDR4 is currently the latest technology, but this is soon to be followed by DDR5 which will offer some significant improvements in performance.

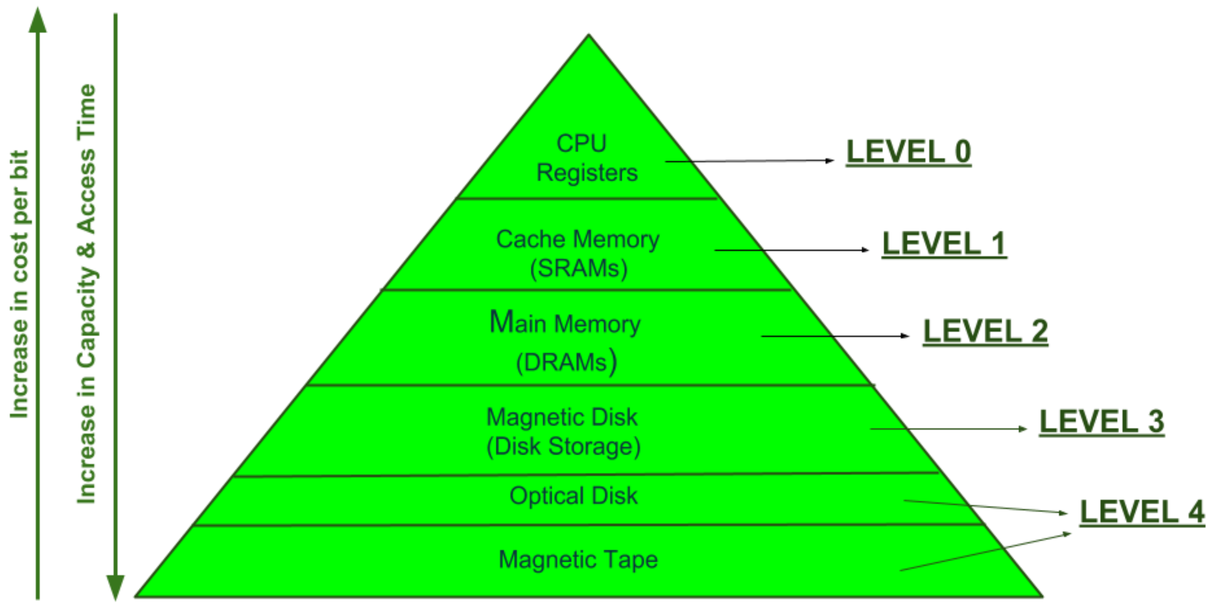
- ***SRAM:*** Static Random Access Memory. This form of semiconductor memory gains its name from the fact that, unlike DRAM, the data does not need to be refreshed dynamically.

  These semiconductor devices are able to support faster read and write times than DRAM (typically 10 ns against 60 ns for DRAM), and in addition its cycle time is much shorter because it does not need to pause between accesses. However they consume more power, they are less dense and more expensive than DRAM. As a result of this SRAM is normally used for caches, while DRAM is used as the main semiconductor memory technology.

  Semiconductor memory technology is developing at a fast rate to meet the ever growing needs of the electronics industry. Not only are the existing technologies themselves being developed, but considerable amounts of research are being invested in new types of semiconductor memory technology.
  In terms of the memory technologies currently in use, SDRAM versions like DDR4 are being further developed to provide DDR5 which will offer significant performance improvements. In time, DDR5 will be developed to provide the next generation of SDRAM.
  Other forms of memory are seen around the home in the form of USB memory sticks, Compact Flash, CF cards or SD memory cards for cameras and other applications as well as solid state hard drives for computers.
  The semiconductor devices are available in a wide range of formats to meet the differing PCB assembly and other needs.

42. Discuss about the need of the memory organization.

Ans: In the Computer System Design, Memory Hierarchy is an enhancement to organize the memory such that it can minimize the access time. The Memory Hierarchy was developed based on a program behavior known as locality of references.The figure below clearly demonstrates the different levels of memory hierarchy :



MEMORY HIERARCHY DESIGN

This Memory Hierarchy Design is divided into 2 main types:

1. **External Memory or Secondary Memory** –
   Comprising of Magnetic Disk, Optical Disk, Magnetic Tape i.e. peripheral storage devices which are accessible by the processor via I/O Module.
2. **Internal Memory or Primary Memory** –
   Comprising of Main Memory, Cache Memory & CPU registers. This is directly accessible by the processor.
   We can infer the following characteristics of Memory Hierarchy Design from above figure:

1. **Capacity:**
   It is the global volume of information the memory can store. As we move from top to bottom in the Hierarchy, the capacity increases.

2. **Access Time:**
   It is the time interval between the read/write request and the availability of the data. As we move from top to bottom in the Hierarchy, the access time increases.
3. **Performance:**
   Earlier when the computer system was designed without Memory Hierarchy design, the speed gap increases between the CPU registers and Main Memory due to large difference in access time. This results in lower performance of the system and thus, enhancement was required. This enhancement was made in the form of Memory Hierarchy Design because of which the performance of the system increases. One of the most significant ways to increase system performance is minimizing how far down the memory hierarchy one has to go to manipulate data.
4. **Cost per bit:**
   As we move from bottom to top in the Hierarchy, the cost per bit increases i.e. Internal Memory is costlier than External Memory.

43: Difference between Input and output subsystems.

Ans: Input and output (I/O) devices allow us to communicate with the computer system. I/O is the transfer of data between primary memory and various I/O peripherals. Input devices such as keyboards, mice, card readers, scanners, voice recognition systems, and touch screens enable us to enter data into the computer. Output devices such as monitors, printers, plotters, and speakers allow us to get information from the computer.

These devices are not connected directly to the CPU. Instead, there is an interface that handles the data transfers. This interface converts the system bus signals to and from a format that is acceptable to the given device. The CPU communicates to these external devices via I/O registers. ...

44. List out the types of I/O devices.
Ans: Types of I/O
There are three types of I/O operations:
Sensory input. digital input. analog input.
Control output. direct digital output. modulated digital output. analog output.
Data transfer. parallel. serial.

45. What is the role of I/O device interface?

Ans: Input Output Interface provides a method for transferring information between internal storage and external I/O devices. Peripherals connected to a computer need special communication links for interfacing them with the central processing unit.

46. What do you mean by software interrupts?

Ans: Software Interrupt Definition. A software interrupt, also called an exception, is an interrupt that is caused by software, usually by a program in user mode. ... Examples of events that cause them are requests by an application program for certain services from the operating system or the termination of such programs.

47. What do know about privileged and non privileged instructions?

Ans: Privileged Instructions possess the following characteristics :
(i) If any attempt is made to execute a Privileged Instruction in User Mode, then it will not be executed and treated as an illegal instruction. The Hardware traps it to the Operating System.

(ii) Before transferring the control to any User Program, it is the responsibility of the Operating System to ensure that the Timer is set to interrupt. Thus, if the timer interrupts then the Operating System regains the control.
Thus, any instruction which can modify the contents of the Timer is a Privileged Instruction.

(iii) Privileged Instructions are used by the Operating System in order to achieve correct operation.

(iv) Various examples of Privileged Instructions include:

I/O instructions and Halt instructions
Turn off all Interrupts
Set the Timer
Context Switching
Clear the Memory or Remove a process from the Memory

Modify entries in Device-status table.

Various examples of Non-Privileged Instructions include:

- Reading the status of Processor
- Reading the System Time
- Generate any Trap Instruction
- Sending the final prinout of Printer

Also, it is important to note that in order to change the mode from Privileged to Non-Privileged, we require a Non-privileged Instruction that does not generate any interrupt.

48. What is the role of interrupts in process state transitions?

Ans: There are two ways for a process to transition from the running state to the ready state depending on the OS implements multitasking:

- With preemptive multitasking, the OS uses timer interrupts (there is one timer for each core or processor in the system) to regularly interrupt whatever process is currently running. The interrupt handler then invokes the OS scheduler to determine whether to schedule another process or continue running the same process. If the scheduler decided to run another process, then the current process transition from the running state to the ready state.
- With cooperative multitasking, the OS does not use interrupts to scheduler processes. Instead, a running process should voluntarily yield control to the scheduler to allow it to schedule another process. So processes do not transition between the running and ready states using interrupts, but only voluntarily.

It seems to me that the figure from the Modern Operating Systems book applies to both multitasking methods while the figure from the Operating System Concepts is specifically about preemptive multitasking. Although by changing the word "interrupt" to something more inclusive like "yield," then the other figure would also apply to cooperative multitasking.

49. Difference between Interrupts and Exceptions.

Ans: **Exceptions and interrupts** are unexpected events which will disrupt the normal flow of execution of instruction(that is currently executing by processor). An exception is an unexpected event from within the processor. Interrupt is an unexpected event from outside the process.
Whenever an exception or interrupt occurs, the hardware starts executing the code that performs an action in response to the exception. This action may involve killing a process, outputting an error message, communicating with an external device, or horribly crashing the entire computer system by initiating a “Blue Screen of Death” and halting the CPU. The instructions responsible for this action reside in the operating system kernel, and the code that performs this action is called the interrupt handler code. Now, We can think of handler code as an operating system subroutine. Then, After the handler code is executed, it may be possible to continue execution after the instruction where the execution or interrupt occurred.

Exception and Interrupt Handling :

Whenever an exception or interrupt occurs, execution transition from user mode to kernel mode where the exception or interrupt is handled. In detail, the following steps must be taken to handle an exception or interrupts.

While entering the kernel, the context (values of all CPU registers) of the currently executing process must first be saved to memory. The kernel is now ready to handle the exception/interrupt.

1. Determine the cause of the exception/interrupt.

2. Handle the exception/interrupt.

When the exception/interrupt have been handled the kernel performs the following steps:

1. Select a process to restore and resume.
2. Restore the context of the selected process.
3. Resume execution of the selected process.

At any point in time, the values of all the registers in the CPU defines the context of the CPU. Another name used for CPU context is CPU state.

The exception/interrupt handler uses the same CPU as the currently executing process. When entering the exception/interrupt handler, the values in all CPU registers to be used by the exception/interrupt handler must be saved to memory. The saved register values can later restored before resuming execution of the process.

The handler may have been invoked for a number of reasons. The handler thus needs to determine the cause of the exception or interrupt. Information about what caused the exception or interrupt can be stored in dedicated registers or at predefined addresses in memory.

Next, the exception or interrupt needs to be serviced. For instance, if it was a keyboard interrupt, then the key code of the key press is obtained and stored somewhere or some other appropriate action is taken. If it was an arithmetic overflow exception, an error message may be printed or the program may be terminated.

The exception/interrupt have now been handled and the kernel. The kernel may choose to resume the same process that was executing prior to handling the exception/interrupt or resume execution of any other process currently in memory.

The context of the CPU can now be restored for the chosen process by reading and restoring all register values from memory.

The process selected to be resumed must be resumed at the same point it was stopped. The address of this instruction was saved by the machine when the interrupt occurred, so it is simply a matter of getting this address and make the CPU continue to execute at this address.

50. What are the different types of interrupts present in 8086 microprocessor?

Ans: An interrupt is a condition that halts the microprocessor temporarily to work on a different task and then return to its previous task. Interrupt is an event or signal that request to attention of CPU. This halt allows peripheral devices to access the microprocessor.

Whenever an interrupt occurs the processor completes the execution of the current instruction and starts the execution of an Interrupt Service Routine (ISR) or Interrupt Handler. ISR is a program that tells the processor what to do when the interrupt occurs. After the execution of ISR, control returns back to the main routine where it was interrupted.

In 8086 microprocessor following tasks are performed when microprocessor encounters an interrupt:

1. The value of flag register is pushed into the stack. It means that first the value of SP (Stack Pointer) is decremented by 2 then the value of flag register is pushed to the memory address of stack segment.
2. The value of starting memory address of CS (Code Segment) is pushed into the stack.
3. The value of IP (Instruction Pointer) is pushed into the stack.
4. IP is loaded from word location (Interrupt type) * 04.
5. CS is loaded from the next word location.
6. Interrupt and Trap flag are reset to 0.

The different types of interrupts present in 8086 microprocessor are given by:

1. **Hardware Interrupts** –
Hardware interrupts are those interrupts which are caused by any peripheral device by sending a

signal through a specified pin to the microprocessor. There are two hardware interrupts in 8086 microprocessor. They are:

- *(A) NMI (Non Maskable Interrupt)* – It is a single pin non maskable hardware interrupt which cannot be disabled. It is the highest priority interrupt in 8086 microprocessor. After its execution, this interrupt generates a TYPE 2 interrupt. IP is loaded from word location 00008 H and CS is loaded from the word location 0000A H.
- *(B) INTR (Interrupt Request)* – It provides a single interrupt request and is activated by I/O port. This interrupt can be masked or delayed. It is a level triggered interrupt. It can receive any interrupt type, so the value of IP and CS will change on the interrupt type received.

2. **Software Interrupts** – These are instructions that are inserted within the program to generate interrupts. There are 256 software interrupts in 8086 microprocessor. The instructions are of the format INT type where type ranges from 00 to FF. The starting address ranges from 00000 H to 003FF H. These are 2 byte instructions. IP is loaded from type * 04 H and CS is loaded from the next address give by (type * 04) + 02 H. Some important software interrupts are:
   - (A) *TYPE 0* corresponds to division by zero(0).
   - (B) *TYPE 1* is used for single step execution for debugging of program.
   - (C) *TYPE 2* represents NMI and is used in power failure conditions.
   - (D) *TYPE 3* represents a break-point interrupt.
   - (E) *TYPE 4* is the overflow interrupt.

51. Discuss the different Mode of Data Transfer.

Ans: The method that is used to transfer information between internal storage and external I/O devices is known as I/O interface. The CPU is interfaced using special communication links by the peripherals connected to any computer system. These communication links are used to resolve the differences between CPU and peripheral. There exists special hardware components between CPU and peripherals to supervise and synchronize all the input and output transfers that are called interface units.

Mode of Transfer:

The binary information that is received from an external device is usually stored in the memory unit. The information that is transferred from the CPU to the external device is originated from the memory unit. CPU merely processes the information but the source and target is always the memory unit. Data transfer between CPU and the I/O devices may be done in different modes.

Data transfer to and from the peripherals may be done in any of the three possible ways

1. Programmed I/O.
2. Interrupt- initiated I/O.
3. Direct memory access(DMA).

Now let's discuss each mode one by one.

1. **Programmed I/O:** It is due to the result of the I/O instructions that are written in the computer program. Each data item transfer is initiated by an instruction in the program. Usually the transfer is from a CPU register and memory. In this case it requires constant monitoring by the CPU of the peripheral devices.

   Example of Programmed I/O: In this case, the I/O device does not have direct access to the memory unit. A transfer from I/O device to memory requires the execution of several instructions by the CPU, including an input instruction to transfer the data from device to the CPU and store instruction to transfer the data from CPU to memory. In programmed I/O, the CPU stays in the program loop until the I/O unit indicates that it is ready for data transfer. This is a time consuming process since it needlessly keeps the CPU busy. This situation can be avoided by using an interrupt facility. This is discussed below.

2. **Interrupt- initiated I/O:** Since in the above case we saw the CPU is kept busy unnecessarily. This situation can very well be avoided by using an interrupt driven method for data transfer. By using interrupt facility and special commands to inform the interface to issue an interrupt request signal whenever data is available from any device. In the meantime the CPU can proceed for any other program execution. The interface meanwhile keeps monitoring the device. Whenever it is determined that the device is ready for data transfer it initiates an interrupt request signal to the computer. Upon detection of an external interrupt signal the CPU stops momentarily the task that it was already performing, branches to the service program to process the I/O transfer, and then return to the task it was originally performing.

   Note: Both the methods programmed I/O and Interrupt-driven I/O require the active intervention of the
   processor to transfer data between memory and the I/O module, and any data transfer must transverse a path through the processor. Thus both these forms of I/O suffer from two inherent drawbacks.

   - The I/O transfer rate is limited by the speed with which the processor can test and service a device.
   - The processor is tied up in managing an I/O transfer; a number of instructions must be executed
     for each I/O transfer.

3. **Direct Memory Access**: The data transfer between a fast storage media such as magnetic disk and memory unit is limited by the speed of the CPU. Thus we can allow the peripherals directly communicate with each other using the memory buses, removing the intervention of the CPU. This type of data transfer technique is known as DMA or direct memory access. During DMA the CPU is idle and it has no control over the memory buses. The DMA controller takes over the buses to manage the transfer directly between the I/O devices and the memory unit.

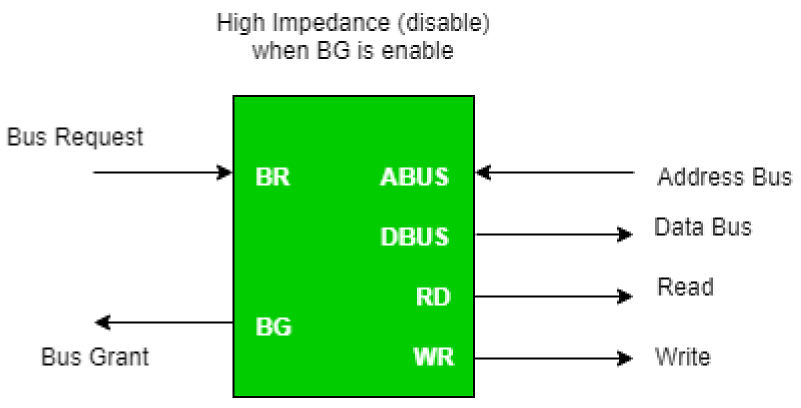


Figure - CPU Bus Signals for DMA Transfer

Bus Request : It is used by the DMA controller to request the CPU to relinquish the control of the buses.
Bus Grant : It is activated by the CPU to Inform the external DMA controller that the buses are in high impedance state and the requesting DMA can take control of the buses. Once the DMA has taken the control of the buses it transfers the data. This transfer can take place in many ways.

52. Difference between Maskable and Non Maskable Interrupt.

Ans: **Difference between maskable and nonmaskable interrupt :**

SR.NO.	MASKABLE INTERRUPT	NON MASKABLE INTERRUPT
1	Maskable interrupt is a hardware Interrupt that can be disabled or ignored by the instructions of CPU.	A non-maskable interrupt is a hardware interrupt that cannot be disabled or ignored by the instructions of CPU.
2	When maskable interrupt occur, it can be handled after executing the current instruction.	When non-maskable interrupts occur, the current instructions and status are stored in stack for the CPU to handle the interrupt.
3	Maskable interrupts help to handle lower priority tasks.	Non-maskable interrupt help to handle higher priority tasks such as watchdog timer.
4	Maskable interrupts used to interface with peripheral device.	Non maskable interrupt used for emergency purpose e.g power failure, smoke detector etc .
5	In maskable interrupts, response time is high.	In non maskable interrupts, response time is low.
6	It may be vectored or non-vectored.	All are vectored interrupts.
7	Operation can be masked or made pending.	Operation Cannot be masked or made pending.
8	RST6.5, RST7.5, and RST5.5 of	Trap of 8085 microprocessor is an

SR.NO.	MASKABLE INTERRUPT	NON MASKABLE INTERRUPT
	8085 are some common examples of maskable Interrupts.	example for non-maskable interrupt.

53. Difference between Interrupt and Polling.

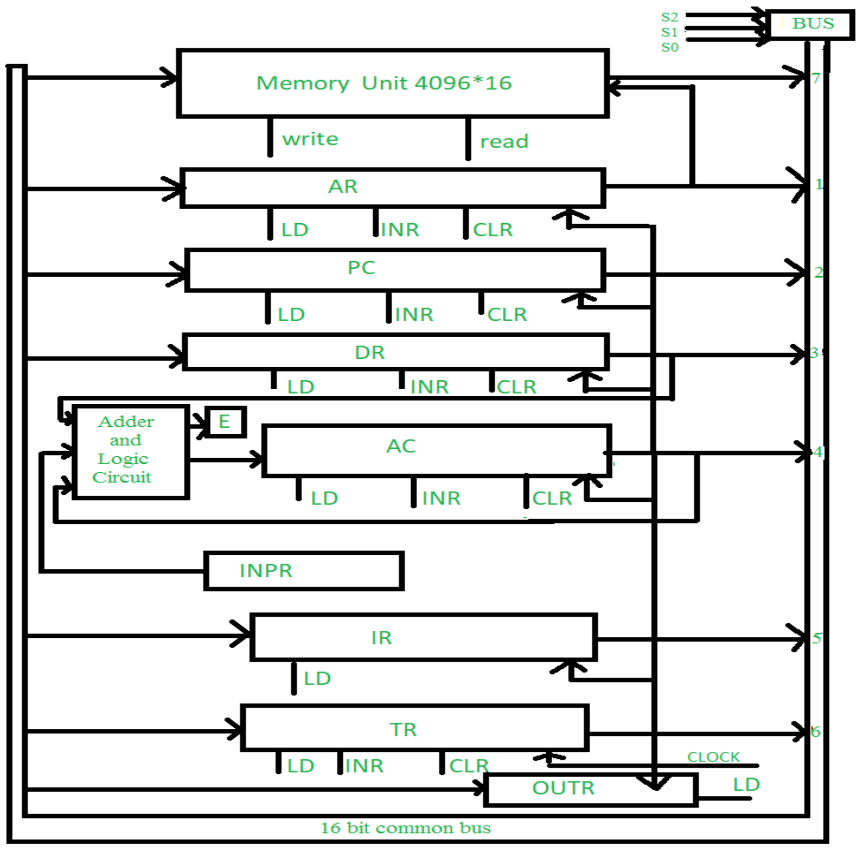
Ans: **Interrupt:**
Interrupt is a hardware mechanism in which, the device notices the CPU that it requires its attention. Interrupt can take place at any time. So when CPU gets an interrupt signal trough the indication interrupt-request line, CPU stops the current process and respond to the interrupt by passing the control to interrupt handler which services device.
Polling:
In polling is not a hardware mechanism, its a protocol in which CPU steadily checks whether the device needs attention. Wherever device tells process unit that it desires hardware processing, in polling process unit keeps asking the I/O device whether or not it desires CPU processing. The CPU ceaselessly check every and each device hooked up thereto for sleuthing whether or not any device desires hardware attention.
Each device features a command-ready bit that indicates the standing of that device, i.e., whether or not it's some command to be dead by hardware or not. If command bit is ready one, then it's some command to be dead else if the bit is zero, then it's no commands.
Let's see that the difference between interrupt and polling:

S.NO	INTERRUPT	POLLING
1.	In interrupt, the device notices the CPU that it requires its attention.	Whereas, in polling, CPU steadily checks whether the device needs attention.
2.	An interrupt is not a protocol, its a hardware mechanism.	Whereas it isn't a hardware mechanism, its a protocol.
3.	In interrupt, the device is serviced by interrupt handler.	While in polling, the device is serviced by CPU.
4.	Interrupt can take place at any time.	Whereas CPU steadily ballots the device at regular or

S.NO	INTERRUPT	POLLING
		proper interval.
5.	In interrupt, interrupt request line is used as indication for indicating that device requires servicing.	While in polling, Command ready bit is used as indication for indicating that device requires servicing.
6.	In interrupts, processor is simply disturbed once any device interrupts it.	On the opposite hand, in polling, processor waste countless processor cycles by repeatedly checking the command-ready little bit of each device.

54. What is the role of Common Bus System?

Ans: A basic computer has 8 registers, memory unit and a control unit. The diagram of the common bus system is as shown below.



Connections:
The outputs of all the registers except the OUTR (output register) are connected to the common bus.

The output selected depends upon the binary value of variables S2, S1 and S0. The lines from common bus are connected to the inputs of the registers and memory. A register receives the information from the bus when its LD (load) input is activated while in case of memory the Write input must be enabled to receive the information. The contents of memory are placed onto the bus when its Read input is activated.

Various Registers:

4 registers DR, AC, IR and TR have 16 bits and 2 registers AR and PC have 12 bits. The INPR and OUTR have 8 bits each. The INPR receives character from input device and delivers it to the AC while the OUTR receives character from AC and transfers it to the output device. 5 registers have 3 control inputs LD (load), INR (increment) and CLR (clear). These types of registers are similar to a binary counter.

ABBREVIATION	REGISTER NAME
OUTR	Output register
TR	Temporary register
IR	Instruction register
INPR	Input register
AC	Accumulator
DR	Data register
PC	Program counter
AR	Address register

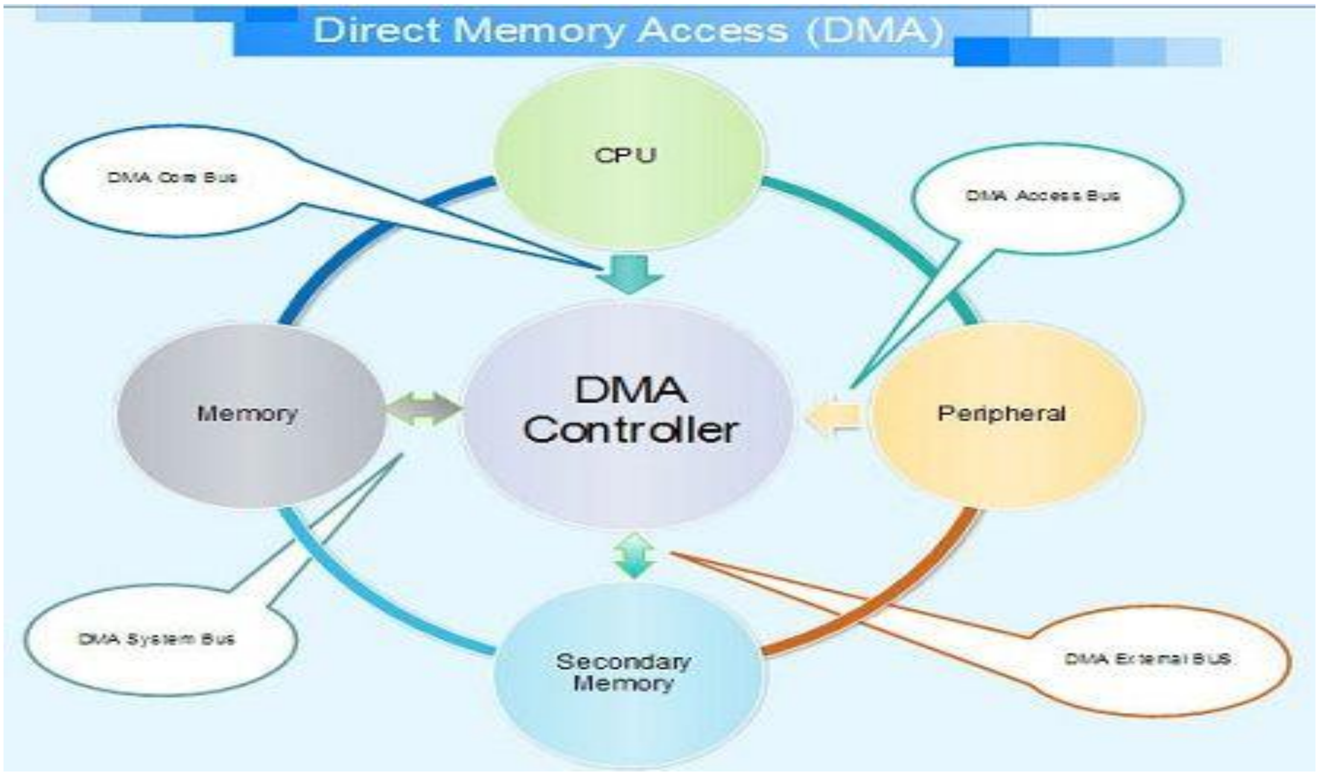
Adder and logic circuit:

The adder and logic circuit provides the 16 inputs of AC. This circuit has 3 sets of inputs. One set comes from the outputs of AC which implements register micro operations. The other set comes from the DR (data register) which are used to perform arithmetic and logic micro operations. The result of these operations is sent to AC while the end around carry is stored in E as shown in diagram. The third set of inputs is from INPR.

Note:
The content of any register can be placed on the common bus and an operation can be performed in the adder and logic circuit during the same clock cycle.

55. What do you mean by DMA?

Ans: DMA stands for "Direct Memory Access" and is a method of transferring data from the computer's RAM to another part of the computer without processing it using the CPU. While most data that is input or output from your computer is processed by the CPU, some data does not require processing, or can be processed by another device.
In these situations, DMA can save processing time and is a more efficient way to move data from the computer's memory to other devices. In order for devices to use direct memory access, they must be assigned to a DMA channel. Each type of port on a computer has a set of DMA channels that can be assigned to each connected device. For example, a PCI controller and a hard drive controller each have their own set of DMA channels.



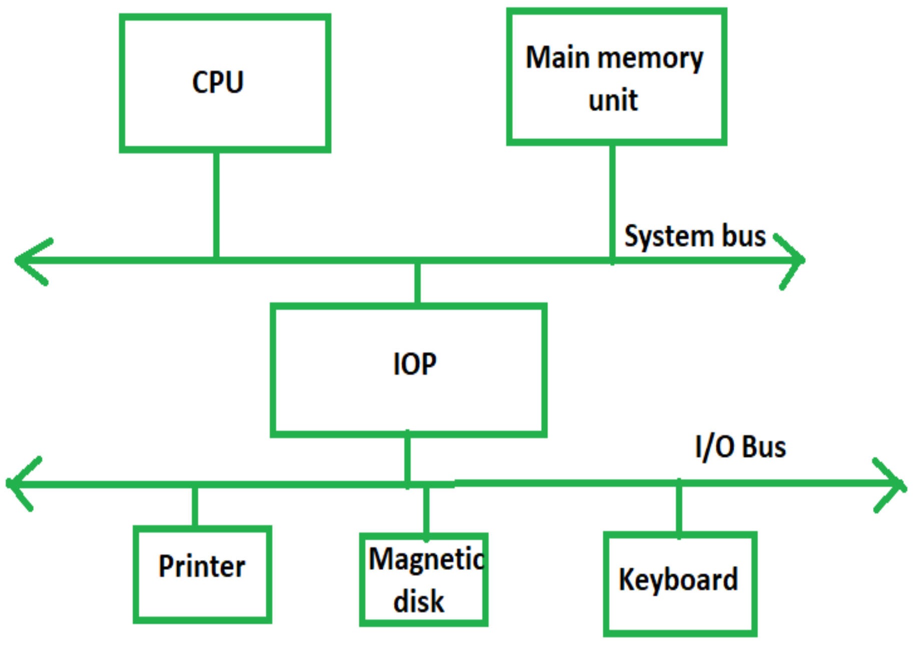
For example, a sound card may need to access data stored in the computer's RAM, but since it can process the data itself, it may use DMA to bypass the CPU. Video cards that support DMA can also access the system memory and process graphics without needing the CPU. Ultra DMA hard drives use DMA to transfer data faster than previous hard drives that required the data to first be run through the CPU.
An alternative to DMA is the Programmed Input/Output (PIO) interface in which all data transmitted between devices goes through the processor. A newer protocol for the ATAIIDE interface is Ultra DMA, which provides a burst data transfer rate up to 33 mbps. Hard drives that come with Ultra DMAl33 also support PIO modes 1, 3, and 4, and multiword DMA mode 2 at 16.6 mbps.

56. What do know about Input-Output Processor (IOP) or IO channel?

Ans: The **DMA mode** of data transfer reduces CPU’s overhead in handling I/O operations. It also allows parallelism in CPU and I/O operations. Such parallelism is necessary to avoid wastage of valuable CPU time while handling I/O devices whose speeds are much slower as compared to CPU. The concept of DMA operation can be extended to relieve the CPU further from getting involved with the execution of I/O operations. This gives rises to the development of special purpose processor called **Input-Output Processor (IOP) or IO channel**.

The Input Output Processor (IOP) is just like a CPU that handles the details of I/O operations. It is more equipped with facilities than those are available in typical DMA controller. The IOP can fetch and execute its own instructions that are specifically designed to characterize I/O transfers. In addition to the I/O – related tasks, it can perform other processing tasks like arithmetic, logic, branching and code translation. The main memory unit takes the pivotal role. It communicates with processor by the means of DMA.

The block diagram –



Input-Output Processor

The Input Output Processor is a specialized processor which loads and stores data into memory along with the execution of I/O instructions. It acts as an interface between system and devices. It involves a sequence of events to executing I/O operations and then store the results into the memory.

Advantages –

- The I/O devices can directly access the main memory without the intervention by the processor in I/O processor based systems.
- It is used to address the problems that are arises in Direct memory access method.

Objective

57. Technically speaking, what is the most accurate description of a peripheral device?

a) A device that is connected to a computer but is not part of the core computer architecture
b) A device that plugs into a computer using a USB connection
c) A device that provides input and output functions for a computer system
d) A device connected to a computer using a cable.

Ans: a)

58. What are the three general types of peripheral devices?

a) Input, output and storage
b) Mouse, keyboard and monitor
c) Audio, video and print output.
d) Internal, wired external and wireless external.

Ans: a)

59. Which of the following is NOT a computer peripheral?

a) Central processing unit
b) Scanner
b) Printer
d) Microphone

Ans: a)

60. The interrupt-request line is a part of the ___________

a) Data line b) Control line c) Address line d) None of the mentioned

Ans: b)

61. The return address from the interrupt-service routine is stored on the ___________

a) System heap b) Processor register c) Processor stack d) Memory

Ans: c)

62. The signal sent to the device from the processor to the device after receiving an interrupt is ___________

a) Interrupt-acknowledge b) Return signal c) Service signal d) Permission signal

Ans: a)

63. When the process is returned after an interrupt service ______ should be loaded again.

i) Register contents
ii) Condition codes
iii) Stack contents
iv) Return addresses

a) i, iv b) ii, iii and iv c) iii, iv d) i, ii

Ans: d)

64. The time between the receiver of an interrupt and its service is ______

a) Interrupt delay b) Interrupt latency c) Cycle time d) Switching time

Ans: b)

65. Interrupts form an important part of _____ systems.

a) Batch processing b) Multitasking c) Real-time processing d) Multi-user

Ans: c)

66. An interrupt that can be temporarily ignored is ___________

a) Vectored interrupt b) Non-maskable interrupt
c) Maskable interrupt d) High priority interrupt

Ans: c)

Module 3

67. What are the Difference between RISC and CISC processor?

Ans: Below are few differences between RISC and CISC:

CISC	RISC
A large number of instructions are present in the architecture.	Very fewer instructions are present. The number of instructions are generally less than 100.
Some instructions with long execution times. These include instructions that copy an entire block from one part of memory to another and others that copy multiple registers to and from memory.	No instruction with a long execution time due to very simple instruction set. Some early RISC machines did not even have an integer multiply instruction, requiring compilers to implement multiplication as a sequence of additions.
Variable-length encodings of the instructions. **Example:** IA32 instruction size can range from 1 to 15 bytes.	Fixed-length encodings of the instructions are used. **Example:** In IA32, generally all instructions are encoded as 4 bytes.
Multiple formats are supported for specifying operands. A memory operand specifier can have many different combinations of displacement, base and index registers.	Simple addressing formats are supported. Only base and displacement addressing is allowed.
CISC supports array.	RISC does not supports array.
Arithmetic and logical operations can be applied to both memory and register operands.	Arithmetic and logical operations only use register operands. Memory referencing is only allowed by load and store instructions, i.e. reading from memory into a register and writing from a register to memory respectively.
Implementation programs are hidden from machine level programs. The ISA provides a clean abstraction between	Implementation programs exposed to machine level programs. Few RISC machines do not allow specific

CISC	RISC
programs and how they get executed.	instruction sequences.
Condition codes are used.	No condition codes are used.
The stack is being used for procedure arguments and return addresses.	Registers are being used for procedure arguments and return addresses. Memory references can be avoided by some procedures.

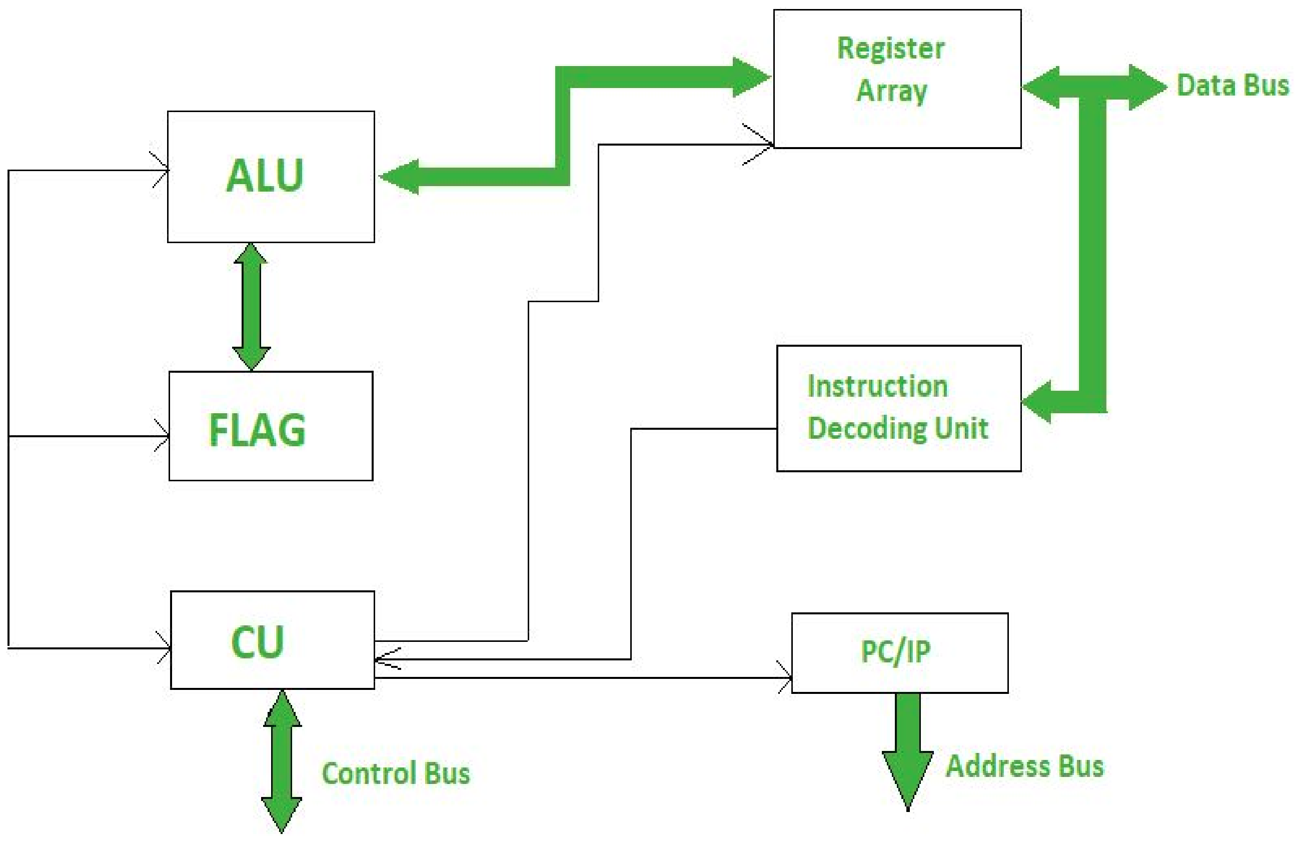
68. Define the concept of classification of Microprocessors.

Ans: A Microprocessor is an important part of a computer architecture without which you will not be able to perform anything on your computer. It is a programmable device that takes in input perform some arithmetic and logical operations over it and produce desired output. In simple words, a Microprocessor is a digital device on a chip which can fetch instruction from memory, decode and execute them and give results.

Basics of Microprocessor –
A Microprocessor takes a bunch of instructions in machine language and executes them, telling the processor what it has to do. Microprocessor performs three basic things while executing the instruction:

1. It performs some basic operations like addition, subtraction, multiplication, division and some logical operations using its Arithmetic and Logical Unit (ALU). New Microprocessors also perform operations on floating point numbers also.
2. Data in Microprocessor can move from one location to another.
3. It has a Program Counter (PC) register that stores the address of next instruction based on the value of PC, Microprocessor jumps from one location to another and takes decision.

A typical Microprocessor structure looks like this.



Clock Speed of different Microprocessor:

- **16-bit Microprocessor –**
- 8086: 4.7MHz, 8MHz, 10MHz
-
- 8088: more than 5MHz
-
- 80186/80188: 6MHz
-

80286: 8MHz

- **32-bit Microprocessor –**
- INTEL 80386: 16MHz to 33MHz
-
- INTEL 80486: 16MHz to 100MHz
-

PENTIUM: 66MHz

- **64-bit Microprocessor –**
- INTEL CORE-2: 1.2GHz to 3GHz
-
- INTEL i7: 66GHz to 3.33GHz
-
- INTEL i5: 2.4GHz to 3.6GHz
-

INTEL i3: 2.93GHz to 3.33GHz

We do not have any 128-bit Microprocessor in work at present one among the reasons for this is that we are a long way from exhausting the 64 bit address space itself, we use it a constant rate of roughly 2 bits every 3 years. At present we have only used 48 bits of 64 bits so why require 128 bit address space. Also 128 bit Microprocessor would be much slower than the 64 bit Microprocessor.

Types of Processor:

- **Complex Instruction Set Computer (CISC) –** CISC or Complex Instruction Set Computer is a computer architecture where instructions are such that a single instruction can execute multiple low level operations like loading from memory, storing into memory or an arithmetic operation etc. It has multiple addressing nodes within single instruction.CISC makes use of very few registers.

Example:
1. Intel 386
2. Intel 486
3. Pentium
4. Pentium Pro
5. Pentium II
6. Pentium III
7. Motorola 68000
8. Motorola 68020
9. Motorola 68040 etc.

- **Reduced Instruction Set Computer (RISC) –** RISC or Reduced Instruction Set Computer is a computer architecture where instruction are simple and designed to get executed quickly. Instructions get completed in one clock cycle this is because of the optimization of instructions and pipelining (a technique that allows for simultaneous execution of parts, or stages, of instructions to more efficiently process instructions). RISC makes use of multiple registers to avoid large interactions with memory. It has few addressing nodes.

Example:
1. IBM RS6000
2. MC88100
3. DEC Alpha 21064
4. DEC Alpha 21164
5. DEC Alpha 21264

- **Explicitly Parallel Instruction Computing (EPIC) –** EPIC or Explicitly Parallel Instruction Computing permits computer to execute instructions parallel using compilers.It allows complex instructions execution without using higher clock frequencies.EPIC encodes its instruction into 128 bit bundles.each bundle contains three instructions which are encoded in 41 bits each and a 5-bit template field(contains information about types of instructions in bundle and which instructions can be executed in parallel).

Example:
1. IA-64 (Intel Architecture-64)

69. What are the characterstics of a micro processor?

Ans: **Characteristics of Microprocessors**
A Microprocessor’s performance depends on the following characteristics:

· Clock speed

· Instruction set

· Word size

a) Clock Speed

Every microprocessor has an **internal clock** that regulates the speed at which it executes instructions. The speed at which the microprocessor executes instructions is called the **clock speed**. Clock speed is measured in MHz (Mega Hertz) or in GHz (Giga Hertz).

b) Instruction Set

A command which is given to a computer to perform an operation on data is called an **instruction**. Basic set of machine level instructions that a microprocessor is designed to execute is called as an **instruction set**. This instruction set carries out the following types of operations:

- Data transfer
- Arithmetic operations
- Logical operations
- Control flow
- Input/output

c) Word Size

The number of bits that can be processed by a processor in a single instruction is called its word size. **Word size** determines the amount of RAM that can be accessed by a microprocessor at one time and the total number of pins on the microprocessor. Total number of input and output pins in turn determines the architecture of the microprocessor.

Speed Measurement
Hertz – abbreviated as Hz is the standard unit of measurement used for measuring frequency.
Since frequency is measured in cycles per second, one hertz equals one cycle per second.
Hertz is commonly used to measure wave frequencies, such as sound waves, light waves, and radio waves. For example, the average human ear can detect sound waves between 20 and 20,000 Hz. Sound waves close to 20 Hz have a low pitch and are called "bass" frequencies. Sound waves above 5,000 Hz have a high pitch and are called "treble" frequencies.

While hertz can be used to measure wave frequencies, it is also used to measure the speed of computer processors. For example, each CPU is rated at a specific clock speed. This number indicates how many instruction cycles the processor can perform in every second. Since modern processors can perform millions or even billions of instructions per second, clock speeds are typically measured in megahertz or gigahertz.

70. What is Vector processor classification?

Ans: According to from where the operands are retrieved in a vector processor, pipe lined vector computers are classified into two architectural configurations:

1. **Memory to memory architecture –**

In memory to memory architecture, source operands, intermediate and final results are retrieved (read) directly from the main memory. For memory to memory vector instructions, the information of the base address, the offset, the increment, and the vector length must be specified in order to enable streams of data transfers between the main memory and pipelines. The processors like *TI-ASC, CDC STAR-100, and Cyber-205* have vector instructions in memory to memory formats. The main points about memory to memory architecture are:

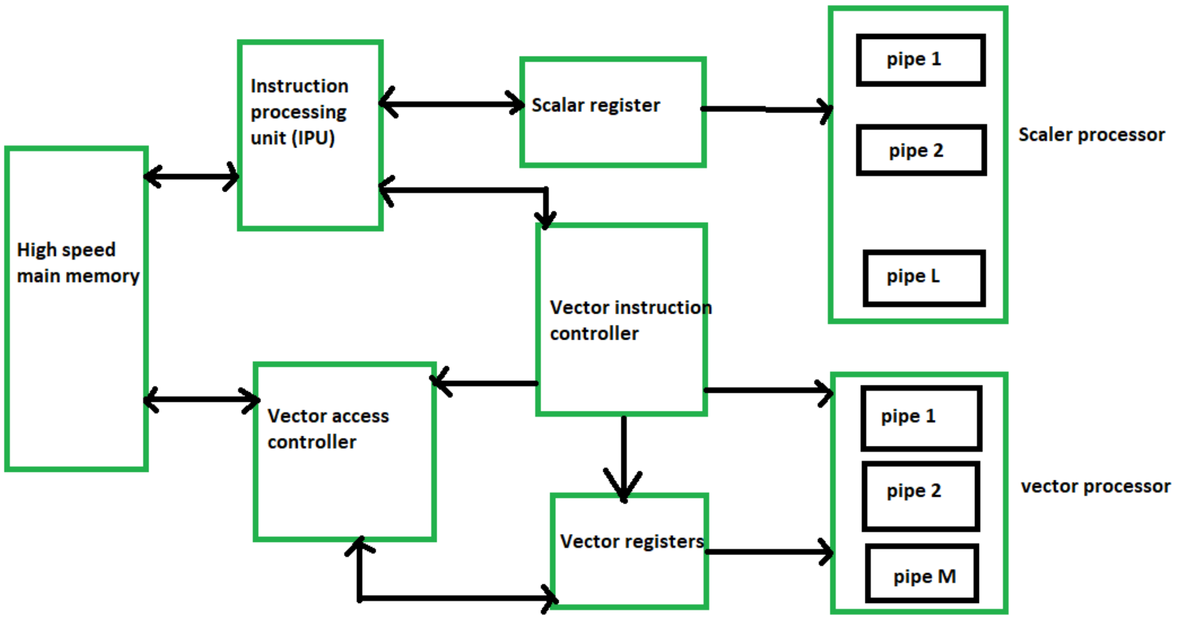
- There is no limitation of size
- Speed is comparatively slow in this architecture

2. **Register to register architecture –**

In register to register architecture, operands and results are retrieved indirectly from the main memory through the use of large number of vector registers or scalar registers. The processors like *Cray-1 and the Fujitsu VP-200* use vector instructions in register to register formats. The main points about register to register architecture are:

- Register to register architecture has limited size.
- Speed is very high as compared to the memory to memory architecture.
- The hardware cost is high in this architecture.

A block diagram of a modern multiple pipeline vector computer is shown below:



A typical pipe lined vector processor.

71. List out the difference between Desktop and Tablets.

Ans:

1. **Desktop :** Desktop is a physical computer unit that consists of a monitor, CPU, key-board and a mouse. It is a graphical user work space on a software operating system. It is designed for regular use at one location. It requires main power supply so that it can not be portable.
2. **Tablet :** Tablet or Tablet Computer is a device generally operated with a mobile operating system. It has the touchscreen display and there is a rechargeable battery inbuilt in it. It is basically a thin and flat device. It does not have the physical key-board with it. The lasting time of battery in tablet is more as compared to battery life of the laptop. It is light weighted and is portable easily.

Difference between Desktop and Tablet :

DESKTOP	TABLET
It is a computer device that needs external devices to be fully functional.	It is a touch screen display computer device which generally operates on mobile operating systems.
It is large in size.	While it is small in size.
It has a physical key-board externally connected with it.	While it does not have the physical key-board, it has onscreen key-board.
It has a separate mouse connected with it.	While in tablet all the work is done with touch screen.
It is heavy-weighted in comparison of tablet.	While it is light-weighted comparatively.
It is not portable.	While it is easily portable.
It runs on main power supply.	While it runs on battery.
It has very powerful processor.	While it is operated with mobile operating system.
The repairing of desktops is easy work as compared to tablets.	While the repairing of tablets is little complex.
It has wide rage of screen size.	While the range of screen size in tablets is limited.
Components of desktop can be easily removed.	Components of tablets are not removable.
It does not have slot for sim-cards.	Some tablets may have slot for sim-cards.
It is more expensive than tablet.	While it is less expensive comparatively.
It has more features in	It has less features comparatively.

DESKTOP	TABLET
comparison of tablet.	
It may have CD or DVD player inbuilt in CPU.	While it does not have such functionality.

72. What is the characteristic of RISC?

Ans: **Characteristic of RISC –**

1. Simpler instruction, hence simple instruction decoding.
2. Instruction come under size of one word.
3. Instruction take single clock cycle to get executed.
4. More number of general purpose register.
5. Simple Addressing Modes.
6. Less Data types.
7. Pipeling can be achieved.

73. What are the characteristic of CISC ?

Ans: **Characteristic of CISC –**

1. Complex instruction, hence complex instruction decoding.
2. Instruction are larger than one word size.
3. Instruction may take more than single clock cycle to get executed.
4. Less number of general purpose register as operation get performed in memory itself.
5. Complex Addressing Modes.
6. More Data types.

74. What are the characteristics of Parallel processing?

Ans: A parallel processing system has the following characteristics:

- Each processor in a system can perform tasks concurrently.
- Tasks may need to be synchronized.
- Nodes usually share resources, such as data, disks, and other devices.

75. Write the advantages of Parallel Computing over Serial Computing.

Ans: Before taking a toll on Parallel Computing, first let's take a look at the background of computations of a computer software and why it failed for the modern era.

Computer software were written conventionally for serial computing. This meant that to solve a problem, an algorithm divides the problem into smaller instructions. These discrete instructions are then executed on Central Processing Unit of a computer one by one. Only after one instruction is finished, next one starts.

Real life example of this would be people standing in a queue waiting for movie ticket and there is only cashier.Cashier is giving ticket one by one to the persons. Complexity of this situation increases when there are 2 queues and only one cashier.

So, in short Serial Computing is following:

1. In this, a problem statement is broken into discrete instructions.
2. Then the instructions are executed one by one.
3. Only one instruction is executed at any moment of time.

Look at point 3. This was causing a huge problem in computing industry as only one instruction was getting executed at any moment of time. This was a huge waste of hardware resources as only one part of the hardware will be running for a particular instruction and of time. As problem statements were getting heavier and bulkier, so does the amount of time in execution of those statements. Example of processors are Pentium 3 and Pentium 4.

Now let's come back to our real life problem. We could definitely say that complexity will decrease when there are 2 queues and 2 cashier giving tickets to 2 persons simultaneously. This is an example of Parallel Computing.

Parallel Computing –

It is the use of multiple processing elements simultaneously for solving any problem. Problems are broken down into instructions and are solved concurrently as each resource which has been applied to work is working at the same time.

Advantages of Parallel Computing over Serial Computing are as follows:

1. It saves time and money as many resources working together will reduce the time and cut potential costs.
2. It can be impractical to solve larger problems on Serial Computing.
3. It can take advantage of non-local resources when the local resources are finite.
4. Serial Computing 'wastes' the potential computing power, thus Parallel Computing makes better work of hardware.

Types of Parallelism:

1. **Bit-level parallelism:** It is the form of parallel computing which is based on the increasing processor's size. It reduces the number of instructions that the system must execute in order to perform a task on large-sized data.
   Example: Consider a scenario where an 8-bit processor must compute the sum of two 16-bit integers. It must first sum up the 8 lower-order bits, then add the 8 higher-order bits, thus requiring two instructions to perform the operation. A 16-bit processor can perform the operation with just one instruction.
2. **Instruction-level parallelism:** A processor can only address less than one instruction for each clock cycle phase. These instructions can be re-ordered and grouped which are later on executed concurrently without affecting the result of the program. This is called instruction-level parallelism.
3. **Task Parallelism:** Task parallelism employs the decomposition of a task into subtasks and then allocating each of the subtasks for execution. The processors perform execution of sub tasks concurrently.

Why parallel computing?

- The whole real world runs in dynamic nature i.e. many things happen at a certain time but at different places concurrently. This data is extensively huge to manage.
- Real world data needs more dynamic simulation and modeling, and for achieving the same, parallel computing is the key.
- Parallel computing provides concurrency and saves time and money.

- Complex, large datasets, and their management can be organized only and only using parallel computing's approach.
- Ensures the effective utilization of the resources. The hardware is guaranteed to be used effectively whereas in serial computation only some part of hardware was used and the rest rendered idle.
- Also, it is impractical to implement real-time systems using serial computing.

Applications of Parallel Computing:

- Data bases and Data mining.
- Real time simulation of systems.
- Science and Engineering.
- Advanced graphics, augmented reality and virtual reality.

Limitations of Parallel Computing:

- It addresses such as communication and synchronization between multiple sub-tasks and processes which is difficult to achieve.
- The algorithms must be managed in such a way that they can be handled in the parallel mechanism.
- The algorithms or program must have low coupling and high cohesion. But it's difficult to create such programs.
- More technically skilled and expert programmers can code a parallelism based program well.

Future of Parallel Computing: The computational graph has undergone a great transition from serial computing to parallel computing. Tech giant such as Intel has already taken a step towards parallel computing by employing multicore processors. Parallel computation will revolutionize the way computers work in the future, for the better good. With all the world connecting to each other even more than before, Parallel Computing does a better role in helping us stay that way. With faster networks, distributed systems, and multi-processor computers, it becomes even more necessary.

76. Explain Pipelining and its types.

Ans: To improve the performance of a CPU we have two options:
1) Improve the hardware by introducing faster circuits.
2) Arrange the hardware such that more than one operation can be performed at the same time.

Since, there is a limit on the speed of hardware and the cost of faster circuits is quite high, we have to adopt the 2nd option.

Pipelining : Pipelining is a process of arrangement of hardware elements of the CPU such that its overall performance is increased. Simultaneous execution of more than one instruction takes place in a pipelined processor.

Let us see a real life example that works on the concept of pipelined operation. Consider a water bottle packaging plant. Let there be 3 stages that a bottle should pass through, Inserting the bottle(**I**), Filling water in the bottle(**F**), and Sealing the bottle(**S**). Let us consider these stages as stage 1, stage 2 and stage 3 respectively. Let each stage take 1 minute to complete its operation.
Now, in a non pipelined operation, a bottle is first inserted in the plant, after 1 minute it is moved to stage 2 where water is filled. Now, in stage 1 nothing is happening. Similarly, when the bottle moves to stage 3, both stage 1 and stage 2 are idle. But in pipelined operation, when the bottle is in stage 2, another bottle can be loaded at stage 1. Similarly, when the bottle is in stage 3, there can be one bottle each in stage 1 and stage 2. So, after each minute, we get a new bottle at the end of stage 3. Hence, the average time taken to manufacture 1 bottle is :

Without pipelining = 9/3 minutes = 3m

| | | I F S | | |

| | | | | | I F S (9 minutes)

With pipelining = 5/3 minutes = 1.67m

I F S | |

| I F S |

| | I F S (5 minutes)

Thus, pipelined operation increases the efficiency of a system.

Design of a basic pipeline

- In a pipelined processor, a pipeline has two ends, the input end and the output end. Between these ends, there are multiple stages/segments such that output of one stage is connected to input of next stage and each stage performs a specific operation.

- Interface registers are used to hold the intermediate output between two stages. These interface registers are also called latch or buffer.

- All the stages in the pipeline along with the interface registers are controlled by a common clock.

77. What are important factors in the Design of a basic pipeline?

Ans: There are many factors that affect the pipe-wall-thickness requirement, which include:

- The maximum and working pressures
- Maximum and working temperatures
- Chemical properties of the fluid
- The fluid velocity
- The pipe material and grade
- The safety factor or code design application

78. Discuss the importance of Pipeline Stages in Pipelining.

Ans: **Pipeline Stages**

RISC processor has 5 stage instruction pipeline to execute all the instructions in the RISC instruction set. Following are the 5 stages of RISC pipeline with their respective operations:

- **Stage 1 (Instruction Fetch)**

  In this stage the CPU reads instructions from the address in the memory whose value is present in the program counter.

- **Stage 2 (Instruction Decode)**

  In this stage, instruction is decoded and the register file is accessed to get the values from the registers used in the instruction.

- **Stage 3 (Instruction Execute)**

In this stage, ALU operations are performed.

- **Stage 4 (Memory Access)**

  In this stage, memory operands are read and written from/to the memory that is present in the instruction.

- **Stage 5 (Write Back)**

  In this stage, computed/fetched value is written back to the register present in the instructions.

79. How you will calculate Performance of a pipelined processor?

Ans: Consider a ‘k’ segment pipeline with clock cycle time as ‘Tp’. Let there be ‘n’ tasks to be completed in the pipelined processor. Now, the first instruction is going to take ‘k’ cycles to come out of the pipeline but the other ‘n – 1’ instructions will take only ‘1’ cycle each, i.e, a total of ‘n – 1’ cycles. So, time taken to execute ‘n’ instructions in a pipelined processor:

$ET_{pipeline} = k + n – 1$ cycles
$= (k + n – 1)\ Tp$

In the same case, for a non-pipelined processor, execution time of ‘n’ instructions will be:

$ET_{non\text{-}pipeline} = n * k * Tp$

So, speedup (S) of the pipelined processor over non-pipelined processor, when ‘n’ tasks are executed on the same processor is:

S = Performance of pipelined processor /
Performance of Non-pipelined processor

As the performance of a processor is inversely proportional to the execution time, we have,

$S = ET_{non\text{-}pipeline} / ET_{pipeline}$
$\Rightarrow S = [n * k * Tp] / [(k + n – 1) * Tp]$
$S = [n * k] / [k + n – 1]$

When the number of tasks ‘n’ are significantly larger than k, that is, n >> k

$S = n * k / n$
$S = k$

where ‘k’ are the number of stages in the pipeline.

Also, **Efficiency** = Given speed up / Max speed up = S / S_{max}
We know that, Smax = k
So, **Efficiency** = S / k

80. Write all the dependencies possible in a pipelined processor.

Ans: **Dependencies in a pipelined processor**

There are mainly three types of dependencies possible in a pipelined processor. These are :
1) Structural Dependency
2) Control Dependency
3) Data Dependency

These dependencies may introduce stalls in the pipeline.

Stall : A stall is a cycle in the pipeline without new input.

Structural dependency

This dependency arises due to the resource conflict in the pipeline. A resource conflict is a situation when more than one instruction tries to access the same resource in the same cycle. A resource can be a register, memory, or ALU.

Example:

INSTRUCTION / CYCLE	**1**	**2**	**3**	**4**	**5**
I_1	IF(Mem)	ID	EX	Mem	
I_2		IF(Mem)	ID	EX	
I_3			IF(Mem)	ID	EX
I_4				IF(Mem)	ID

In the above scenario, in cycle 4, instructions I_1 and I_4 are trying to access same resource (Memory) which introduces a resource conflict.
To avoid this problem, we have to keep the instruction on wait until the required resource (memory in our case) becomes available. This wait will introduce stalls in the pipeline as shown below:

CYCLE	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
I_1	IF(Mem)	ID	EX	Mem	WB			
I_2		IF(Mem)	ID	EX	Mem	WB		
I_3			IF(Mem)	ID	EX	Mem	WB	
I_4				–	–	–	IF(Mem)	

Solution for structural dependency
To minimize structural dependency stalls in the pipeline, we use a hardware mechanism called Renaming.
Renaming : According to renaming, we divide the memory into two independent modules used to store the instruction and data separately called Code memory(CM) and Data memory(DM) respectively. CM will contain all the instructions and DM will contain all the operands that are required for the instructions.

INSTRUCTION/ CYCLE	1	2	3	4	5	6	7
I_1	IF(CM)	ID	EX	DM	WB		
I_2		IF(CM)	ID	EX	DM	WB	
I_3			IF(CM)	ID	EX	DM	WB
I_4				IF(CM)	ID	EX	DM
I_5					IF(CM)	ID	EX
I_6						IF(CM)	ID
I_7							IF(CM)

Control Dependency (Branch Hazards)
This type of dependency occurs during the transfer of control instructions such as BRANCH, CALL, JMP, etc. On many instruction architectures, the processor will not know the target address of these instructions when it needs to insert the new instruction into the pipeline. Due to this, unwanted instructions are fed to the pipeline.
Consider the following sequence of instructions in the program:
100: I_1
101: I_2 (JMP 250)
102: I_3
.
.
250: BI_1
Expected output: I_1 -> I_2 -> BI_1
NOTE: Generally, the target address of the JMP instruction is known after ID stage only.

INSTRUCTION/ CYCLE	1	2	3	4	5	6

INSTRUCTION/ CYCLE	1	2	3	4	5	6
I_1	IF	ID	EX	MEM	WB	
I_2		IF	ID (PC:250)	EX	Mem	WB
I_3			IF	ID	EX	Mem
BI_1				IF	ID	EX

Output Sequence: I_1 -> I_2 -> I_3 -> BI_1

So, the output sequence is not equal to the expected output, that means the pipeline is not implemented correctly.
To correct the above problem we need to stop the Instruction fetch until we get target address of branch instruction. This can be implemented by introducing delay slot until we get the target address.

INSTRUCTION/ CYCLE	1	2	3	4	5	6
I_1	IF	ID	EX	MEM	WB	
I_2		IF	ID (PC:250)	EX	Mem	WB
Delay	–	–	–	–	–	–
BI_1				IF	ID	EX

Output Sequence: I_1 -> I_2 -> Delay (Stall) -> BI_1
As the delay slot performs no operation, this output sequence is equal to the expected output sequence. But this slot introduces stall in the pipeline.
Solution for Control dependency Branch Prediction is the method through which stalls due to control dependency can be eliminated. In this at 1st stage prediction is done about which branch will be taken.For branch prediction Branch penalty is zero.
Branch penalty : The number of stalls introduced during the branch operations in the pipelined processor is known as branch penalty.
NOTE : As we see that the target address is available after the ID stage, so the number of stalls introduced in the pipeline is 1. Suppose, the branch target address would have been present after the ALU stage, there would have been 2 stalls. Generally, if the target address is present after the k^{th} stage, then there will be (k – 1) stalls in the pipeline.
Total number of stalls introduced in the pipeline due to branch instructions = **Branch frequency * Branch Penalty**

Data Dependency (Data Hazard)
Let us consider an ADD instruction S, such that
S : ADD R1, R2, R3
Addresses read by S = I(S) = {R2, R3}
Addresses written by S = O(S) = {R1}

Now, we say that instruction S2 depends in instruction S1, when

$$[I(S1) \cap O(S2)] \cup [O(S1) \cap I(S2)] \cup [O(S1) \cap O(S2)] \neq \phi$$

This condition is called Bernstein condition.
Three cases exist:

- Flow (data) dependence: O(S1) ∩ I (S2), S1 → S2 and S1 writes after something read by S2
- Anti-dependence: I(S1) ∩ O(S2), S1 → S2 and S1 reads something before S2 overwrites it
- Output dependence: O(S1) ∩ O(S2), S1 → S2 and both write the same memory location.

Example: Let there be two instructions I1 and I2 such that:
I1 : ADD R1, R2, R3
I2 : SUB R4, R1, R2
When the above instructions are executed in a pipelined processor, then data dependency condition will occur, which means that I_2 tries to read the data before I_1 writes it, therefore, I_2 incorrectly gets the old value from I_1.

INSTRUCTION / CYCLE	**1**	**2**	**3**	**4**
I_1	IF	ID	EX	DM
I_2		IF	ID(Old value)	EX

To minimize data dependency stalls in the pipeline, **operand forwarding** is used.
Operand Forwarding : In operand forwarding, we use the interface registers present between the stages to hold intermediate output so that dependent instruction can access new value from the interface register directly.
Considering the same example:
I1 : ADD R1, R2, R3
I2 : SUB R4, R1, R2

INSTRUCTION / CYCLE	**1**	**2**	**3**	**4**
I_1	IF	ID	EX	DM
I_2		IF	ID	EX

Data Hazards
Data hazards occur when instructions that exhibit data dependence, modify data in different stages of a pipeline. Hazard cause delays in the pipeline. There are mainly three types of data hazards:

1) RAW (Read after Write) [Flow/True data dependency]
2) WAR (Write after Read) [Anti-Data dependency]
3) WAW (Write after Write) [Output data dependency]
Let there be two instructions I and J, such that J follow I. Then,

- RAW hazard occurs when instruction J tries to read data before instruction I writes it. Eg:
  I: R2 <- R1 + R3
  J: R4 <- R2 + R3
- WAR hazard occurs when instruction J tries to write data before instruction I reads it. Eg:
  I: R2 <- R1 + R3
  J: R3 <- R4 + R5
- WAW hazard occurs when instruction J tries to write output before instruction I writes it. Eg:
  I: R2 <- R1 + R3
  J: R2 <- R4 + R5

WAR and WAW hazards occur during the out-of-order execution of the instructions.

81. What is Data Hazards?

Ans: **Data hazards** occur when instructions that exhibit **data** dependence modify **data** in different stages of a **pipeline**. Ignoring potential **data hazards** can result in race conditions (also termed race **hazards**). There are three situations in which a **data hazard** can occur: read after write (RAW), a true dependency.

82. What is the Vector Instruction Format in Vector Processors?

Ans: Different **Instruction formats** are used by different vector processors. Vector instructions are generally specified by some fields. The main fields that are used in **vector instruction set** are given below:

1. **Operations Code (Opcode) –**
   The operation code must be specified to select the functional unit or to reconfigure a multi-functional unit to perform the specified operation dictated by this field. Usually, microcode control is used to set up the required resources.
   For example:
   Opcode – 0001 mnemonic – ADD operation – add the content of memory to the content of accumulator
   Opcode – 0010 mnemonic – SUB operation – subtract the content of memory to the content of accumulator
   Opcode – 1111 mnemonic – HLT operation – stop processing
2. **Base addresses –**
   For a memory reference instruction, the base addresses are needed for both source operands and result vectors. The designated vector registers must be specified in the instruction, if the operands and results are located in the vector register file, i.e., collection of registers.
   For example:
   ```
   ADD R1, R2
   ```
   Here, R1 and R2 are the addresses of the register.

3. **Offset (or Displacement) –**
   This field is required to get the effective memory address of operand vector. The address offset relative

to the base address should be specified. Using the base address and the offset (positive or negative), the **effective address** is calculated.

4. **Address Increment –**
   The address increment between the scalar elements of vector operand must be specified. Some computers, i.e., the increment is always 1. Some other computers, like **TI-ASC**, can have a variable increment, which offers higher flexibility in application.
   For example:
   ```
   R1 <- 400
   ```
   Auto incr-R1 is incremented the value of R1 by 1.
   ```
   R1 = 399
   ```
5. **Vector length –**
   The vector length (positive integer) is needed to determine the termination of a vector instruction.

83. What are the types of Parallelism?

Ans: **Types of Parallelism:**

1. **Bit-level parallelism:** It is the form of parallel computing which is based on the increasing processor's size. It reduces the number of instructions that the system must execute in order to perform a task on large-sized data.
   Example: Consider a scenario where an 8-bit processor must compute the sum of two 16-bit integers. It must first sum up the 8 lower-order bits, then add the 8 higher-order bits, thus requiring two instructions to perform the operation. A 16-bit processor can perform the operation with just one instruction.
2. **Instruction-level parallelism:** A processor can only address less than one instruction for each clock cycle phase. These instructions can be re-ordered and grouped which are later on executed concurrently without affecting the result of the program. This is called instruction-level parallelism.
3. **Task Parallelism:** Task parallelism employs the decomposition of a task into subtasks and then allocating each of the subtasks for execution. The processors perform execution of sub tasks concurrently.

Objective					
84 Ans: c)	**______ have been developed specifically for pipelined systems.**	**a) Utility software**	**b) Speed up utilities**	**c) Optimizing compilers**	**d) None of the mentioned**
85 Ans: b)	**The pipelining process is also called as ______**	**a) Superscalar operation**	**b) Assembly line operation**	**c) Von Neumann cycle**	**d) None of the mentioned**
86 Ans: b)	**The fetch and execution cycles are interleaved with the help of**	**a) Modification in processor architecture**	**b) Special unit**	**b) Clock**	**d) Control unit**

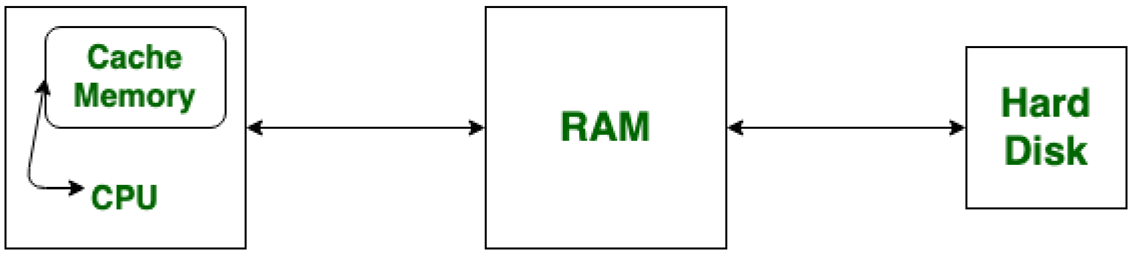
	________				
87 Ans: a)	**Each stage in pipelining should be completed within ___________ cycle.**	**a) 1**	**b) 2**	**c) 3**	**d) 4**
88. Ans: b)	**In pipelining the task which requires the least time is performed first.**	**a) True**	**b) False**	**c) None**	**d) Equal**
89. Ans: c)	**To increase the speed of memory access in pipelining, we make use of _______**	**a) Special memory locations**	**b) Special purpose registers**	**c) Cache**	**d) Buffers**
90. Ans: d)	**The periods of time when the unit is idle is called as _____**	**a) Stalls**	**b) Bubbles**	**c) Hazards**	**d) Both Stalls and Bubbles**
91. Ans: a)	**The contention for the usage of a hardware device is called ______**	**a) Structural hazard**	**b) Stalk**	**c) Deadlock**	**d) None of the mentioned**
92. Ans: a)	**The situation wherein the data of operands are not available is called ______**	**a) Data hazard**	**b) Stock**	**c) Deadlock**	**d) Structural hazard**
93 Ans: d)	**The simplest scheme to handle branches is to**	**a) Flush the pipeline**	**b) Freezing the pipeline**	**c) Depth of the pipeline**	**d) Both a and b**

Module 4

94. Difference between Random Access Memory (RAM) and Content Addressable Memory (CAM)
Ans: RAM:
Random Access Memory (RAM) is used to read and write. It is the part of primary memory and used in order to store running applications (programs) and program's data for performing operation. It is mainly of two types: Dynamic RAM (or DRAM) and Static RAM (or SRAM).

CAM:
Content Addressable Memory (CAM) is also known as Associative Memory, in which the user supplies data word and associative memory searches its entire memory and if the data word is found, It returns the list of addresses where that data word was located.

95. Difference between Virtual memory and Cache memory.
Ans:
Cache memory increases the accessing speed of CPU. It is not a technique but a memory unit i.e a storage device. In cache memory, recently used data is copied. Whenever the program is ready to be executed, it is fetched from main memory and then copied to the cache memory. But, if its copy is already present in the cache memory then the program is directly executed.

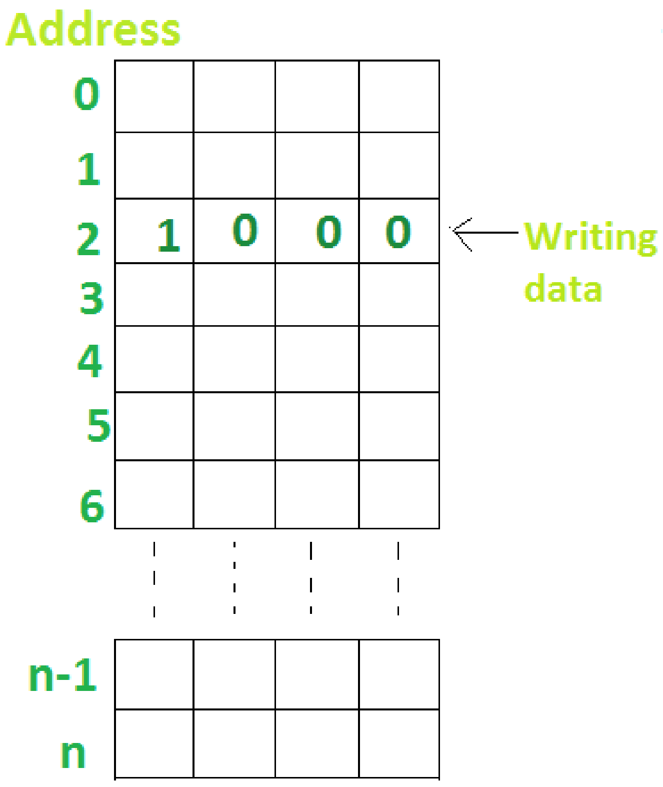


Virtual Memory increases the capacity of main memory. Virtual memory is not a storage unit, its a technique. In virtual memory, even such programs which have a larger size than the main memory are allowed to be executed.
Difference between Virtual memory and Cache memory:

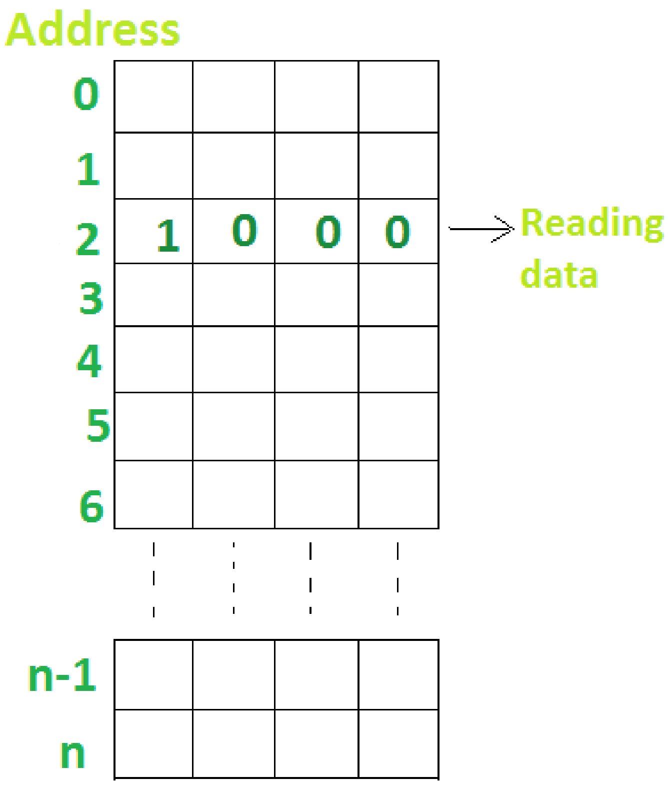
S.NO	VIRTUAL MEMORY	CACHE MEMORY
1.	Virtual memory increases the capacity of main memory.	While cache memory increase the accessing speed of CPU.
2.	Virtual memory is not a memory unit, its a technique.	Cache memory is exactly a memory unit.
3.	The size of virtual memory is greater than the cache memory.	While the size of cache memory is less than the virtual memory.
4.	Operating System manages the Virtual memory.	On the other hand hardware manages the cache memory.

5.	In virtual memory, The program with size larger than the main memory are executed.	While in cache memory, recently used data is copied into.
6.	In virtual memory, mapping frameworks is needed for mapping virtual address to physical address.	While in cache memory, no such mapping frameworks is needed.

96. Define memory and memory units.
 Ans: Memories are made up of registers. Each register in the memory is one storage location. Storage location is also called as memory location. Memory locations are identified using **Address**. The total number of bit a memory can store is its **capacity**.
A storage element is called a **Cell**. Each register is made up of storage element in which one bit of data is stored. The data in a memory are stored and retrieved by the process called **writing** and **reading** respectively.

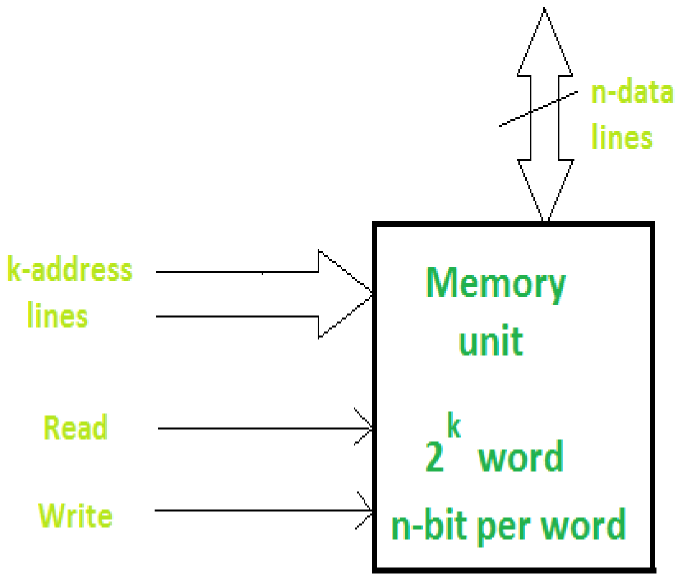


A)Write operation



B)Read Operation

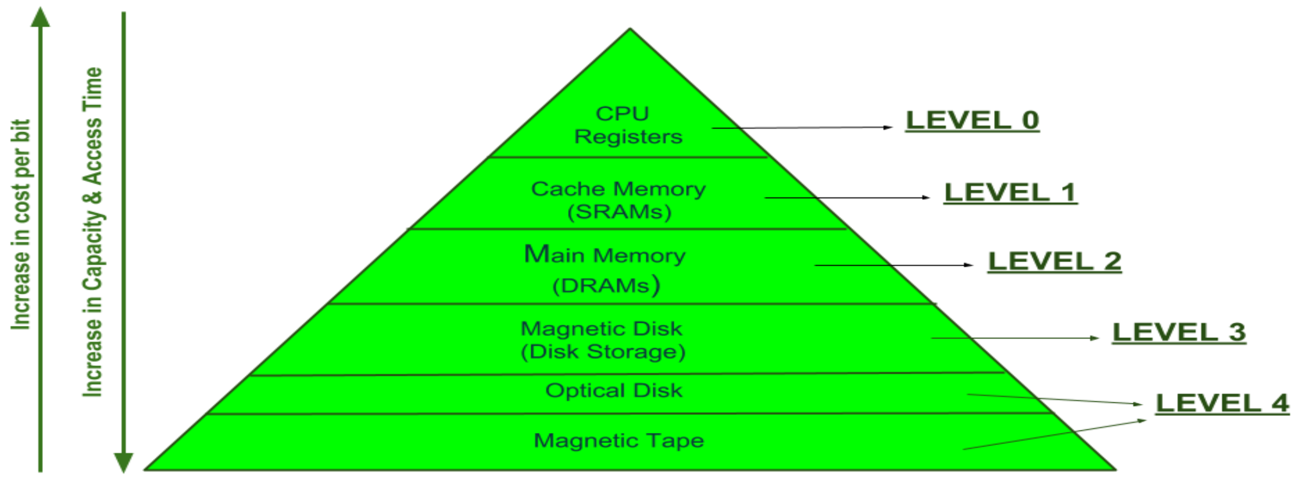
A **word** is a group of bits where a memory unit stores binary information. A word with group of 8 bits is calleda **byte**.
A memory unit consists of data lines, address selection lines, and control lines that specify the direction of transfer. The block diagram of a memory unit is shown below:



Data lines provide the information to be stored in memory. The control inputs specify the direction transfer. The k-address lines specify the word chosen.
When there are k address lines, 2^k memory word can be accessed.

97. Define Secondary Memory.
Ans: **Secondary memory** is where programs and data are kept on a long-term basis.
Common **secondary** storage devices are the hard disk and optical disks. The hard disk has enormous storage capacity compared to main **memory**. The hard disk is usually contained inside the case of a computer.

98. Define a concept of Memory hierarchy design.
Ans: In the Computer System Design, Memory Hierarchy is an enhancement to organize the memory such that it can minimize the access time. The Memory Hierarchy was developed based on a program behavior known as locality of references.The figure below clearly demonstrates the different levels of memory hierarchy :



MEMORY HIERARCHY DESIGN

This Memory Hierarchy Design is divided into 2 main types:

1. **External Memory or Secondary Memory** – Comprising of Magnetic Disk, Optical Disk, Magnetic Tape i.e. peripheral storage devices which are accessible by the processor via I/O Module.
2. **Internal Memory or Primary Memory** – Comprising of Main Memory, Cache Memory & CPU registers. This is directly accessible by the processor.

   We can infer the following characteristics of Memory Hierarchy Design from above figure:

1. **Capacity:**
   It is the global volume of information the memory can store. As we move from top to bottom in the Hierarchy, the capacity increases.
2. **Access Time:**
   It is the time interval between the read/write request and the availability of the data. As we move from top to bottom in the Hierarchy, the access time increases.
3. **Performance:**
   Earlier when the computer system was designed without Memory Hierarchy design, the speed gap increases between the CPU registers and Main Memory due to large difference in access time. This results in lower performance of the system and thus, enhancement was required. This enhancement was made in the form of Memory Hierarchy Design because of which the performance of the system increases. One of the most significant ways to increase system performance is minimizing how far down the memory hierarchy one has to go to manipulate data.
4. **Cost per bit:**
   As we move from bottom to top in the Hierarchy, the cost per bit increases i.e. Internal Memory is costlier than External Memory.

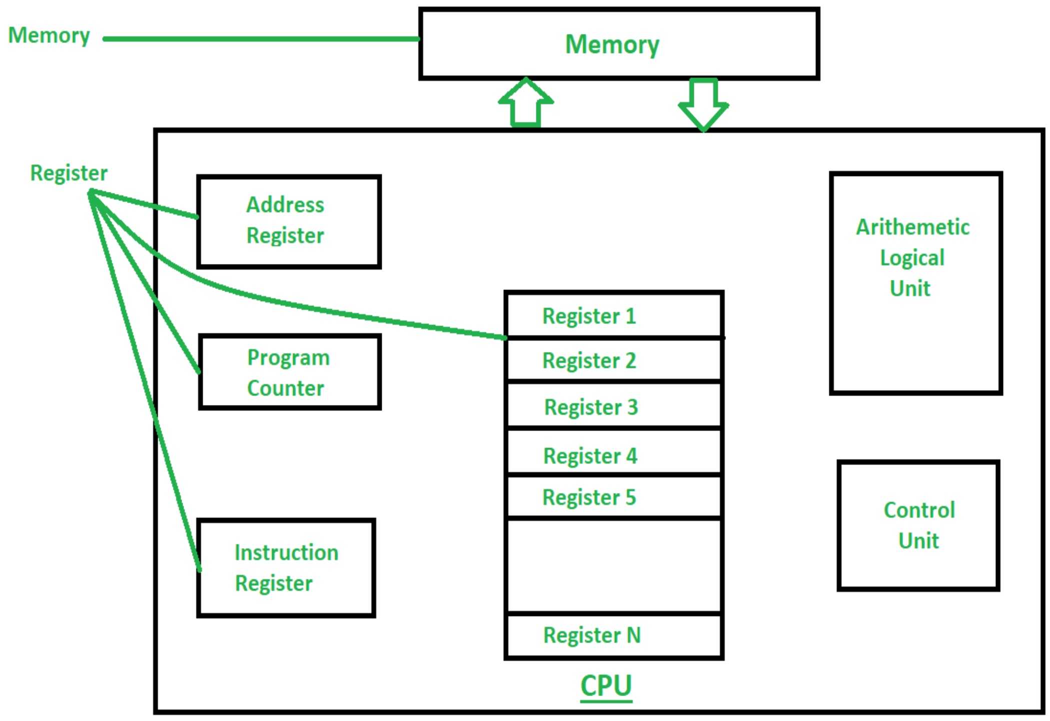
99. Define Memory Organisation.
Ans: **Memory Organization** in Computer Architecture. A **memory** unit is the collection of storage units or devices together. The **memory** unit stores the binary information in the form of bits. Generally, **memory**/storage is classified into 2 categories: Volatile **Memory**: This loses its data, when power is switched off.

100. What are the difference between Byte Addressable Memory and Word Addressable Memory?
Ans:

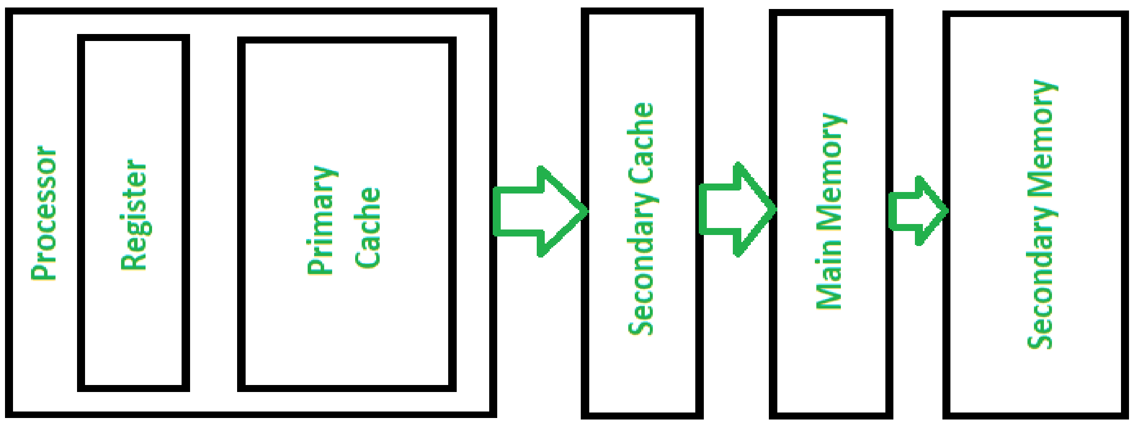
BYTE ADDRESSABLE MEMORY	WORD ADDRESSABLE MEMORY
When the ***data space in the cell = 8 bits*** then the corresponding ***address space*** is called as **Byte Address**.	When the ***data space in the cell = word length of CPU*** then the corresponding ***address space*** is called as **Word Address**.
Based on this data storage i.e. *Bytewise storage*, the memory chip configuration is named as **Byte Addressable Memory**.	Based on this data storage i.e. *Wordwise storage*, the memory chip configuration is named as **Word Addressable Memory**.
For eg. : **64K X 8** chip has 16 bit Address and **cell size = 8 bits (1 Byte)** which means that in this chip, data is stored **byte by byte**.	For eg. : For a 16-bit CPU, **64K X 16** chip has 16 bit Address & **cell size = 16 bits (Word Length of CPU)** which means that in this chip, data is stored **word by word**.

NOTE :
i) The most important point to be noted is that in case of either of Byte Address or Word Address, *the address size* can be any number of bits (depends on the number of cells in the chip) but the *cell size* differs in each case.
ii)The default memory configuration in the Computer design is Byte Addressable .

102. Difference between Register and Memory
Ans:
1. Register :
Registers are the smallest data holding elements that are built into the processor itself. These are the memory locations that are directly accessible by the processor. It may hold an instruction, a storage address or any kind of data such as a bit sequence or individual characters. For example, an instruction may specify that the contents of two defined registers be multiplied together and then placed in a specific register.
Example: Accumulator register, Program counter, Instruction register, Address register, etc.



2. Memory :
Memory is a hardware device used to store computer programs, instructions and data. The memory that is internal to the processor is a primary memory (RAM), and the memory that is external to the processor is a secondary memory (Hard Drive). Memory can also be categorized on the basis of volatile and non-volatile memory. Volatile memory is memory that loses its contents when the computer or hardware device loses power. RAM (Random Access Memory) is an example of volatile memory. Non-volatile memory is the memory that keeps its contents even if power gets lost. EPROM is an example of non-volatile memory.
Example : RAM, EPROM etc.



Difference between Register and Memory :

S.NO.	REGISTER	MEMORY
1.	Registers hold the operands or instruction that CPU is currently processing.	Memory holds the instructions and the data that the currently executing program in CPU requires.
2.	Register holds the small amount of data around 32-bits to 64-bits.	Memory of the computer can range from some GB to TB.
3.	CPU can operate on register contents at the rate of more than one operation in one clock cycle.	CPU accesses memory at the slower rate than register.
4.	Types are Accumulator register, Program counter, Instruction register, Address register, etc.	Type of memory are RAM,etc.
5.	Registers can be control i.e. you can store and retrieve information from them.	Memory is almost not controllable.

S.NO.	REGISTER	MEMORY
6.	Registers are faster than memory.	RAM is much slower than registers.

103. Difference between Memory and Storage.
Ans: **Memory**
The term memory refers to the component within your computer that allows you to access data that is stored for a short term. You may recognize this component as DRAM, or dynamic random-access memory. Your computer performs many operations by accessing data stored in its short-term memory. Some examples of such operations include editing a document, loading applications and browsing the Internet. The speed and performance of your system depends on the amount of memory that is installed on your computer. If you have a desk and a filing cabinet, the desk represents the memory of your computer. Items you will need to use soon are kept in your desk for easy access. However, not much can be stored in a desk due to its size limitations.
Storage
Whereas memory refers to the location of short-term data, storage is the component of your computer that allows you to store and access data on a long-term basis. Usually, storage comes in the form of a solid-state drive or a hard drive. Storage allows you to access and store your applications, operating system and files for an indefinite period of time.
While the desk represents the computer's memory, the filing cabinet represents the storage of your computer. Items that must be kept yet won't necessarily be accessed soon are stored in the filing cabinet. Due to the size of the filing cabinet, many things can be stored.
An important distinction between memory and storage is that the former clears when the computer is turned off. On the other hand, storage remains intact no matter how many times you shut off your computer. Therefore, in the desk and filing cabinet analogy, any files that are left on your desk when you leave the office will be thrown away. Everything in your filing cabinet will remain.

104. What is CPU register?
Ans: The term CPU Register is often used to refer only to the group of registers that can be directly indexed for input or output of an instruction, as defined by the instruction set. More properly, these are called the "*architected registers*". For instance, the x86 instruction set defines a set of eight 32-bit registers, but a CPU that implements the X86 instruction set will contain many more hardware registers than just these eight.

There are several other classes of registers:
(a) **Accumulator**: It is most frequently used register used to store data taken from memory. Its number varies from microprocessor to microprocessor.
(b) **General Purpose registers**: General purpose registers are used to store data and intermediate results during program execution. Its contents can be accessed through assembly programming.
(c) **Special purpose Registers**: Users do not access these registers. These are used by computer system at the time of program execution. Some types of special purpose registers are given below:

- **Memory Address Register (MAR)**: It stores address of data or instructions to be fetched from memory.
- **Memory Buffer Register (MBR)**: It stores instruction and data received from the memory and sent from the memory.

- **Instruction Register (IR):** Instructions are stored in instruction register. When one instruction is completed, next instruction is fetched in memory for processing.
- **Program Counter (PC):** It counts instructions.
  The instruction cycle is completed into two phases: (a) **Fetch Cycle** and (b) **Execute Cycle**. There are two parts in instruction- opcode and operand. In fetch cycle **opcode** of instruction is fetched into CPU. The opcode, at first, is reached to Data Register (DR), then to Instruction Register (IR). Decoder accesses the opcode and it decodes opcode and type of operation is declared to CPU and execution cycle is started.

105. What is Cache memory?
Ans: Cache memory is an extremely fast memory type that acts as a buffer between RAM and the CPU. It holds frequently requested data and instructions so that they are immediately available to the CPU when needed. Cache memory is used to reduce the average time to access data from the Main memory.

106. Difference between Primary and Secondary Memory.
Ans: The difference between Primary memory and Secondary memory:

SR.NO.	PRIMARY MEMORY	SECONDARY MEMORY
1.	Primary memory is temporary.	Secondary memory is permanent.
2.	Primary memory is directly accessible by Processor/CPU.	Secondary memory is not directly accessible by the CPU.
3.	Nature of Parts of Primary memory varies, RAM- volatile in nature. ROM- Non-volatile.	It’s always Non-volatile in nature.
4.	Primary memory devices are more expensive than secondary storage devices.	Secondary memory devices are less expensive when compared to primary memory devices.
5.	The memory devices used for primary memory are semiconductor memories.	The secondary memory devices are magnetic and optical memories.
6.	Primary memory is also known as	Secondary memory is also known as External memory or

SR.NO.	PRIMARY MEMORY	SECONDARY MEMORY
	Main memory or Internal memory.	Auxiliary memory.
7.	Examples: RAM, ROM, Cache memory, PROM, EPROM, Registers, etc.	Examples: Hard Disk, Floppy Disk, Magnetic Tapes, etc.

107. Difference between Memory based and Register based Addressing Modes.
Ans: **Addressing modes** are the operations field specifies the operations which need to be performed. The operation must be executed on some data which is already stored in computer registers or in the memory. The way of choosing operands during program execution is dependent on addressing modes of the instruction. "The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually referenced. "Basically how we are interpreting the operand which is given in the instruction is known as addressing mode. Addressing mode very much depend on the type of CPU organisation. There are three types of CPU organisation:

1. Single Accumulator organisation
2. General register organisation
3. Stack organisation

Addressing modes is used for one or both of the purpose. These can also be said as the **advantages** of using addressing mode:

1. To give programming versatility to the user by providing such facilities as pointers to memory, counter for loop control, indexing of data, and program relocation.
2. To reduce the number of bits in the addressing field of the instruction.

There are numbers of addressing modes available and it depends on the architecture and CPU organisation which of the addressing modes can be applied.

MEMORY BASED ADDRESSING MODES	REGISTER BASED ADDRESSING MODES
The operand is present in memory and its address is given in the instruction itself. This addressing mode is taking proper advantage of memory address, e.g., Direct addressing mode	An operand will be given in one of the register and register number will be provided in the instruction.With the register number present in instruction, operand is fetched, e.g., Register mode
The memory address specified in instruction may give the address where the effective address is stored in	The register contains the address of the operand. The effective address can be derived from the content of the register specified

MEMORY BASED ADDRESSING MODES	REGISTER BASED ADDRESSING MODES
the memory. In this case effective memory address is present in the memory address which is specified in the instruction, e.g., Indirect Addressing Mode	in the instruction. The content of the register might not be the effective address. This mode takes full advantage of registers, e.g., Register indirect mode
The content of base register is added to the address part of the instruction to obtain the effective address. A base register is assumed to hold a base address and the address field of the instruction gives displacement relative to the base address, e.g., Base Register Addressing Mode	If we are having a table of data and our program needs to access all the values one by one we need something which decrements the program counter/or any register which has base address. Though in this case register is basically decreased, it is register based addressing mode, e.g., In Auto decrements mode
The content of the index register is added to the address part that is given in the instruction to obtain the effective address. Index Mode is used to access an array whose elements are in successive memory locations, e.g., Indexed Addressing Mode	If we are having a table of data and our program needs to access all the values one by one we need something which increment the program counter/or any register which has base address, e.g., Auto increment mode
The content of program counter is added to the address part of the instruction in order to obtain the effective address. The address part of the instruction in this case is usually a signed number which can be either positive or negative, e.g., Relative addressing mode	Instructions generally used for initializing registers to a constant value is register based addressing mode,and this technique is very useful approach, e.g., Immediate mode.

Memory based addressing modes are mostly rely on Memory address and content present at some memory location. Register based addressing modes are mostly rely on Registers and content present at some register either it is data or some memory address.

108. Define addressing modes.

Ans: Addressing Modes– The term addressing modes refers to the way in which the operand of an instruction is specified. The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually executed.

Addressing modes for 8086 instructions are divided into two categories:

1) Addressing modes for data

2) Addressing modes for branch

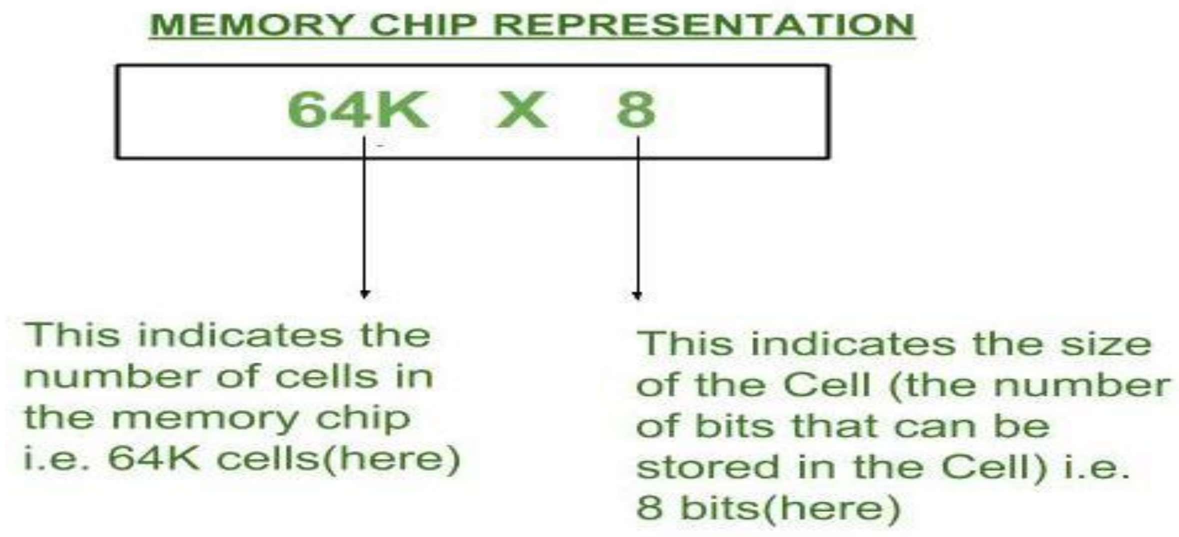
The 8086 memory addressing modes provide flexible access to memory, allowing you to easily access variables, arrays, records, pointers, and other complex data types. The key to good assembly language programming is the proper use of memory addressing modes.

109. Difference between Volatile Memory and Non-Volatile Memory.
Ans:
Volatile Memory:
It is that the quite hardware that stores information quickly. it's additionally referred as temporary memory. The information within the volatile memory is hold on solely till the ability is provided to the system, once the system is turned off the information gift within the volatile memory is deleted mechanically. RAM (Random Access Memory) and Cache Memory are the common example of the volatile memory. It's quite quick and economical in nature and may be accessed apace.
Non-Volatile Memory:
It is the type of memory in which data or information remains keep within the memory albeit power is completed. ROM (Read Only Memory) is the most common example of non-volatile memory. it's not that a lot of economical and quick in nature as compare to volatile memory however stores information for the longer amount. Non-volatile memory is slow concerning accessing. All such information that must be hold on for good or for a extended amount is hold on in non-volatile memory. Non-volatile memory has a huge impact on a system's storage capacity.
Let's see that the difference between volatile and non-volatile memory:

S.NO	VOLATILE MEMORY	NON-VOLATILE MEMORY
1.	Volatile memory is the type of memory in which data isn't keep in memory as before long as power is gone.	Non-volatile memory is the type of memory in which data or information remains keep within the memory albeit power is completed.
2.	Volatile memory is not a permanent memory.	Non-volatile memory is a permanent memory.
3.	It is faster than non-volatile memory.	It is slow than volatile memory.
4.	**RAM** is the example of volatile memory.	**ROM** is the example of non-volatile memory.
5.	In volatile memory, data can	In non-volatile memory, data can

S.NO	VOLATILE MEMORY	NON-VOLATILE MEMORY
	be easily transferred in comparison of non-volatile memory.	not be easily transferred in comparison of volatile memory.
6.	Volatile memory can read and write.	Non-volatile memory can’t write, it only read.
7.	Volatile memory has less storage.	Non-volatile memory has more storage than volatile memory.
8.	In volatile memory, the program’s data are stored which are currently in process by the CPU.	In non-volatile memory, any kind of data which has to be saved permanently are stored.
9.	Volatile memory is more costly per unit size.	Non-volatile memory is less costly per unit size.
10.	Volatile memory has a huge impact on the system’s performance.	Non-volatile memory has a huge impact on a system’s storage capacity.
11.	In volatile memory, processor has direct access to data.	In non-volatile memory, processor has no direct access to data.
12.	Volatile memory chips are generally lies on the memory slot.	Non-volatile memory chips are contained on the motherboard.

110. Difference between Byte Addressable Memory and Word Addressable Memory.
Ans: Memory is a storage component in the Computer used to store application programs. The Memory Chip is divided into equal parts called as *“CELLS”*. Each Cell is uniquely identified by a binary number called as *“ADDRESS”*. For example, the Memory Chip configuration is represented as **’64 K x 8′** as shown in the figure below.


MEMORY CHIP REPRESENTATION
64K X 8
This indicates the number of cells in the memory chip i.e. 64K cells(here)
This indicates the size of the Cell (the number of bits that can be stored in the Cell) i.e. 8 bits(here)

The following information can be obtained from the memory chip representation shown above:

1. Data Space in the Chip = 64K X 8

2. Data Space in the Cell = 8 bits

3. Address Space in the Chip = =16 bits

Now we can clearly state the difference between Byte Addressable Memory & Word Addressable Memory.

BYTE ADDRESSABLE MEMORY	**WORD ADDRESSABLE MEMORY**
When the ***data space in the cell = 8 bits*** then the corresponding ***address space*** is called as **Byte Address**.	When the ***data space in the cell = word length of CPU*** then the corresponding ***address space*** is called as **Word Address**.
Based on this data storage i.e. *Bytewise storage*, the memory chip configuration is named as **Byte Addressable Memory**.	Based on this data storage i.e. *Wordwise storage*, the memory chip configuration is named as **Word Addressable Memory**.
For eg. : **64K X 8** chip has 16 bit Address and **cell size = 8 bits (1 Byte)** which means that in this chip, data is stored **byte by byte**.	For eg. : For a 16-bit CPU, **64K X 16** chip has 16 bit Address & **cell size = 16 bits (Word Length of CPU)** which means that in this chip, data is stored **word by word**.

NOTE :

i) The most important point to be noted is that in case of either of Byte Address or Word Address, *the address size* can be any number of bits (depends on the number of cells in the chip) but the *cell size* differs in each case.

ii)The default memory configuration in the Computer design is Byte Addressable .

111. What is Cache Performance and how it is calculated?

Ans: **Cache Performance**

- *Average memory access time is a useful measure to evaluate the performance of a memory-hierarchy configuration.*

$$\text{Avg mem access time} = \text{hit time} + \text{miss rate} \times \text{miss penalty}$$

- *It tells us how much penalty the memory system imposes on each access (on average).*

- *It can easily be converted into clock cycles for a particular CPU.*

- *But leaving the penalty in nanoseconds allows two systems with different clock cycles times to be compared to a single memory system.*

Cache Performance

- *There may be different penalties for Instruction and Data accesses.*
    - *In this case, you may have to compute them separately.*

    - *This requires knowledge of the fraction of references that are instructions and the fraction that are data.*
        - *The text gives 75% instruction references to 25% data references.*

    - *We can also compute the write penalty separately from the read penalty.*

    - *This may be necessary for two reasons:*
- *Miss rates are different for each situation.*
- *Miss penalties are different for each situation.*

- *Treating them as a single quantity yields a useful CPU time formula:*

$$\text{CPU time} = \text{IC} \times \left(\text{CPI}_{\text{execution}} + \frac{\text{Memory access}}{\text{Instruction}} \times \text{Miss rate} \times \text{Miss penalty}\right) \times \text{Clock Cycle Time}$$

An Example

- *Compare the performance of a* **64KB unified** *cache with a split cache with* **32KB data** *and* **16KB instruction** .

    - *The miss penalty for either cache is 100 ns, and the CPU clock runs at 200 MHz.*

    - *Don't forget that the cache requires an extra cycle for load and store hits on a unified cache because of the structural conflict.*

    - *Calculate the effect on* **CPI** *rather than the average memory access time.*

    - *Assume miss rates are as follows (Fig. 5.7 in text):*
- *64K Unified cache: 1.35%*
- *16K instruction cache: 0.64%*
- *32K data cache: 4.82%*

    - *Assume a data access occurs once for every 3 instructions, on average.*

An Example

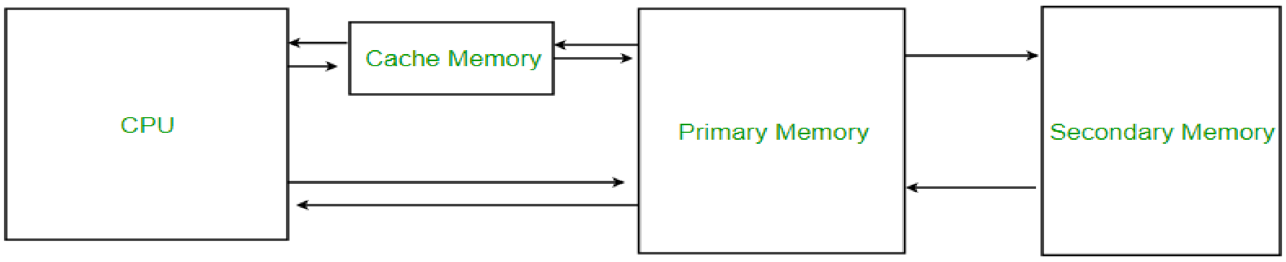
- *The solution is to figure out the penalty to CPI separately for instructions and data.*

- *First, we figure out the miss penalty in terms of clock cycles: 100 ns/5 ns = 20 cycles.*
    - *For the unified cache, the per-instruction penalty is (0 + 1.35% x 20) =* **0.27** *cycles.*
    - *For data accesses, which occur on about 1/3 of all instructions, the penalty is (1 + 1.35% x 20) = 1.27 cycles per access, or* **0.42** *cycles per instruction.*
    - *The total penalty is* **0.69 CPI** *.*

- *In the split cache, the per-instruction penalty is (0 + 0.64% x 20) =* **0.13** *CPI.*
    - *For data accesses, it is (0 + 4.82% x 20) x (1/3) =* **0.32** *CPI.*
    - *The total penalty is* **0.45 CPI** *.*

- *In this case, the split cache performs better because of the lack of a stall on data accesses.*

60. Define Cache Memory in Computer Organization.
Ans: **Cache Memory** is a special very high-speed memory. It is used to speed up and synchronizing with high-speed CPU. Cache memory is costlier than main memory or disk memory but economical than CPU registers. Cache memory is an extremely fast memory type that acts as a buffer between RAM and the CPU. It holds frequently requested data and instructions so that they are immediately available to the CPU when needed.
Cache memory is used to reduce the average time to access data from the Main memory. The cache is a smaller and faster memory which stores copies of the data from frequently used main memory locations. There are various different independent caches in a CPU, which store instructions and data.



61. How cache mapping is done?
Ans: When cache hit occurs,
- The required word is present in the cache memory.
- The required word is delivered to the CPU from the cache memory.

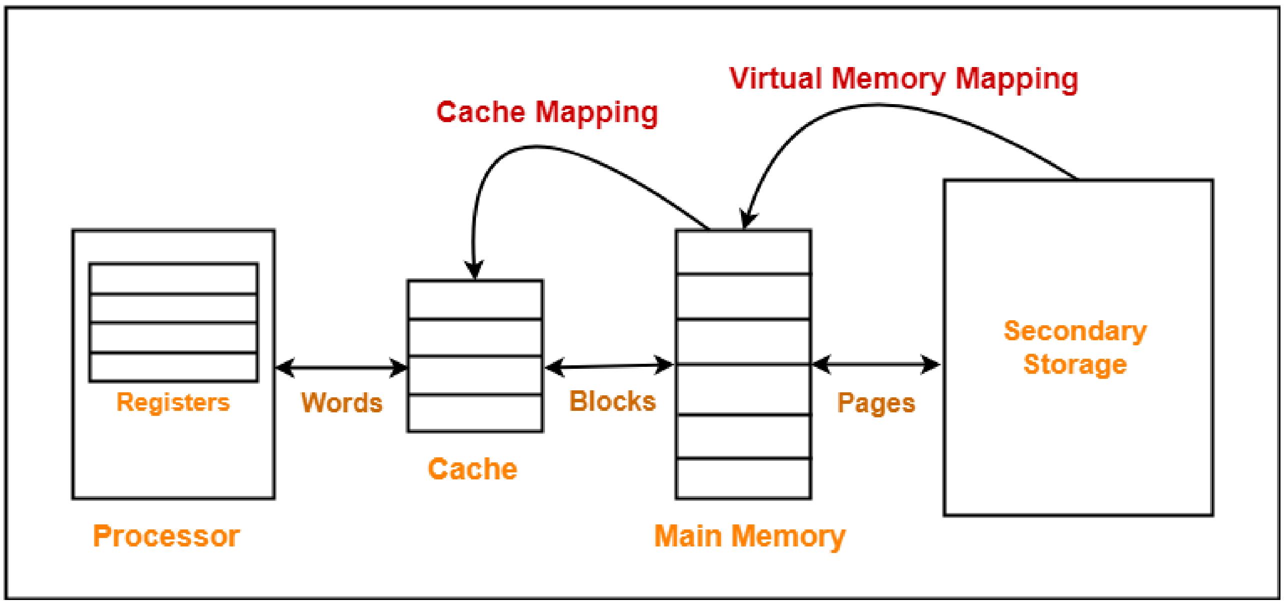
When cache miss occurs,
- The required word is not present in the cache memory.
- The page containing the required word has to be mapped from the main memory.
- This mapping is performed using cache mapping techniques.

In this article, we will discuss different cache mapping techniques.

Cache Mapping-

- Cache mapping defines how a block from the main memory is mapped to the cache memory in case of a cache miss.
OR
- Cache mapping is a technique by which the contents of main memory are brought into the cache memory.

The following diagram illustrates the mapping process-



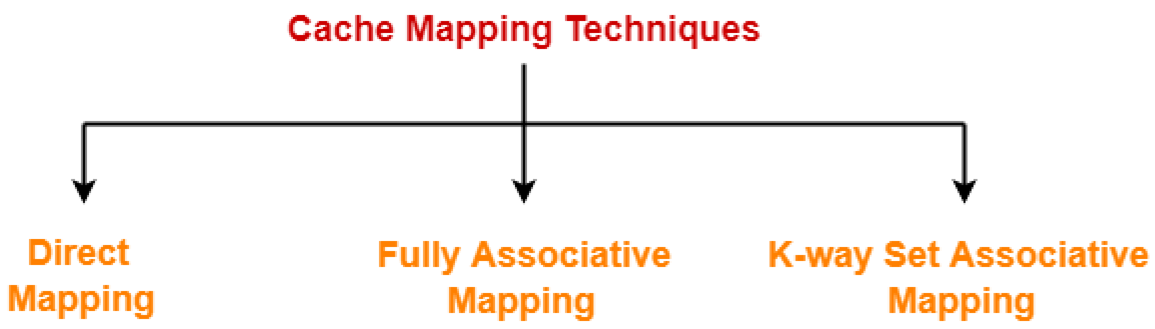
Now, before proceeding further, it is important to note the following points-

NOTES

- Main memory is divided into equal size partitions called as **blocks** or **frames**.
- Cache memory is divided into partitions having same size as that of blocks called as **lines**.
- During cache mapping, block of main memory is simply copied to the cache and the block is not actually brought from the main memory.

Cache Mapping Techniques-

Cache mapping is performed using following three different techniques-



1. Direct Mapping
2. Fully Associative Mapping
3. K-way Set Associative Mapping

1. Direct Mapping-

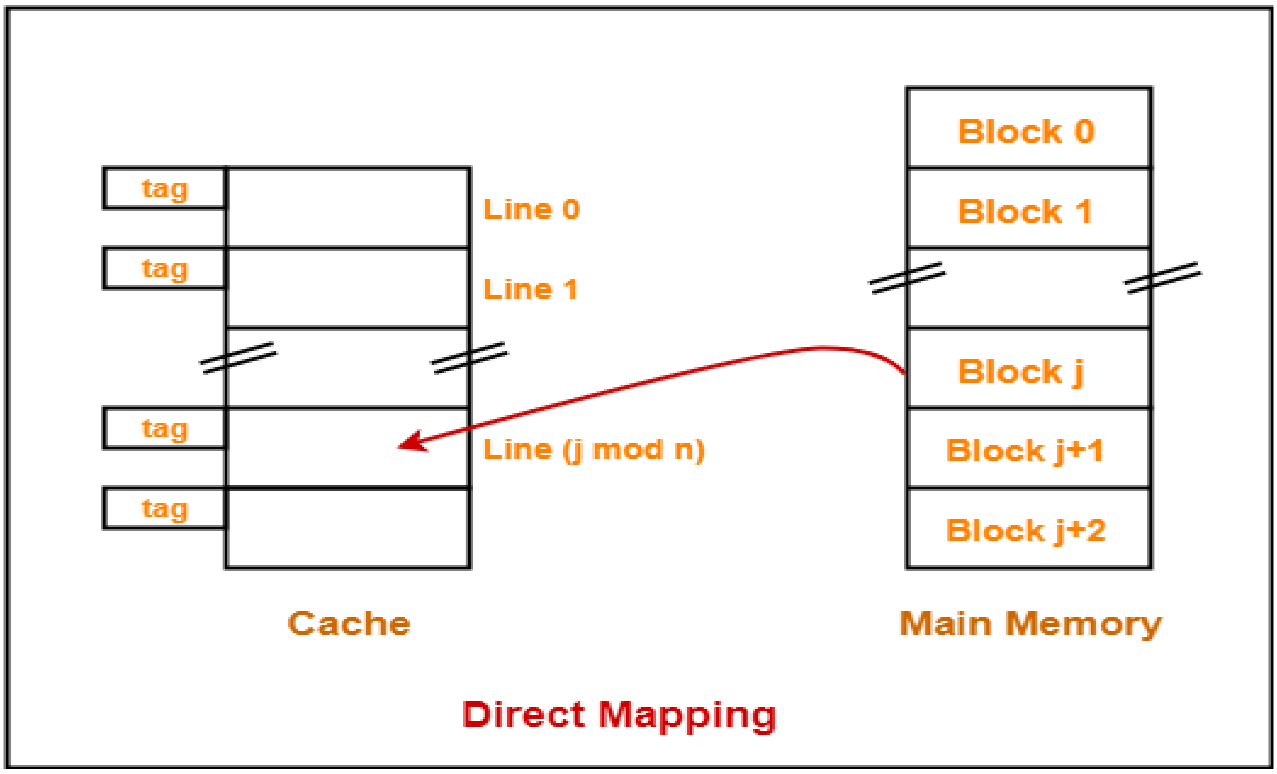
In direct mapping,

- A particular block of main memory can map only to a particular line of the cache.
- The line number of cache to which a particular block can map is given by-

Cache line number
= (Main Memory Block Address) Modulo (Number of lines in Cache)

Example-

- Consider cache memory is divided into 'n' number of lines.
- Then, block 'j' of main memory can map to line number (j mod n) only of the cache.

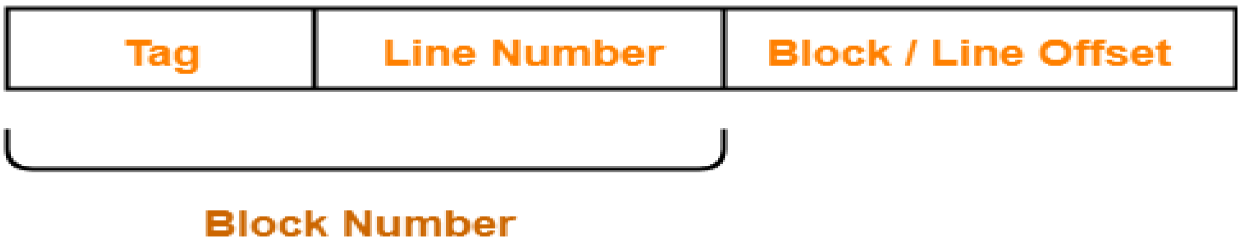


Need of Replacement Algorithm-

In direct mapping,
- There is no need of any replacement algorithm.
- This is because a main memory block can map only to a particular line of the cache.
- Thus, the new incoming block will always replace the existing block (if any) in that particular line.

Division of Physical Address-

In direct mapping, the physical address is divided as-



Division of Physical Address in Direct Mapping

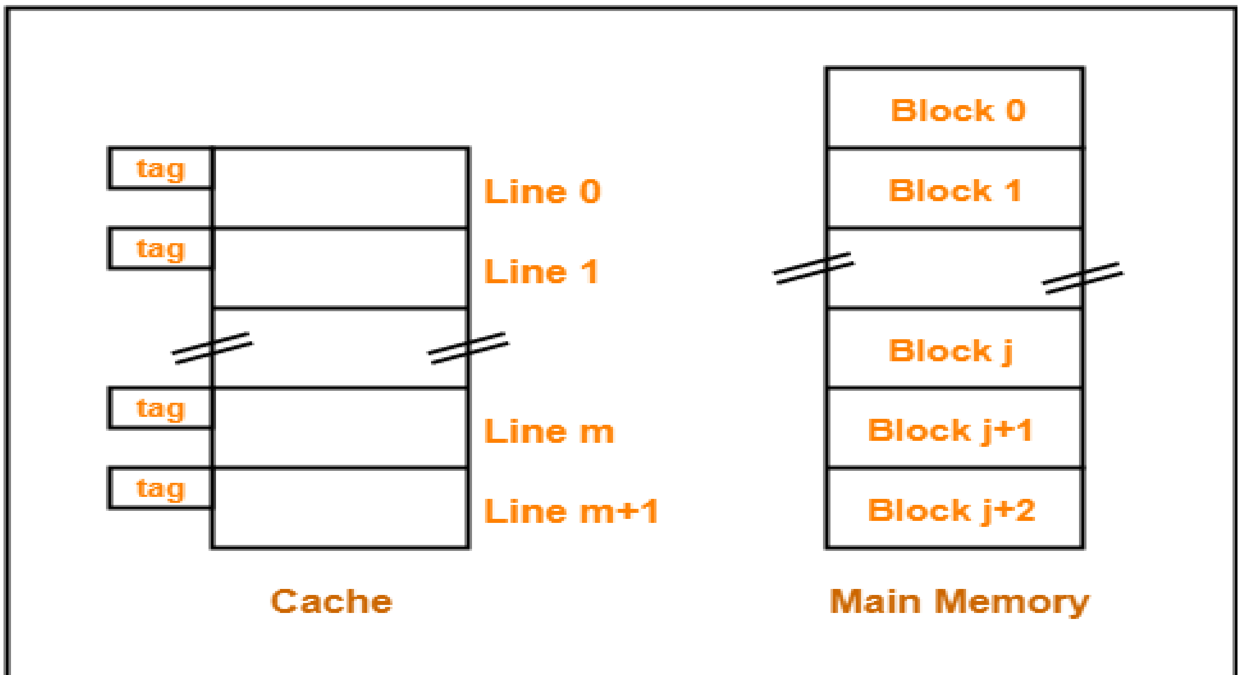
2. Fully Associative Mapping-

In fully associative mapping,

- A block of main memory can map to any line of the cache that is freely available at that moment.
- This makes fully associative mapping more flexible than direct mapping.

Example-

Consider the following scenario-



Fully Associative Mapping

Here,

- All the lines of cache are freely available.
- Thus, any block of main memory can map to any line of the cache.
- Had all the cache lines been occupied, then one of the existing blocks will have to be replaced.

Need of Replacement Algorithm-

In fully associative mapping,

- A replacement algorithm is required.
- Replacement algorithm suggests the block to be replaced if all the cache lines are occupied.
- Thus, replacement algorithm like FCFS Algorithm, LRU Algorithm etc is employed.

Division of Physical Address-

In fully associative mapping, the physical address is divided as-

Block Number / Tag	Block / Line Offset

Division of Physical Address in Fully Associative Mapping

3. K-way Set Associative Mapping-

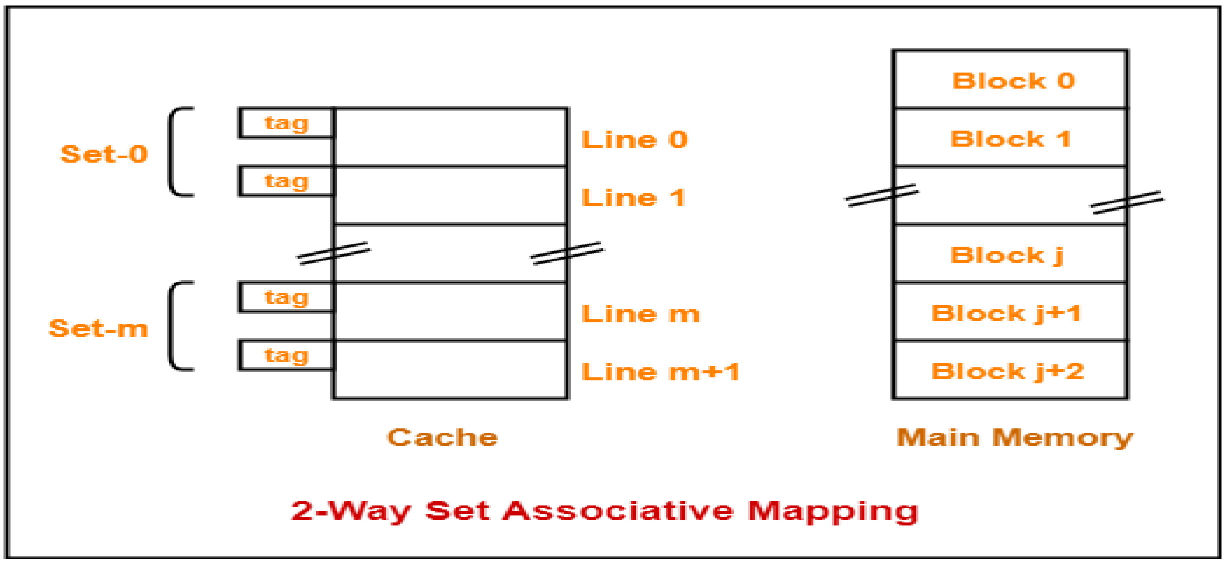
In k-way set associative mapping,

- Cache lines are grouped into sets where each set contains k number of lines.
- A particular block of main memory can map to only one particular set of the cache.
- However, within that set, the memory block can map any cache line that is freely available.
- The set of the cache to which a particular block of the main memory can map is given by-

Cache set number
= (Main Memory Block Address) Modulo (Number of sets in Cache)

Example-

Consider the following example of 2-way set associative mapping-



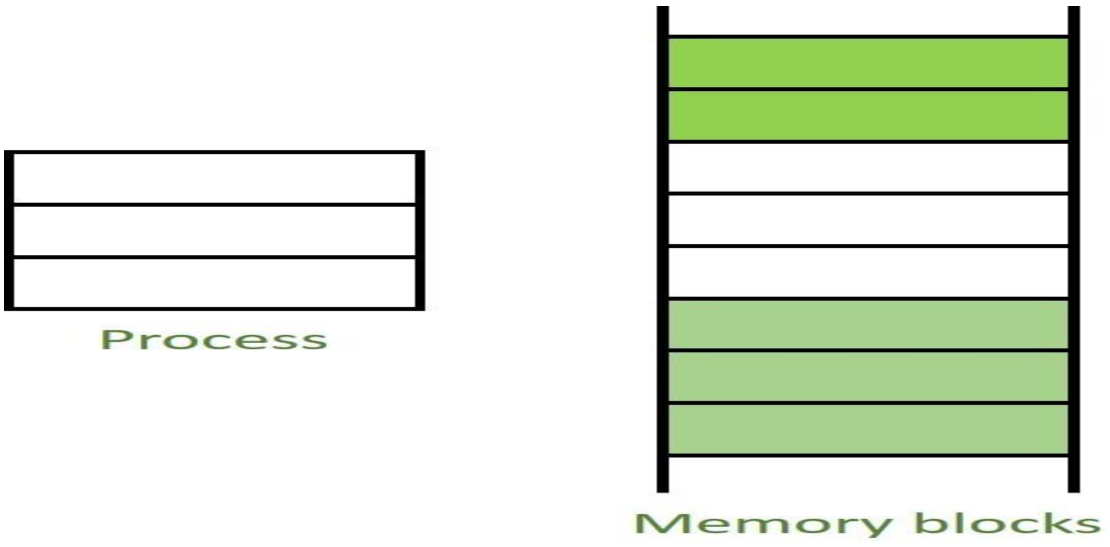
2-Way Set Associative Mapping

Here,

- k = 2 suggests that each set contains two cache lines.
- Since cache contains 6 lines, so number of sets in the cache = 6 / 2 = 3 sets.
- Block ‘j’ of main memory can map to set number (j mod 3) only of the cache.
- Within that set, block ‘j’ can map to any cache line that is freely available at that moment.
- If all the cache lines are occupied, then one of the existing blocks will have to be replaced.

112. Difference between Contiguous and Non-contiguous Memory Allocation
Ans:

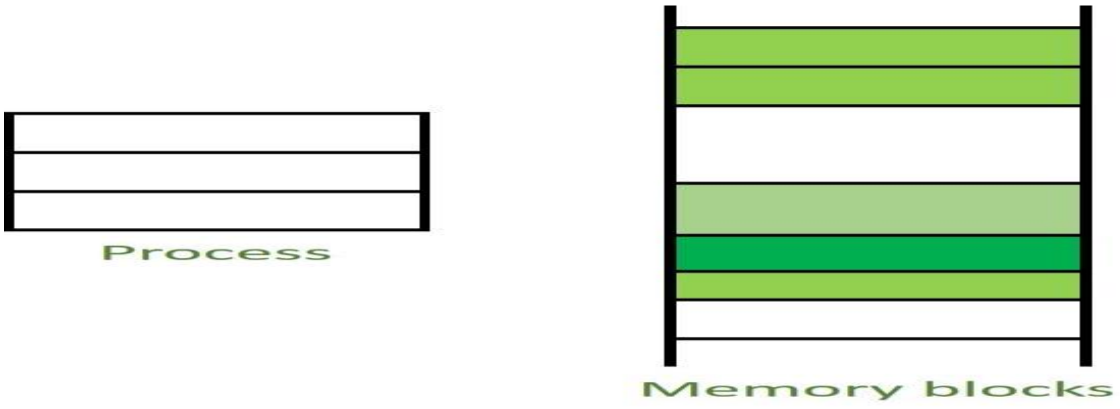
1. Contiguous Memory Allocation :
Contiguous memory allocation is basically a method in which a single contiguous section/part of memory is allocated to a process or file needing it. Because of this all the available memory space resides at the same place together, which means that the freely/unused available memory partitions are not distributed in a random fashion here and there across the whole memory space.



Contiguous Memory Allocation

The main memory is a combination of two main portions- one for the operating system and other for the user program. We can implement/achieve contiguous memory allocation by dividing the memory partitions into fixed size partitions.

3. Non-Contiguous Memory Allocation :
Non-Contiguous memory allocation is basically a method on the contrary to contiguous allocation



Noncontiguous Memory Allocation

method, allocates the memory space present in different locations to the process as per it’s requirements. As all the available memory space is in a distributed pattern so the freely available memory space is also scattered here and there.

This technique of memory allocation helps to reduce the wastage of memory, which eventually gives rise to Internal and external fragmentation.

Difference between Contiguous and Non-contiguous Memory Allocation :

S.NO.	CONTIGUOUS MEMORY ALLOCATION	NON-CONTIGUOUS MEMORY ALLOCATION
1.	Contiguous memory allocation allocates	Non-Contiguous memory allocation allocates separate blocks of memory to a

S.NO.	CONTIGUOUS MEMORY ALLOCATION	NON-CONTIGUOUS MEMORY ALLOCATION
	consecutive blocks of memory to a file/process.	file/process.
2.	Faster in Execution.	Slower in Execution.
3.	It is easier for the OS to control.	It is difficult for the OS to control.
4.	Overhead is minimum as not much address translations are there while executing a process.	More Overheads are there as there are more address translations.
5.	Internal fragmentation occurs in Contiguous memory allocation method.	External fragmentation occurs in Non-Contiguous memory allocation method.
6.	It includes single partition allocation and multi-partition allocation.	It includes paging and segmentation.
7.	Wastage of memory is there.	No memory wastage is there.

113. What do you mean by Cache Block?
Ans: The basic unit for cache storage. May contain multiple bytes/words of data.

114. Explain various types of cache mapping.
Ans:
There are three different types of mapping used for the purpose of cache memory which are as follows: Direct mapping, Associative mapping, and Set-Associative mapping. These are explained below.
Direct Mapping –

The simplest technique, known as direct mapping, maps each block of main memory into only one possible cache line. Or In Direct mapping, assigne each memory block to a specific line in the cache.

If a line is previously taken up by a memory block when a new block needs to be loaded, the old block is trashed. An address space is split into two parts index field and a tag field. The cache is used to store the tag field whereas the rest is stored in the main memory. Direct mapping`s performance is directly proportional to the Hit ratio.

i = j modulo m

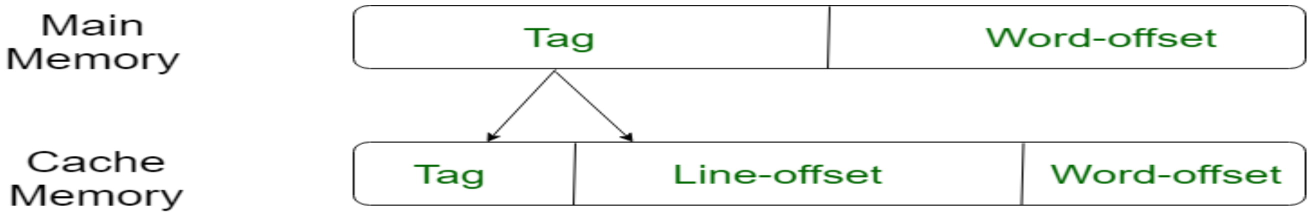
where

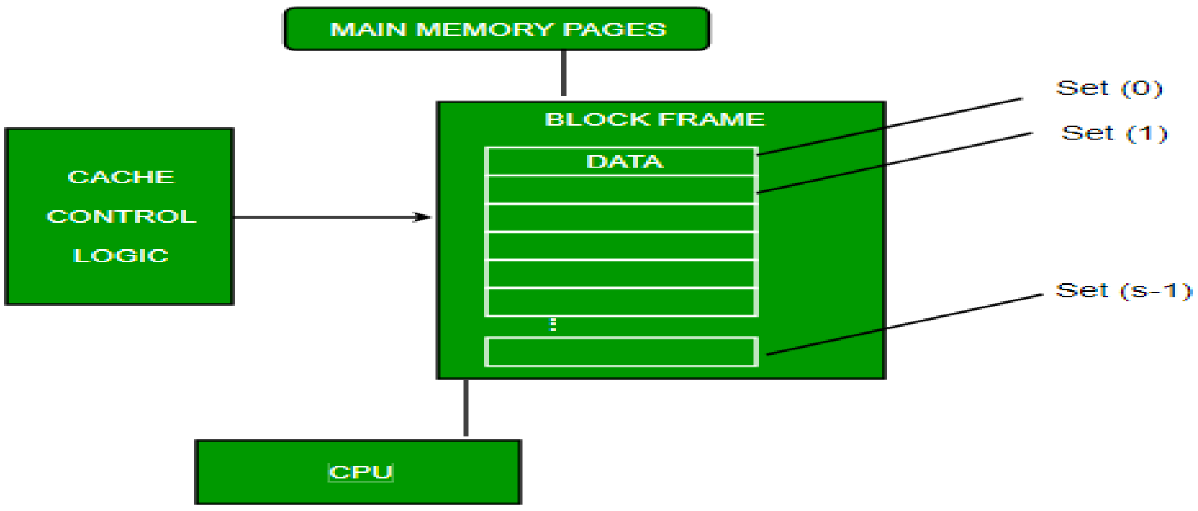
i=cache line number

j= main memory block number

m=number of lines in the cache

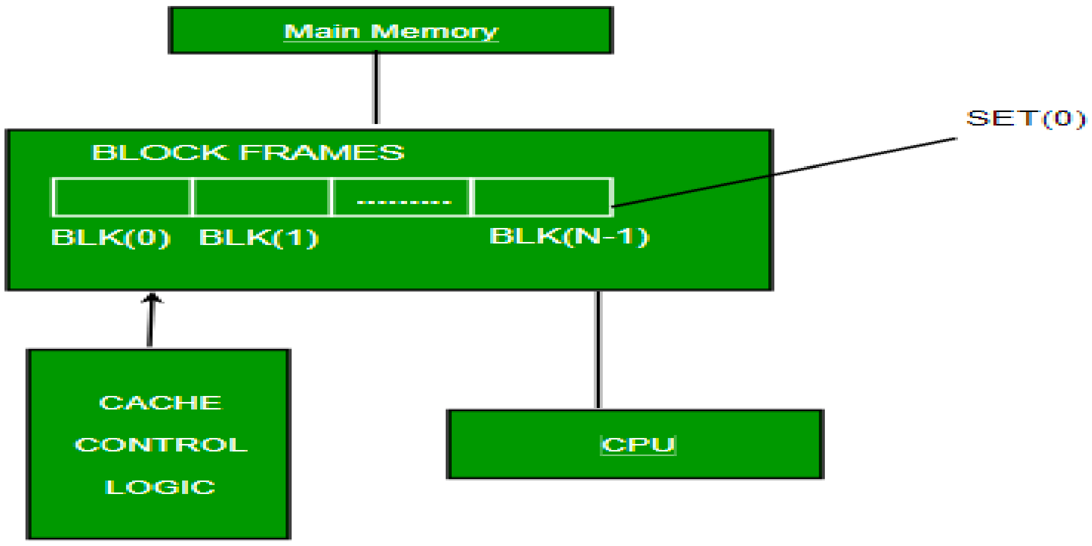
For purposes of cache access, each main memory address can be viewed as consisting of three fields. The least significant w bits identify a unique word or byte within a block of main memory. In most contemporary machines, the address is at the byte level. The remaining s bits specify one of the 2^s blocks of main memory. The cache logic interprets these s bits as a tag of s-r bits (most significant portion) and a line field of r bits. This latter field identifies one of the $m=2^r$ lines of the cache.





Associative Mapping –

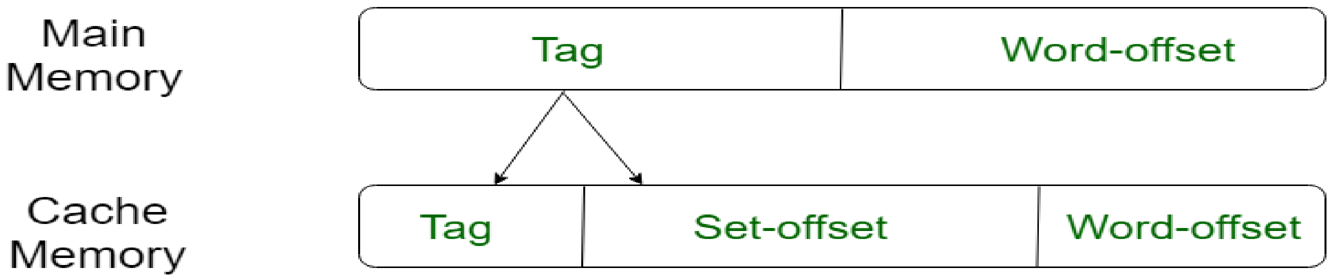
In this type of mapping, the associative memory is used to store content and addresses of the memory word. Any block can go into any line of the cache. This means that the word id bits are used to identify which word in the block is needed, but the tag becomes all of the remaining bits. This enables the placement of any word at any place in the cache memory. It is considered to be the fastest and the most flexible mapping form.

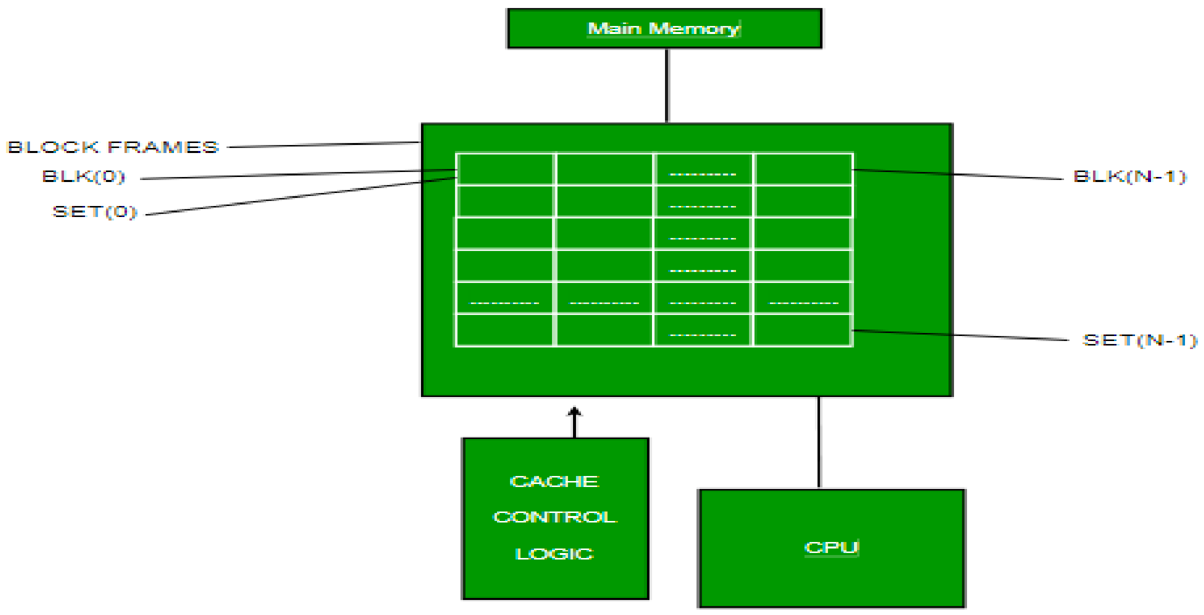


Set-associative Mapping –

This form of mapping is an enhanced form of direct mapping where the drawbacks of direct mapping are removed. Set associative addresses the problem of possible thrashing in the direct mapping method. It does this by saying that instead of having exactly one line that a block can map to in the cache, we will group a few lines together creating a ***set***. Then a block in memory can map to any one of the lines of a specific set..Set-associative mapping allows that each word that is present in the cache can have two or more words in the main memory for the same index address. Set associative cache mapping combines the best of direct and associative cache mapping techniques.
In this case, the cache consists of a number of sets, each of which consists of a number of lines. The relationships are
m = v * k
i= j mod v

where
i=cache set number
j=main memory block number
v=number of sets
m=number of lines in the cache number of sets
k=number of lines in each set



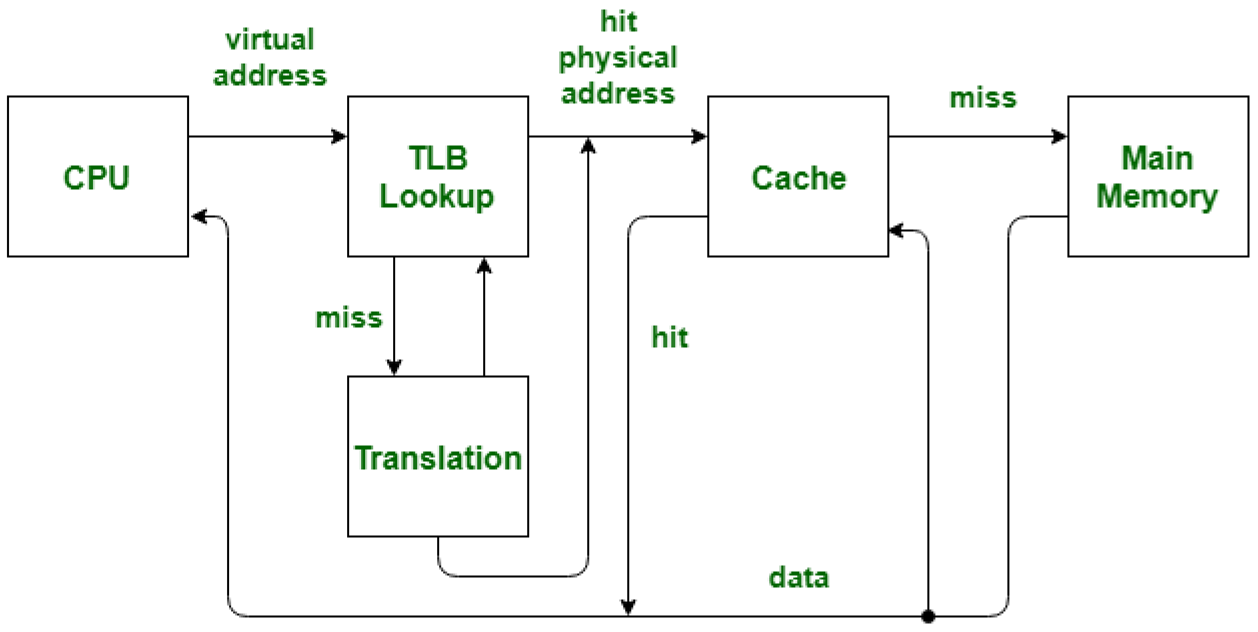


Application of Cache Memory –

1. Usually, the cache memory can store a reasonable number of blocks at any given time, but this number is small compared to the total number of blocks in the main memory.
2. The correspondence between the main memory blocks and those in the cache is specified by a mapping function.

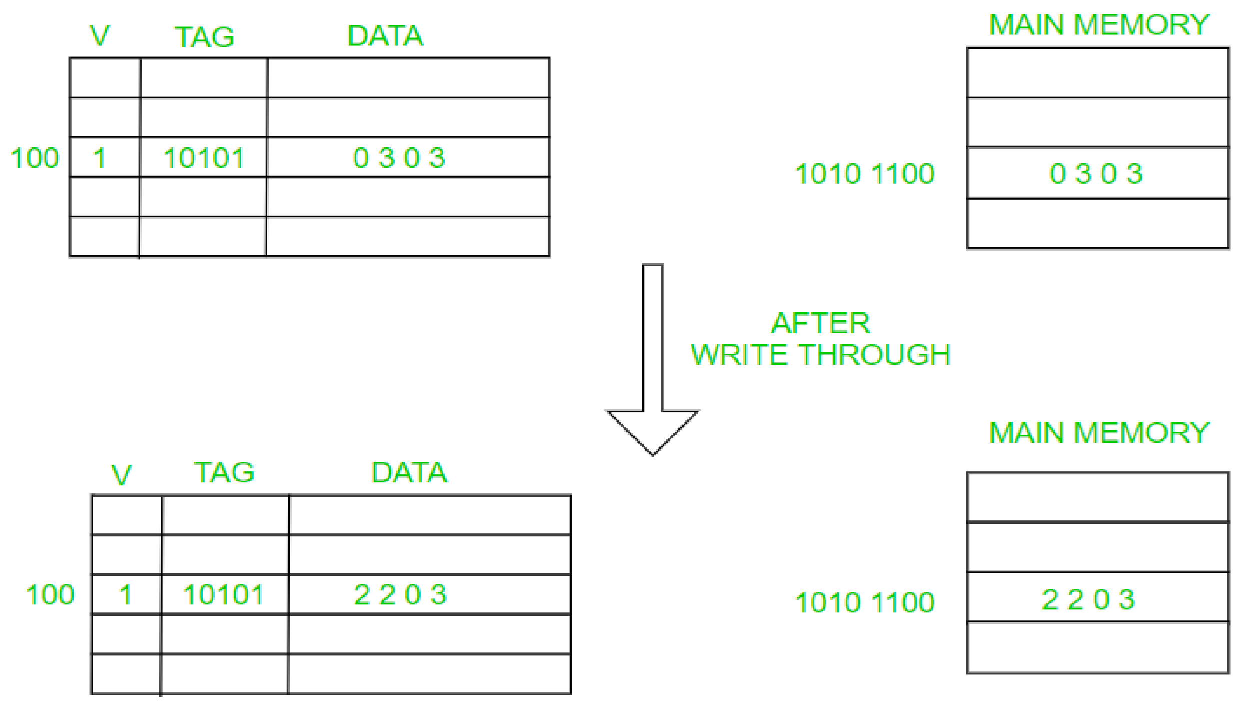
115. What's difference between CPU Cache and TLB?

Ans: **CPU Cache** is a fast memory which is used to improve latency of fetching information from Main memory (RAM) to CPU registers. So CPU Cache sits between Main memory and CPU. And this cache stores information temporarily so that the next access to the same information is faster. A CPU cache which used to store executable instructions, it's called Instruction Cache (I-Cache). A CPU cache which is used to store data, it's called Data Cache (D-Cache). So I-Cache and D-Cache speeds up fetching time for instructions and data respectively. A modern processor contains both I-Cache and D-Cache. For completeness, let us discuss about D-cache hierarchy as well. D-Cache is typically organized in a hierarchy i.e. Level 1 data cache, Level 2 data cache etc.. It should be noted that L1 D-Cache is faster/smaller/costlier as compared to L2 D-Cache. But the basic idea of '*CPU cache*' is to speed up instruction/data fetch time from Main memory to CPU.
Translation Lookaside Buffer (i.e. TLB) is required only if Virtual Memory is used by a processor. In short, TLB speeds up translation of virtual address to physical address by storing page-table in a faster memory. In fact, TLB also sits between CPU and Main memory. Precisely speaking, TLB is used by MMU when virtual address needs to be translated to physical address. By keeping this mapping of virtual-physical addresses in a fast memory, access to page-table improves. It should be noted that page-table (which itself is stored in RAM) keeps track of where virtual pages are stored in the physical memory. In that sense, TLB also can be considered as a cache of the page-table.
But the scope of operation for *TLB* and *CPU Cach*e is different. TLB is about 'speeding up address translation for Virtual memory' so that page-table needn't to be accessed for every address. CPU Cache is about 'speeding up main memory access latency' so that RAM isn't accessed always by CPU. TLB operation comes at the time of address translation by MMU while CPU cache operation comes at the time of memory access by CPU. In fact, any modern processor deploys all I-Cache, L1 & L2 D-Cache and TLB.

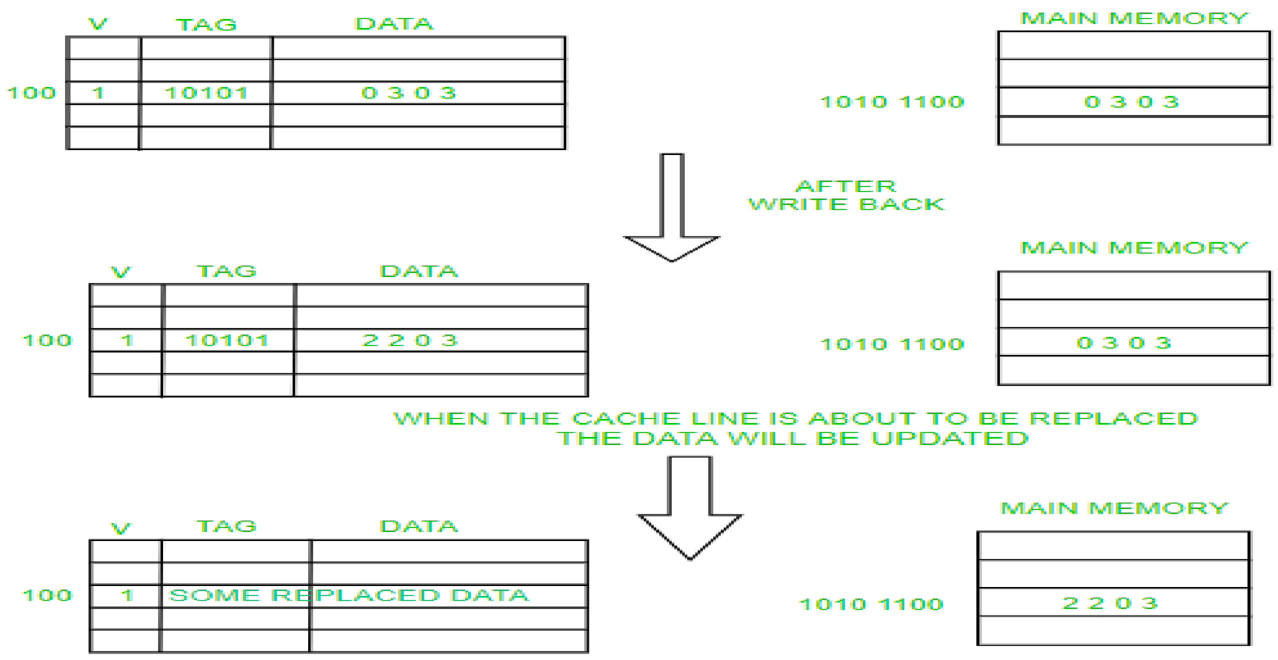


116. Discuss Write Through and Write Back in Cache in detail.
Ans: Cache is a technique of storing a copy of data temporarily in rapidly accessible storage memory. Cache stores most recently used words in small memory to increase the speed in which a data is accessed. It acts like a buffer between RAM and CPU and thus increases the speed in which data is available to the processor.
Whenever a Processor wants to write a word, it checks to see if the address it wants to write the data to, is present in the cache or not. If address is present in the cache i.e., Write Hit.
We can update the value in the cache and avoid a expensive main memory access.But this results in Inconsistent Data Problem.As both cache and main memory have different data, it will cause problem in two or more devices sharing the main memory (as in a multiprocessor system).
This is where Write Through and Write Back comes into picture.

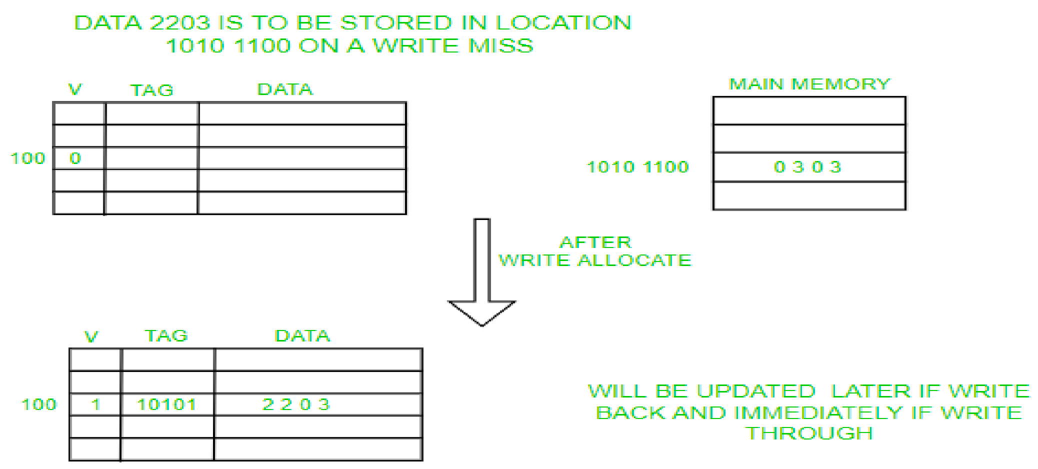
Write Through:



In write through, data is **simultaneously updated to cache and memory**. This process is simpler and more reliable. This is used when there are no frequent writes to the cache (Number of write operation is less).
It helps in data recovery (In case of power outage or system failure). A data write will experience latency (delay) as we have to write to two locations (both Memory and Cache). It Solves the inconsistency problem. But it questions the advantage of having a cache in write operation (As the whole point of using a cache was to avoid multiple accessing to the main memory).
Write Back:

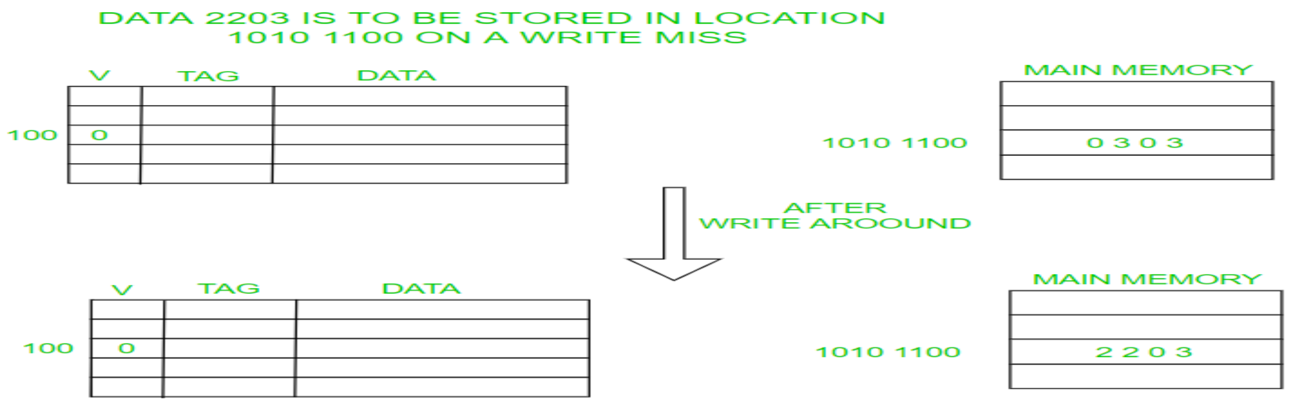


The data is updated only in the cache and updated into the memory in later time. Data is updated in the memory only when the cache line is ready to replaced (cache line replacement is done using Belady's Anomaly, Least Recently Used Algorithm, FIFO, LIFO and others depending on the application).Write Back is also known as Write Deferred.
Dirty Bit : Each Block in the cache needs a bit to indicate if the data present in the cache was modified(Dirty) or not modified(Clean).If it is clean there is no need to write it into the memory. It designed to reduce write operation to a memory. If **Cache fails** or if **System fails or power outage** the modified data will be lost. Because its nearly impossible to restore data from cache if lost.
If write occurs to a location that is not present in the Cache(Write Miss), we use two options, **Write Allocation** and **Write Around**.
Write Allocation:


DATA 2203 IS TO BE STORED IN LOCATION 1010 1100 ON A WRITE MISS
V
TAG
DATA
100
0
MAIN MEMORY
1010 1100
0 3 0 3
AFTER WRITE ALLOCATE
V
TAG
DATA
100
1
10101
2 2 0 3
WILL BE UPDATED LATER IF WRITE BACK AND IMMEDIATELY IF WRITE THROUGH

In Write Allocation data is loaded from the memory into cache and then updated. Write allocation works with both Write back and Write through.But it is generally used with Write Back because it is unnecessary to bring data from the memory to cache and then updating the data in both cache and main memory. Thus Write Through is often used with No write Allocate.

Write Around:



Here data is Directly written/updated to main memory without disturbing cache.It is better to use this when the data is not immediately used again.

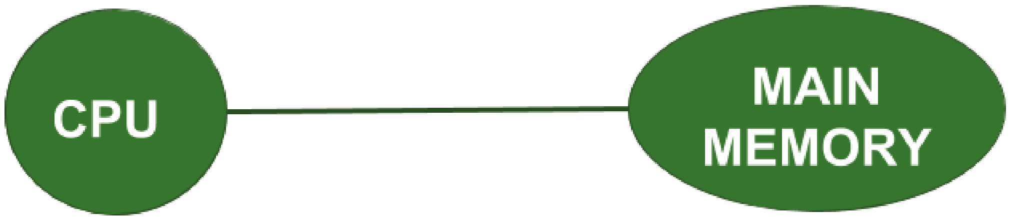
117. Explain in detail Multilevel Cache Organisation .

Ans: **Cache** is a random access memory used by the CPU to reduce the average time taken to access memory.

Multilevel Caches is one of the techniques to improve Cache Performance by reducing the *"MISS PENALTY"*. Miss Penalty refers to the extra time required to bring the data into cache from the Main memory whenever there is a ***"miss"*** in cache .

For clear understanding let us consider an example where CPU requires 10 Memory References for accessing the desired information and consider this scenario in the following 3 cases of System design :

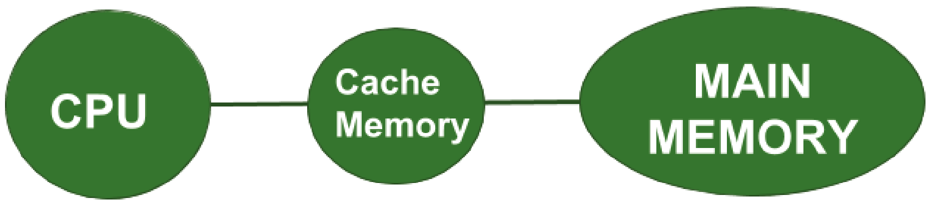
Case 1 : System Design without Cache Memory



Here the CPU directly communicates with the main memory and no caches are involved.

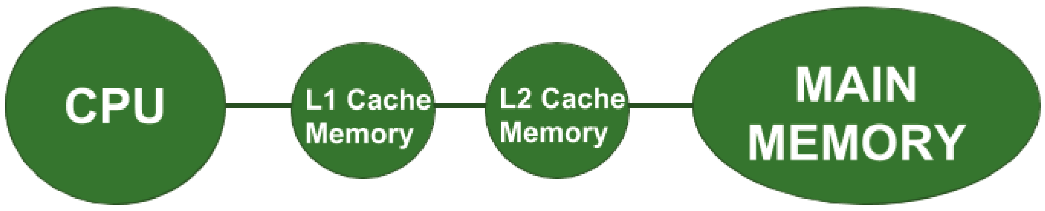
In this case, the CPU needs to access the main memory 10 times to access the desired information.

Case 2 : System Design with Cache Memory



Here the CPU at first checks whether the desired data is present in the Cache Memory or not i.e. whether there is a ***"hit"*** in cache or ***"miss"*** in cache. Suppose there are *3 miss* in Cache Memory then the Main Memory will be accessed only 3 times. We can see that here the miss penalty is reduced because the Main Memory is accessed a lesser number of times than that in the previous case.

Case 3 : System Design with Multilevel Cache Memory



Here the Cache performance is optimized further by introducing multilevel Caches. As shown in the above figure, we are considering 2 level Cache Design. Suppose there are *3 miss* in the L1 Cache Memory and out of these 3 misses there are *2 miss* in the L2 Cache Memory then the Main Memory will be accessed only 2 times. It is clear that here the Miss Penalty is reduced considerably than that in the previous case thereby improving the Performance of Cache Memory.

NOTE :

We can observe from the above 3 cases that we are trying to decrease the number of Main Memory References and thus decreasing the Miss Penalty in order to improve the overall System Performance. Also, it is important to note that in the Multilevel Cache Design, L1 Cache is attached to the CPU and it is small in size but fast. Although, L2 Cache is attached to the Primary Cache i.e. L1 Cache and it is larger in size and slower but still faster than the Main Memory.
Effective Access Time = Hit rate * Cache access time
+ Miss rate * Lower level access time

Average access Time For Multilevel Cache:(T_{avg})
$T_{avg} = H_1 * C_1 + (1 – H_1) * (H_2 * C_2 +(1 – H_2) *M)$

where
H1 is the Hit rate in the L1 caches.
H2 is the Hit rate in the L2 cache.
C1 is the Time to access information in the L1 caches.
C2 is the Miss penalty to transfer information from the L2 cache to an L1 cache.
M is the Miss penalty to transfer information from the main memory to the L2 cache.
Example:
Find the Average memory access time for a processor with a 2 ns clock cycle time, a miss rate of 0.04 misses per instruction, a miss penalty of 25 clock cycles, and a cache access time (including hit detection) of 1 clock cycle. Also, assume that the read and write miss penalties are the same and ignore other write stalls.
Solution:
Average Memory access time(AMAT)= Hit Time + Miss Rate * Miss Penalty.
Hit Time = 1 clock cycle (Hit time = Hit rate * access time) but here Hit time is directly given so,
Miss rate = 0.04
Miss Penalty= 25 clock cycle (this is time taken by the above level of memory after the hit)
so, AMAT= 1 + 0.04 * 25
AMAT= 2 clock cycle
according to question 1 clock cycle = 2 ns

AMAT = 4ns

118. Explain Amdahl’s law.

Ans: It is named after computer scientist Gene Amdahl(a computer architect from IBM and Amdahl corporation), and was presented at the AFIPS Spring Joint Computer Conference in 1967. It is also known as ***Amdahl’s argument***. It is a formula which gives the theoretical speedup in latency of the execution of a task at a fixed workload that can be expected of a system whose resources are improved. In other words, it is a formula used to find the maximum improvement possible by just improving a particular part of a system. It is often used in *parallel computing* to predict the theoretical speedup when using multiple processors.

Speedup-

Speedup is defined as the ratio of performance for the entire task using the enhancement and performance for the entire task without using the enhancement or speedup can be defined as the ratio of execution time for the entire task without using the enhancement and execution time for the entire task using the enhancement.

If **Pe** is the performance for entire task using the enhancement when possible, **Pw** is the performance for entire task without using the enhancement, **Ew** is the execution time for entire task without using the enhancement and **Ee** is the execution time for entire task using the enhancement when possible then,

Speedup = Pe/Pw

or

Speedup = Ew/Ee

Amdahl’s law uses two factors to find speedup from some enhancement –

- **Fraction enhanced** – The fraction of the computation time in the original computer that can be converted to take advantage of the enhancement. For example- if 10 seconds of the execution time of a program that takes 40 seconds in total can use an enhancement , the fraction is 10/40. This obtained value is *Fraction Enhanced*.

Fraction enhanced is always less than 1.

- **Speedup enhanced** – The improvement gained by the enhanced execution mode; that is, how much faster the task would run if the enhanced mode were used for the entire program. For example – If the enhanced mode takes, say 3 seconds for a portion of the program, while it is 6 seconds in the original mode, the improvement is 6/3. This value is Speedup enhanced.

Speedup Enhanced is always greater than 1.

The overall Speedup is the ratio of the execution time:-

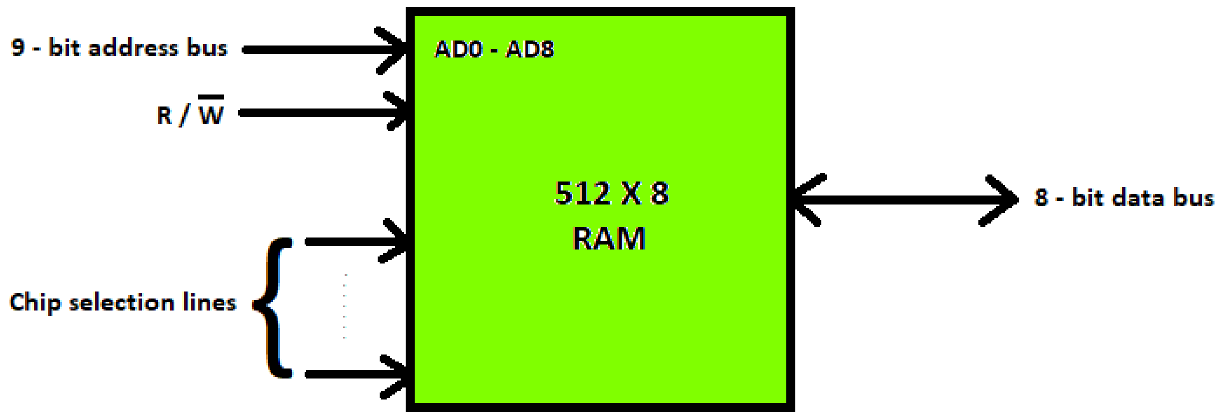
$$\text{Overall Speedup} = \frac{\text{Old execution time}}{\text{New execution time}}$$

$$= \frac{1}{\left((1 - \text{Fraction}_{\text{enhanced}}) + \dfrac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}} \right)}$$

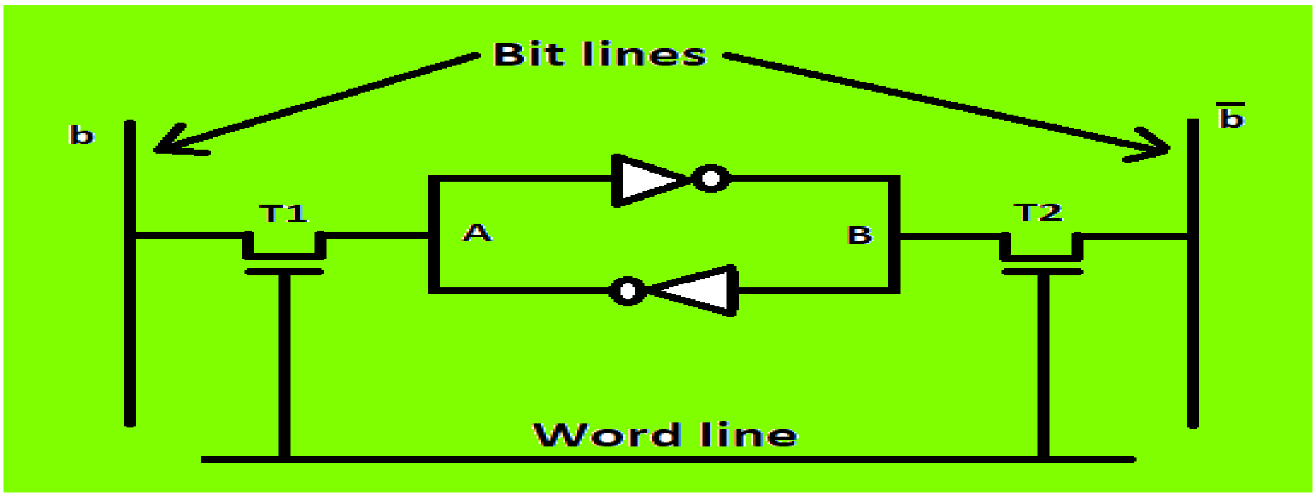
119. Explain about different types of RAM (Random Access Memory).
Ans: RAM(Random Access Memory) is a part of computer's Main Memory which is directly accessible by CPU. RAM is used to Read and Write data into it which is accessed by CPU randomly. RAM is volatile in nature, it means if the power goes off, the stored information is lost. RAM is used to store the data that is currently processed by the CPU. Most of the programs and data that are modifiable are stored in RAM.
Integrated RAM chips are available in two form:

1. SRAM(Static RAM)
2. DRAM(Dynamic RAM)

The block diagram of RAM chip is given below.

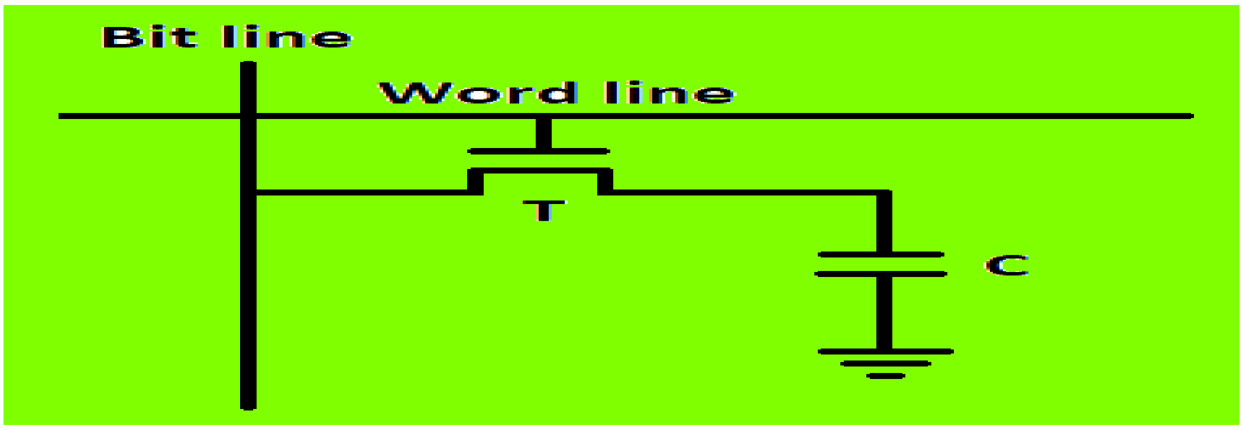


SRAM
The SRAM memories consist of circuits capable of retaining the stored information as long as the power is applied. That means this type of memory requires constant power. SRAM memories are used to build Cache Memory.
SRAM Memory Cell: Static memories(SRAM) are memories that consist of circuits capable of retaining their state as long as power is on. Thus this type of memories is called volatile memories. The below figure shows a cell diagram of SRAM. A latch is formed by two inverters connected as shown in the figure. Two transistors T1 and T2 are used for connecting the latch with two bit lines. The purpose of these transistors is to act as switches that can be opened or closed under the control of the word line, which is controlled by the address decoder. When the word line is at 0-level, the transistors are turned off and the latch remains its information. For example, the cell is at state 1 if the logic value at point A is 1 and at point B is 0. This state is retained as long as the word line is not activated.



For **Read operation**, the word line is activated by the address input to the address decoder. The activated word line closes both the transistors (switches) T1 and T2. Then the bit values at points A

and B can transmit to their respective bit lines. The sense/write circuit at the end of the bit lines sends the output to the processor.
For **Write operation**, the address provided to the decoder activates the word line to close both the switches. Then the bit value that to be written into the cell is provided through the sense/write circuit and the signals in bit lines are then stored in the cell.
DRAM
DRAM stores the binary information in the form of electric charges that applied to capacitors. The stored information on the capacitors tend to lose over a period of time and thus the capacitors must be periodically recharged to retain their usage. The main memory is generally made up of DRAM chips.
DRAM Memory Cell: Though SRAM is very fast, but it is expensive because of its every cell requires several transistors. Relatively less expensive RAM is DRAM, due to the use of one transistor and one capacitor in each cell, as shown in the below figure., where C is the capacitor and T is the transistor. Information is stored in a DRAM cell in the form of a charge on a capacitor and this charge needs to be periodically recharged.
For storing information in this cell, transistor T is turned on and an appropriate voltage is applied to the bit line. This causes a known amount of charge to be stored in the capacitor. After the transistor is turned off, due to the property of the capacitor, it starts to discharge. Hence, the information stored in the cell can be read correctly only if it is read before the charge on the capacitors drops below some threshold value.



120. Explain Interrupts.
Ans: Interrupt is a signal emitted by hardware or software when a process or an event needs immediate attention. It alerts the processor to a high priority process requiring interruption of the current working process. In I/O devices one of the bus control lines is dedicated for this purpose and is called the *Interrupt Service Routine (ISR).*
When a device raises an interrupt at lets say process i, the processor first completes the execution of instruction i. Then it loads the Program Counter (PC) with the address of the first instruction of the ISR. Before loading the Program Counter with the address, the address of the interrupted instruction is moved to a temporary location. Therefore, after handling the interrupt the processor can continue with process i+1.
While the processor is handling the interrupts, it must inform the device that its request has been recognized so that it stops sending the interrupt request signal. Also, saving the registers so that the interrupted process can be restored in the future, increases the delay between the time an interrupt is received and the start of the execution of the ISR. This is called Interrupt Lattency.
Hardware Interrupts:
In a hardware interrupt, all the devices are connected to the Interrupt Request Line. A single request line is used for all the n devices. To request an interrupt, a device closes its associated switch. When

a device requests an interrupts, the value of INTR is the logical OR of the requests from individual devices.

Sequence of events involved in handling an IRQ:

1. Devices raise an IRQ.
2. Processor interrupts the program currently being executed.
3. Device is informed that its request has been recognized and the device deactivates the request signal.
4. The requested action is performed.
5. Interrupt is enabled and the interrupted program is resumed.

121. Difference between Interrupts and Exceptions.

Ans: **Exceptions and interrupts** are unexpected events which will disrupt the normal flow of execution of instruction(that is currently executing by processor). An exception is an unexpected event from within the processor. Interrupt is an unexpected event from outside the process.

Whenever an exception or interrupt occurs, the hardware starts executing the code that performs an action in response to the exception. This action may involve killing a process, outputting an error message, communicating with an external device, or horribly crashing the entire computer system by initiating a "Blue Screen of Death" and halting the CPU. The instructions responsible for this action reside in the operating system kernel, and the code that performs this action is called the interrupt handler code. Now, We can think of handler code as an operating system subroutine. Then, After the handler code is executed, it may be possible to continue execution after the instruction where the execution or interrupt occurred.

Exception and Interrupt Handling :

Whenever an exception or interrupt occurs, execution transition from user mode to kernel mode where the exception or interrupt is handled. In detail, the following steps must be taken to handle an exception or interrupts.

While entering the kernel, the context (values of all CPU registers) of the currently executing process must first be saved to memory. The kernel is now ready to handle the exception/interrupt.

1. Determine the cause of the exception/interrupt.
2. Handle the exception/interrupt.

When the exception/interrupt have been handled the kernel performs the following steps:

1. Select a process to restore and resume.
2. Restore the context of the selected process.
3. Resume execution of the selected process.

At any point in time, the values of all the registers in the CPU defines the context of the CPU. Another name used for CPU context is CPU state.

The exception/interrupt handler uses the same CPU as the currently executing process. When entering the exception/interrupt handler, the values in all CPU registers to be used by the exception/interrupt handler must be saved to memory. The saved register values can later restored before resuming execution of the process.

The handler may have been invoked for a number of reasons. The handler thus needs to determine the cause of the exception or interrupt. Information about what caused the exception or interrupt can be stored in dedicated registers or at predefined addresses in memory.

Next, the exception or interrupt needs to be serviced. For instance, if it was a keyboard interrupt, then the key code of the key press is obtained and stored somewhere or some other appropriate action is taken. If it was an arithmetic overflow exception, an error message may be printed or the program may be terminated.

The exception/interrupt have now been handled and the kernel. The kernel may choose to resume the same process that was executing prior to handling the exception/interrupt or resume execution of any other process currently in memory.

The context of the CPU can now be restored for the chosen process by reading and restoring all register values from memory.
The process selected to be resumed must be resumed at the same point it was stopped. The address of this instruction was saved by the machine when the interrupt occurred, so it is simply a matter of getting this address and make the CPU continue to execute at this address.

122. What are the different types of interrupts present in 8086 microprocessor?
Ans: An interrupt is a condition that halts the microprocessor temporarily to work on a different task and then return to its previous task. Interrupt is an event or signal that request to attention of CPU. This halt allows peripheral devices to access the microprocessor.
Whenever an interrupt occurs the processor completes the execution of the current instruction and starts the execution of an Interrupt Service Routine (ISR) or Interrupt Handler. ISR is a program that tells the processor what to do when the interrupt occurs. After the execution of ISR, control returns back to the main routine where it was interrupted.
In 8086 microprocessor following tasks are performed when microprocessor encounters an interrupt:

1. The value of flag register is pushed into the stack. It means that first the value of SP (Stack Pointer) is decremented by 2 then the value of flag register is pushed to the memory address of stack segment.
2. The value of starting memory address of CS (Code Segment) is pushed into the stack.
3. The value of IP (Instruction Pointer) is pushed into the stack.
4. IP is loaded from word location (Interrupt type) * 04.
5. CS is loaded from the next word location.
6. Interrupt and Trap flag are reset to 0.

The different types of interrupts present in 8086 microprocessor are given by:

1. **Hardware Interrupts-** Hardware interrupts are those interrupts which are caused by any peripheral device by sending a signal through a specified pin to the microprocessor. There are two hardware interrupts in 8086 microprocessor. They are:
   - *(A) NMI (Non Maskable Interrupt)* – It is a single pin non maskable hardware interrupt which cannot be disabled. It is the highest priority interrupt in 8086 microprocessor. After its execution, this interrupt generates a TYPE 2 interrupt. IP is loaded from word location 00008 H and CS is loaded from the word location 0000A H.
   - *(B) INTR (Interrupt Request)* – It provides a single interrupt request and is activated by I/O port. This interrupt can be masked or delayed. It is a level triggered interrupt. It can receive any interrupt type, so the value of IP and CS will change on the interrupt type received.
2. **Software Interrupts –** These are instructions that are inserted within the program to generate interrupts. There are 256 software interrupts in 8086 microprocessor. The instructions are of the format INT type where type ranges from 00 to FF. The starting address ranges from 00000 H to 003FF H. These are 2 byte instructions. IP is loaded from type * 04 H and CS is loaded from the next address give by (type * 04) + 02 H. Some important software interrupts are:
   - (A) *TYPE 0* corresponds to division by zero(0).
   - (B) *TYPE 1* is used for single step execution for debugging of program.
   - (C) *TYPE 2* represents NMI and is used in power failure conditions.
   - (D) *TYPE 3* represents a break-point interrupt.
   - (E) *TYPE 4* is the overflow interrupt.

123. Discuss the diffirent Mode of Data Transfer.
Ans: The binary information that is received from an external device is usually stored in the memory unit. The information that is transferred from the CPU to the external device is originated from the memory unit. CPU merely processes the information but the source and target is always the memory unit. Data transfer between CPU and the I/O devices may be done in different modes.
Data transfer to and from the peripherals may be done in any of the three possible ways

1. Programmed I/O.
2. Interrupt- initiated I/O.
3. Direct memory access(DMA).
   Now let's discuss each mode one by one.
1. **Programmed I/O:** It is due to the result of the I/O instructions that are written in the computer program. Each data item transfer is initiated by an instruction in the program. Usually the transfer is from a CPU register and memory. In this case it requires constant monitoring by the CPU of the peripheral devices.
   Example of Programmed I/O: In this case, the I/O device does not have direct access to the memory unit. A transfer from I/O device to memory requires the execution of several instructions by the CPU, including an input instruction to transfer the data from device to the CPU and store instruction to transfer the data from CPU to memory. In programmed I/O, the CPU stays in the program loop until the I/O unit indicates that it is ready for data transfer. This is a time consuming process since it needlessly keeps the CPU busy. This situation can be avoided by using an interrupt facility. This is discussed below.
2. **Interrupt- initiated I/O:** Since in the above case we saw the CPU is kept busy unnecessarily. This situation can very well be avoided by using an interrupt driven method for data transfer. By using interrupt facility and special commands to inform the interface to issue an interrupt request signal whenever data is available from any device. In the meantime the CPU can proceed for any other program execution. The interface meanwhile keeps monitoring the device. Whenever it is determined that the device is ready for data transfer it initiates an interrupt request signal to the computer. Upon detection of an external interrupt signal the CPU stops momentarily the task that it was already performing, branches to the service program to process the I/O transfer, and then return to the task it was originally performing.
   Note: Both the methods programmed I/O and Interrupt-driven I/O require the active intervention of the
   processor to transfer data between memory and the I/O module, and any data transfer must transverse a path through the processor. Thus both these forms of I/O suffer from two inherent drawbacks.
   - The I/O transfer rate is limited by the speed with which the processor can test and service a device.
   - The processor is tied up in managing an I/O transfer; a number of instructions must be executed
     for each I/O transfer.
3. **Direct Memory Access**: The data transfer between a fast storage media such as magnetic disk and memory unit is limited by the speed of the CPU. Thus we can allow the peripherals directly communicate with each other using the memory buses, removing the intervention of the CPU. This type of data transfer technique is known as DMA or direct memory access. During DMA the CPU is idle and it has no control over the memory buses. The DMA controller takes over the buses to manage the transfer directly between the I/O devices and the memory unit.

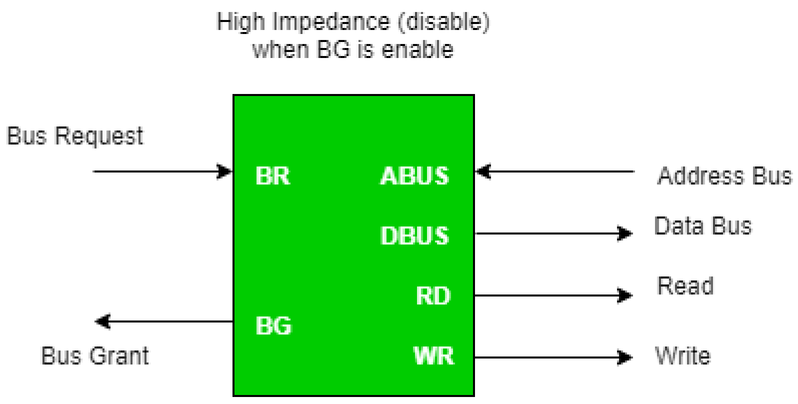


Figure - CPU Bus Signals for DMA Transfer

Bus Request : It is used by the DMA controller to request the CPU to relinquish the control of the buses.
Bus Grant : It is activated by the CPU to Inform the external DMA controller that the buses are in high impedance state and the requesting DMA can take control of the buses. Once the DMA has taken the control of the buses it transfers the data. This transfer can take place in many ways.

124. Difference between Maskable and Non Maskable Interrupt.
Ans: An interrupt is an event caused by a component other than the CPU. It indicates the CPU of an external event that requires immediate attention. Interrupts occur asynchronously. Maskable and non-maskable interrupts are two types of interrupts.

1. **Maskable Interrupt :**

An Interrupt that can be disabled or ignored by the instructions of CPU are called as Maskable Interrupt.The interrupts are either edge-triggered or level-triggered or level-triggered.
Eg:
RST6.5,RST7.5,RST5.5 of 8085

2. **Non-Maskable Interrupt :**

An interrupt that cannot be disabled or ignored by the instructions of CPU are called as Non-Maskable Interrupt.A Non-maskable interrupt is often used when response time is critical or when an interrupt should never be disable during normal system operation. Such uses include reporting non-recoverable hardware errors, system debugging and profiling and handling of species cases like system resets.
Eg:
Trap of 8085

Difference between maskable and nonmaskable interrupt :

SR.NO.	MASKABLE INTERRUPT	NON MASKABLE INTERRUPT
1	Maskable interrupt is a hardware Interrupt that can be	A non-maskable interrupt is a hardware interrupt that cannot be disabled or ignored

SR.NO.	MASKABLE INTERRUPT	NON MASKABLE INTERRUPT
	disabled or ignored by the instructions of CPU.	by the instructions of CPU.
2	When maskable interrupt occur, it can be handled after executing the current instruction.	When non-maskable interrupts occur, the current instructions and status are stored in stack for the CPU to handle the interrupt.
3	Maskable interrupts help to handle lower priority tasks.	Non-maskable interrupt help to handle higher priority tasks such as watchdog timer.
4	Maskable interrupts used to interface with peripheral device.	Non maskable interrupt used for emergency purpose e.g power failure, smoke detector etc .
5	In maskable interrupts, response time is high.	In non maskable interrupts, response time is low.
6	It may be vectored or non-vectored.	All are vectored interrupts.
7	Operation can be masked or made pending.	Operation Cannot be masked or made pending.
8	RST6.5, RST7.5, and RST5.5 of 8085 are some common examples of maskable Interrupts.	Trap of 8085 microprocessor is an example for non-maskable interrupt.

125. Difference between Interrupt and Polling.
Ans:
Interrupt:
Interrupt is a hardware mechanism in which, the device notices the CPU that it requires its attention. Interrupt can take place at any time. So when CPU gets an interrupt signal trough the indication interrupt-request line, CPU stops the current process and respond to the interrupt by passing the control to interrupt handler which services device.
Polling:
In polling is not a hardware mechanism, its a protocol in which CPU steadily checks whether the device needs attention. Wherever device tells process unit that it desires hardware processing, in polling process unit keeps asking the I/O device whether or not it desires CPU processing. The CPU ceaselessly check every and each device hooked up thereto for sleuthing whether or not any device desires hardware attention.
Each device features a command-ready bit that indicates the standing of that device, i.e., whether or not it's some command to be dead by hardware or not. If command bit is ready one, then it's some command to be dead else if the bit is zero, then it's no commands.
Let's see that the difference between interrupt and polling:

S.NO	INTERRUPT	POLLING
1.	In interrupt, the device notices the CPU that it requires its attention.	Whereas, in polling, CPU steadily checks whether the device needs attention.
2.	An interrupt is not a protocol, its a hardware mechanism.	Whereas it isn't a hardware mechanism, its a protocol.
3.	In interrupt, the device is serviced by interrupt handler.	While in polling, the device is serviced by CPU.
4.	Interrupt can take place at any time.	Whereas CPU steadily ballots the device at regular or proper interval.
5.	In interrupt, interrupt request line is used as indication for indicating that device requires servicing.	While in polling, Command ready bit is used as indication for indicating that device requires servicing.
6.	In interrupts, processor is simply	On the opposite hand, in polling, processor

S.NO	INTERRUPT	POLLING
	disturbed once any device interrupts it.	waste countless processor cycles by repeatedly checking the command-ready little bit of each device.

126. What do you mean by DMA?
Ans: **Direct Memory Access**: The data transfer between a fast storage media such as magnetic disk and memory unit is limited by the speed of the CPU. Thus we can allow the peripherals directly communicate with each other using the memory buses, removing the intervention of the CPU. This type of data transfer technique is known as DMA or direct memory access. During DMA the CPU is idle and it has no control over the memory buses. The DMA controller takes over the buses to manage the transfer directly between the I/O devices and the memory unit.

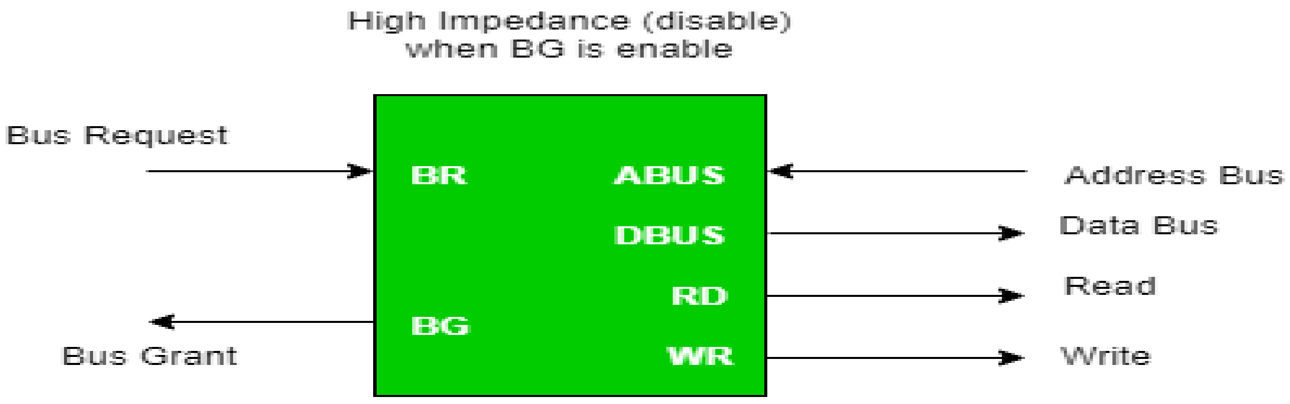


Figure - CPU Bus Signals for DMA Transfer

Bus Request : It is used by the DMA controller to request the CPU to relinquish the control of the buses.
Bus Grant : It is activated by the CPU to Inform the external DMA controller that the buses are in high impedance state and the requesting DMA can take control of the buses. Once the DMA has taken the control of the buses it transfers the data. This transfer can take place in many ways.
Types of DMA transfer using DMA controller:
Burst Transfer :

DMA returns the bus after complete data transfer. A register is used as a byte count, being decremented for each byte transfer, and upon the byte count reaching zero, the DMAC will release the bus. When the DMAC operates in burst mode, the CPU is halted for the duration of the data transfer.
Steps involved are:

1. Bus grant request time.
2. Transfer the entire block of data at transfer rate of device because the device is usually slow than the speed at which the data can be transferred to CPU.

3. Release the control of the bus back to CPU
   So, total time taken to transfer the N bytes
   = Bus grant request time + (N) * (memory transfer rate) + Bus release control time.

Where,
X µsec =data transfer time or preparation time (words/block)
Y µsec =memory cycle time or cycle time or transfer time (words/block)
% CPU idle (Blocked)=(Y/X+Y)*100
% CPU Busy=(X/X+Y)*100
Cyclic Stealing :

An alternative method in which DMA controller transfers one word at a time after which it must return the control of the buses to the CPU. The CPU delays its operation only for one memory cycle to allow the direct memory I/O transfer to "steal" one memory cycle.
Steps Involved are:

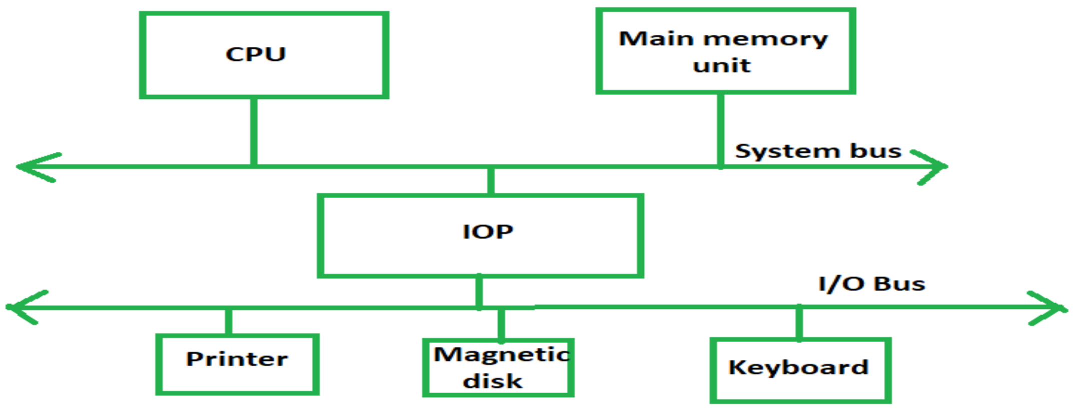
4. Buffer the byte into the buffer
5. Inform the CPU that the device has 1 byte to transfer (i.e. bus grant request)
6. Transfer the byte (at system bus speed)
7. Release the control of the bus back to CPU.

Before moving on transfer next byte of data, device performs step 1 again so that bus isn't tied up and
the transfer won't depend upon the transfer rate of device.
So, for 1 byte of transfer of data, time taken by using cycle stealing mode (T).
= time required for bus grant + 1 bus cycle to transfer data + time required to release the bus, it will be
N x T
In cycle stealing mode we always follow pipelining concept that when one byte is getting transferred then Device is parallel preparing the next byte. "The fraction of CPU time to the data transfer time" if asked then cycle stealing mode is used.
Where,
X µsec =data transfer time or preparation time
(words/block)
Y µsec =memory cycle time or cycle time or transfer
time (words/block)
% CPU idle (Blocked) =(Y/X)*100
% CPU busy=(X/Y)*100
Interleaved mode: In this technique , the DMA controller takes over the system bus when the microprocessor is not using it.An alternate half cycle i.e. half cycle DMA + half cycle processor.

127. What do know about Input-Output Processor (IOP) or IO channel?
Ans: The **DMA mode** of data transfer reduces CPU's overhead in handling I/O operations. It also allows parallelism in CPU and I/O operations. Such parallelism is necessary to avoid wastage of valuable CPU time while handling I/O devices whose speeds are much slower as compared to CPU. The concept of DMA operation can be extended to relieve the CPU further from getting involved with the execution of I/O operations. This gives rises to the development of special purpose processor called **Input-Output Processor (IOP) or IO channel**.
The Input Output Processor (IOP) is just like a CPU that handles the details of I/O operations. It is more equipped with facilities than those are available in typical DMA controller. The IOP can fetch and execute its own instructions that are specifically designed to characterize I/O transfers. In

addition to the I/O – related tasks, it can perform other processing tasks like arithmetic, logic, branching and code translation. The main memory unit takes the pivotal role. It communicates with processor by the means of DMA.

The block diagram –



Input-Output Processor

The Input Output Processor is a specialized processor which loads and stores data into memory along with the execution of I/O instructions. It acts as an interface between system and devices. It involves a sequence of events to executing I/O operations and then store the results into the memory.

Advantages –

- The I/O devices can directly access the main memory without the intervention by the processor in I/O processor based systems.
- It is used to address the problems that are arises in Direct memory access method.

Objective

128. A computer has a 256 KByte, 4-way set associative, write back data cache with the block size of 32 Bytes. The processor sends 32-bit addresses to the cache controller. Each cache tag directory entry contains, in addition, to address tag, 2 valid bits, 1 modified bit and 1 replacement bit. The number of bits in the tag field of an address is

a) 11 b) 14 c)16 d) 27

Answer: (C)

129. Any electronic holding place where data can be stored and retrieved later whenever required is

a) memory
b) drive
c) disk
d) circuit

Answer: a)

130. Which of the following is the fastest means of memory access for CPU?

a) Registers
b) Cache
c) Main memory
d) Virtual Memory

Answer: a)

131. The memory implemented using the semiconductor chips is _________

a) Cache

b) Main
c) Secondary
d) Registers

Answer: b)

132. Which of the following is independent of the address bus?

a) Secondary memory
b) Main memory
c) On board memory
d) Cache memory

133. What is the location of the internal registers of CPU?

a) Internal
b) On-chip
c) External
d) Motherboard

Answer: b)

134. MAR stands for ___________

a) Memory address register
b) Main address register
c) Main accessible register
d) Memory accessible register

Answer: a)

135. If M denotes the number of memory locations and N denotes the word size, then an expression that denotes the storage capacity is ______________

a) M*N
b) M+N
c) 2M+N
d) 2M-N

Answer: a)

136. Size of the ________ memory mainly depends on the size of the address bus.

a) Cache
b) Main
c) Secondary
d) Virtual

Answer: b)

137. Cache memory is the on board storage.

a) True
b) False
c) None
d) Both a and b

Answer: a)