C.S.E – 5th Sem

UNIT-I

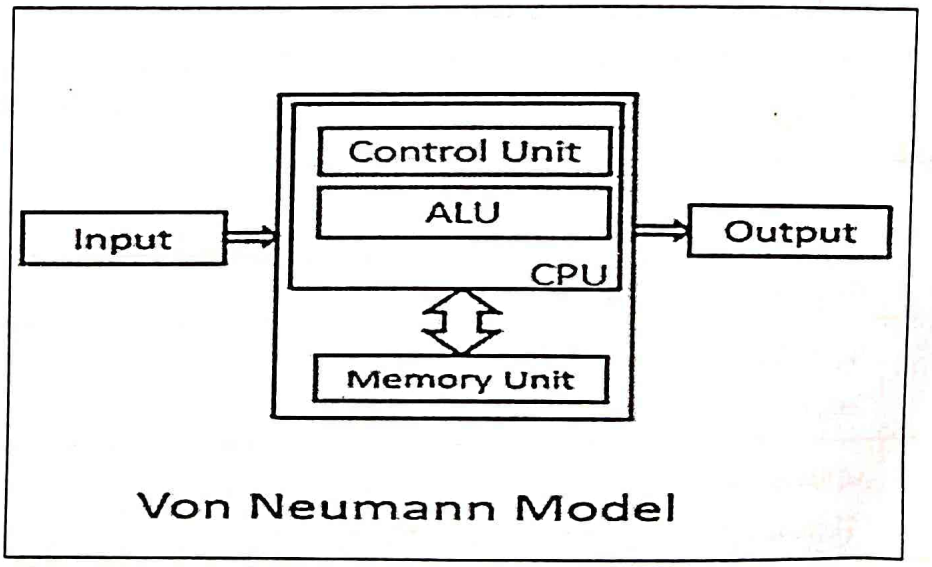
48

Data representation and Computer arithmetic: Introduction to Computer Systems, Organization and architecture, Von Neumann Architecture, evolution and computer generations; Fixed point representation of numbers, digital arithmetic algorithms for Addition, Subtraction, Multiplication using Booth's algorithm and Division using restoring and non-restoring algorithms. Floating point representation with IEEE standards and its arithmetic operations.

Memory Organization: Memory Hierarchy, Main Memory, Auxiliary Memory, Associative Memory, Cache Memory, Virtual Memory, **Cache Coherence and Synchronization Mechanisms (Beyond Curriculum)**

INTRODUCTION TO COMPUTER SYSTEMS

A **computer** is an electronic device that accepts data from the user, processes it, produces results, displays them to the users, and stores the results for future usage. **Data** is a collection of unorganized facts & figures and does not provide any further information regarding patterns, context, etc. Hence data means "unstructured facts and figures". **Information** is a structured data i.e. organized meaningful and processed data. To process the data and convert into information, a computer is used.



Von Neumann Model

Functions of Computers

A computer performs the following functions –

- **Receiving Input**

Data is fed into computer through various input devices like keyboard, mouse, digital pens, etc. Input can also be fed through devices like CD-ROM, pen drive, scanner, etc.

- **Processing the information**

Operations on the input data are carried out based on the instructions provided in the programs.

- **Storing the information**

After processing, the information gets stored in the primary or secondary storage area.

- **Producing output**

The processed information and other details are communicated to the outside world through output devices like monitor, printer, etc.

Sr. No.	COMPUTER CONCEPTS AND DESCRIPTION
1	**History of Computers** The history of the computer dates back to several years. There are five prominent generations of computers. Each generation has witnessed several technological advances which change the functionality of the computers.
2	**Characteristics of Computer System** Characteristics of Computer System involve Speed, Accuracy, Diligence, Versatility, Reliability, Automation, Memory.
3	**Basic Applications of Computer** Computers play a role in every field of life. They are used in homes, business, educational institutions, research organizations, medical field, government offices, entertainment, etc.
4	**Components of Computer System** Computer systems consist of three components: Central Processing Unit, Input devices and Output devices.
5	**Input Devices – Keyboard and Mouse** Input devices help to get input or data from user.
6	**Other Input Devices** There are few other input devices which help to feed data to the computer.
7	**Output Devices** Output devices help to display output to user.
8	**Computer Memory** Computer memory refers to storage area where data is stored. It is of two

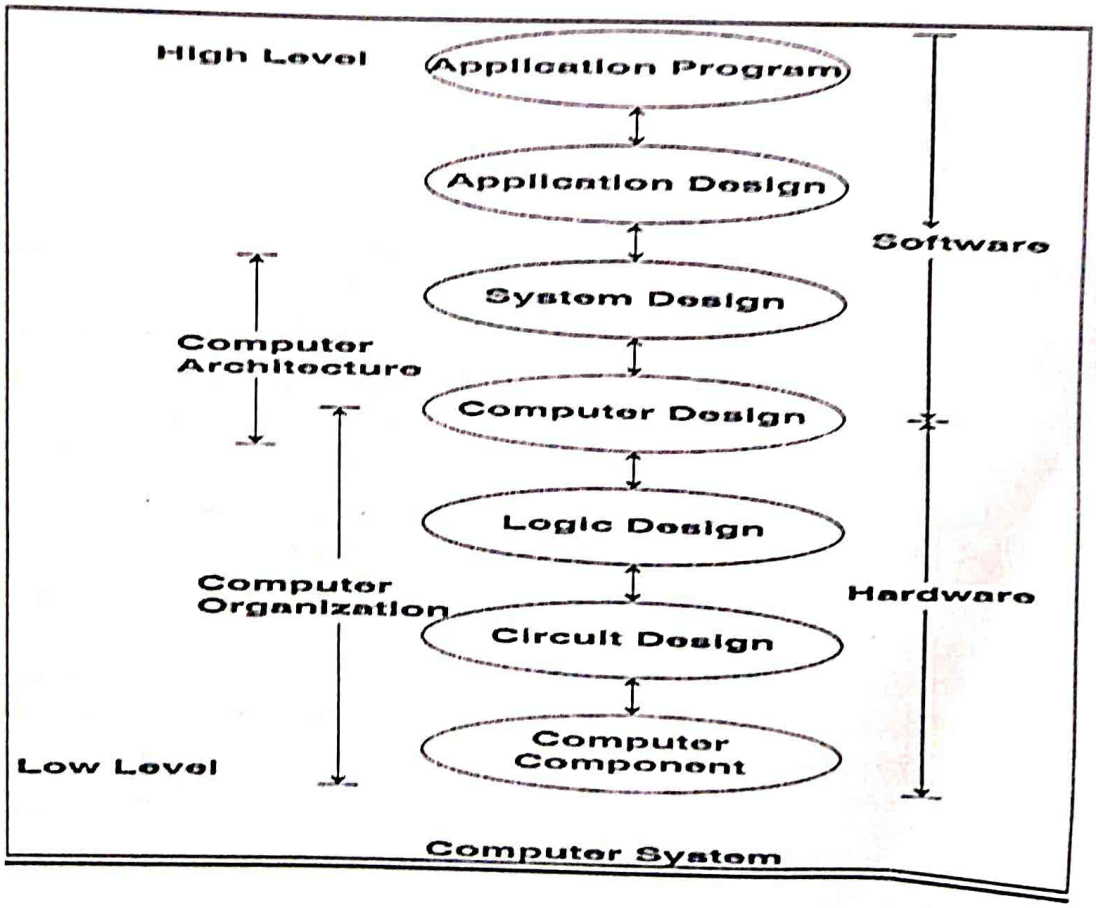
	types Primary Memory & Secondary Memory.
9	**Concept of Hardware and Software** The term hardware refers to mechanical device that makes up computer. Software can be categorized into two types - System software & Application software.
10	**Programming Languages** The languages that are used to write a program or set of instructions are called "Programming languages". Programming languages are broadly categorized into three types - Machine level language, Assembly level language, High-level language.

COMPUTER ORGANIZATION AND ARCHITECTURE

Computer Architecture is a functional description of requirements and design implementation for the various parts of a computer. It deals with the functional behavior of computer systems. It comes before the computer organization while designing a computer.

Architecture describes what the computer does.

Architecture describes what the computer does.



Computer Organization comes after the decision of Computer Architecture first. Computer Organization is how operational attributes are linked together and contribute to realizing the architectural specification. Computer Organization deals with a structural relationship. **The organization describes how it does it.**

Difference between Computer Architecture and Computer Organization:

Sr. No.	Computer Architecture	Computer Organization
1.	Architecture describes what the computer does.	The Organization describes how it does it.
2.	Computer Architecture deals with the functional behavior of computer systems.	Computer Organization deals with a structural relationship.
3.	In the above figure, it's clear that it deals with high-level design issues.	In the above figure, it's also clear that it deals with low-level design issues.
4.	Architecture indicates its hardware.	Where Organization indicates its performance.
5.	As a programmer, you can view architecture as a series of instructions, addressing modes, and registers.	The implementation of the architecture is called organization.
6.	For designing a computer, its architecture is fixed first.	For designing a computer, an organization is decided after its architecture.
7.	Computer Architecture is also called Instruction Set Architecture (ISA).	Computer Organization is frequently called micro-architecture.
8.	Computer Architecture comprises logical functions such as instruction sets, registers, data types, and addressing modes.	Computer Organization consists of physical units like circuit designs, peripherals, and adders.

Sr. No.	Computer Architecture	Computer Organization
9.	The different architectural categories found in our computer systems are as follows: 1. Von-Neumann Architecture 2. Harvard Architecture 3. Instruction Set Architecture 4. Micro-architecture 5. System Design	CPU organization is classified into three categories based on the number of address fields: 1. Organization of a single Accumulator. 2. Organization of general registers 3. Stack organization
10.	It makes the computer's hardware visible.	It offers details on how well the computer performs.
11.	Architecture coordinates the hardware and software of the system.	Computer Organization handles the segments of the network in a system.
12.	The software developer is aware of it.	It escapes the software programmer's detection.
13.	Examples- Intel and AMD created the x86 processor. Sun Microsystems and others created the SPARC processor. Apple, IBM, and Motorola created the PowerPC.	Organizational qualities include hardware elements that are invisible to the programmer, such as interfacing of computer and peripherals, memory technologies, and control signals.

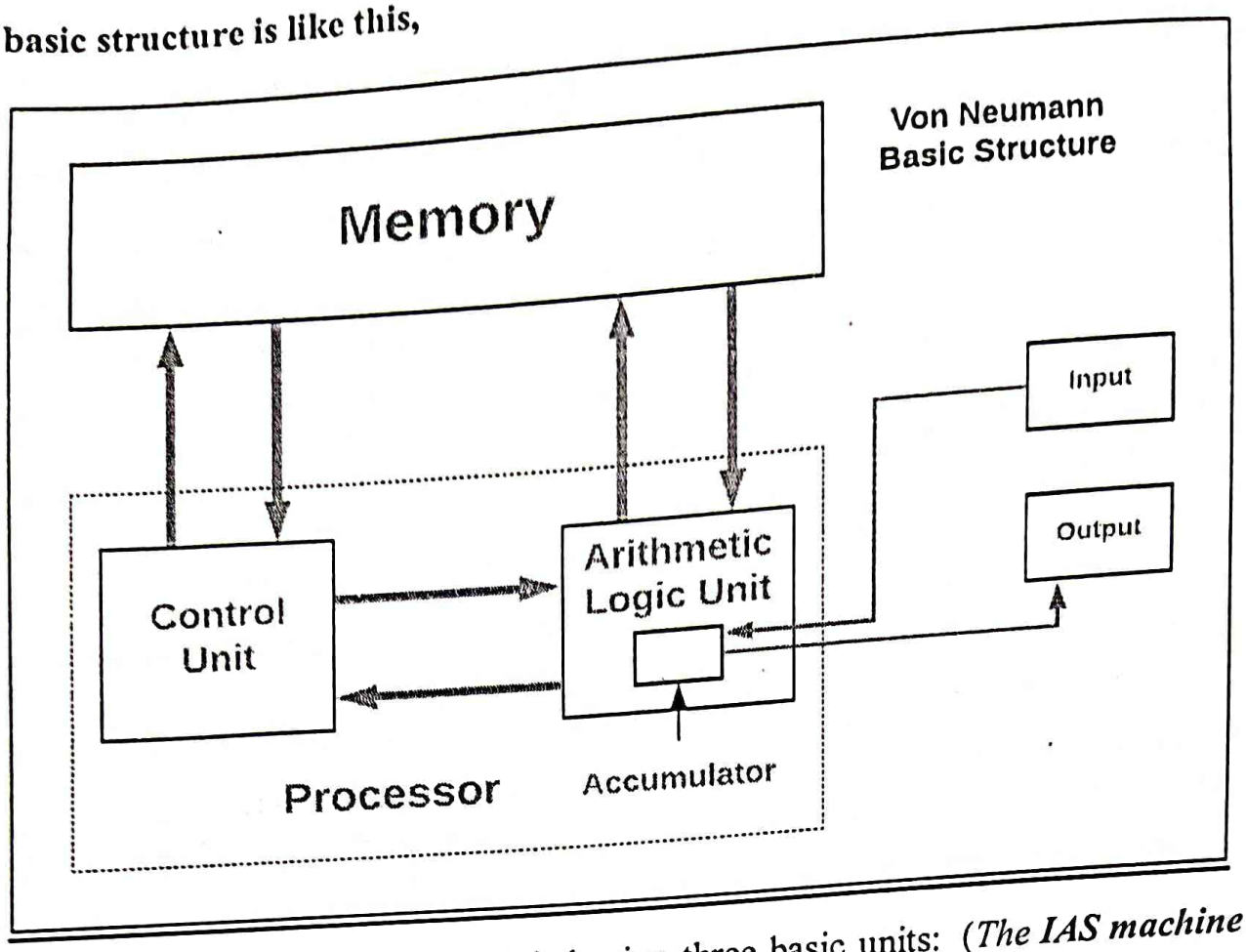
VON NEUMANN ARCHITECTURE | COMPUTER ORGANIZATION

Historically there have been 2 types of Computers:

1. **Fixed Program Computers** – Their function is very specific and they couldn't be re-programmed, e.g. Calculators.
2. **Stored Program Computers** – These can be programmed to carry out many different tasks, applications are stored on them, hence the name.

The modern computers are based on a stored-program concept introduced by John Von Neumann. In this stored-program concept, programs and data are stored in a separate storage unit called memories and are treated the same. This novel idea meant that a computer built with this architecture would be much easier to reprogram.

The basic structure is like this,



It is also known as IAS computer and is having three basic units: (*The **IAS machine** was th*
first electronic computer** built at the **Institute for Advanced Study (IAS)** in **Princeton, Ne
***Jersey**. It is sometimes called the **von Neumann** machine, since the paper describing its desig*
*was **edited by John von Neumann**, a **mathematics professor** at both **Princeton University an***
IAS.)

1. The Central Processing Unit (CPU)
2. The Main Memory Unit
3. The Input/output Device

Let's consider them in details.

- **Control Unit –**
  A control unit (CU) handles all processor control signals. It directs all input and output flow, fetches code for instructions, and controls how data moves around the system.
- **Arithmetic and Logic Unit (ALU) –**
  The arithmetic logic unit is that part of the CPU that handles all the calculations the CPU may need, e.g. Addition, Subtraction, Comparisons. It performs Logical Operations, Bit Shifting Operations, and Arithmetic operations.

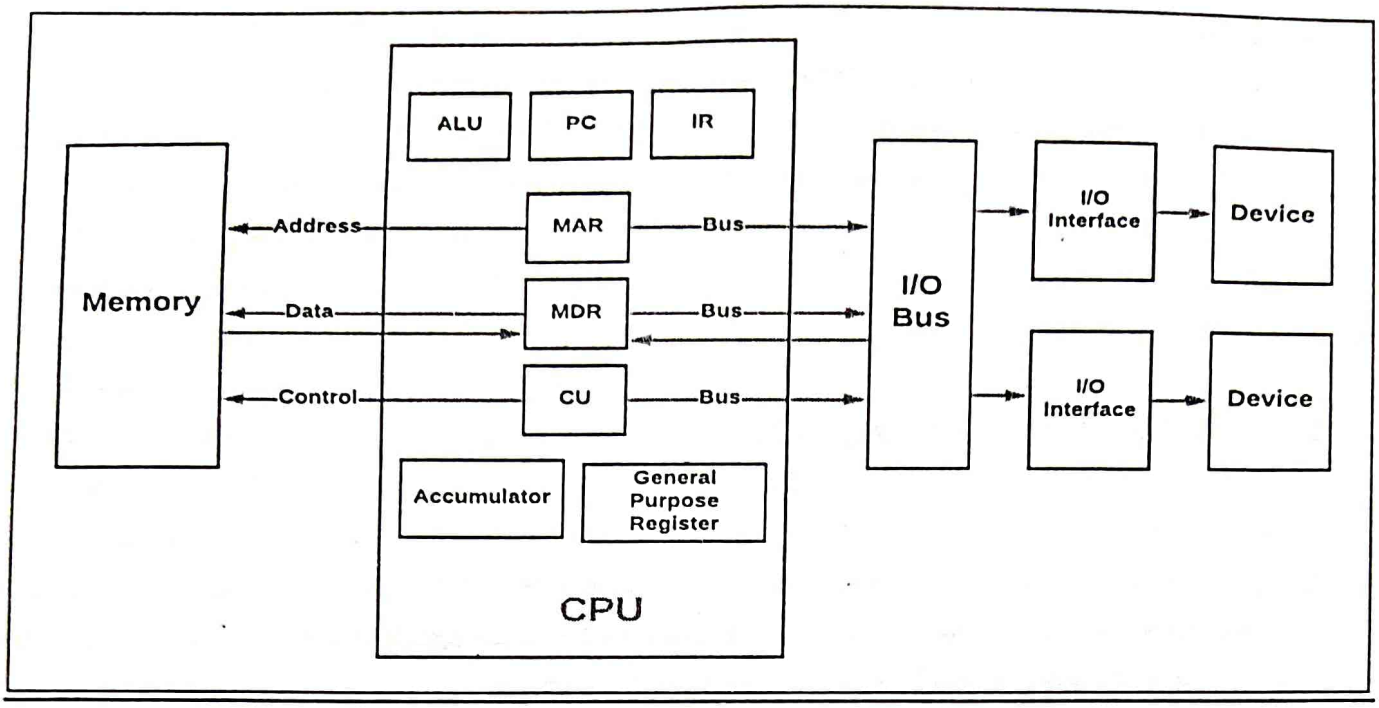


Figure – Basic CPU structure, illustrating ALU

- **Main Memory Unit (Registers) –**
  1. **Accumulator:** Stores the results of calculations made by ALU.
  2. **Program Counter (PC):** Keeps track of the memory location of the next instructions to be dealt with. The PC then passes this next address to Memory Address Register (MAR).
  3. **Memory Address Register (MAR):** It stores the memory locations of instructions that need to be fetched from memory or stored into memory.
  4. **Memory Data Register (MDR):** It stores instructions fetched from memory or any data that is to be transferred to, and stored in, memory.
  5. **Current Instruction Register (CIR):** It stores the most recently fetched instructions while it is waiting to be coded and executed.
  6. **Instruction Buffer Register (IBR):** The instruction that is not to be executed immediately is placed in the instruction buffer register IBR.

- **Input/output Devices** – Program or data is read into main memory from the input device or secondary storage under the control of CPU input instruction. Output devices are used to output the information from a computer. If some results are evaluated by computer and it is stored in the computer, then with the help of output devices, we can present them to the user.

- **Buses** – Data is transmitted from one part of a computer to another, connecting all major internal components to the CPU and memory, by the means of Buses. Types:
  1. **Data Bus:** It carries data among the memory unit, the I/O devices, and the processor.
  2. Address Bus: It carries the address of data (not the actual data) between memory and processor.
  3. **Control Bus:** It carries control commands from the CPU (and status signals from other devices) in order to control and coordinate all the activities within the computer.

EVOLUTION AND COMPUTERS GENERATIONS

A computer is an electronic device that manipulates information or data. It has the ability to store, retrieve, and process data. Nowadays, a computer can be used to type documents, send email, play games, and browse the Web. It can also be used to edit or create spreadsheets, presentations, and even videos. But the evolution of this complex system started around 1940 with the first Generation of Computer and evolving ever since.

There are five generations of computers:

1. FIRST GENERATION

- ***Introduction:***

1. 1946-1959 is the period of first generation computer.
2. J.P.Eckert and J.W.Mauchy invented the first successful electronic computer called ENIAC, ENIAC stands for "Electronic Numeric Integrated And Calculator".

- ***Few Examples are:***

ENIAC, EDVAC, UNIVAC, IBM-701, IBM-650

- ***Advantages:***

1. It made use of vacuum tubes which are the only electronic component available during those days.
2. These computers could calculate in milliseconds.

- ***Disadvantages:***

1. These were very big in size; weight was about 30 tones.
2. These computers were based on vacuum tubes.
3. These computers were very costly.
4. It could store only a small amount of information due to the presence of magnetic drums.
5. As the invention of first generation computers involves vacuum tubes, so another disadvantage of these computers was, vacuum tubes require a large cooling system.
6. Very less work efficiency.

7. Limited programming capabilities and punch cards were used to take inputs.
8. Large amount of energy consumption.
9. Not reliable and constant maintenance is required.

2. SECOND GENERATION

- ***Introduction:***

1959-1965 the period of second-generation computer. Second generation computers were based on Transistor instead of vacuum tubes.

- ***Few Examples are:***

Honeywell 400, IBM 7094, CDC 1604, CDC 3600, UNIVAC 1108many more

- ***Advantages:***

1. Due to the presence of transistors instead of vacuum tubes, the size of electron component decreased. This resulted in reducing the size of a computer as compared to first generation computers.
2. Less energy and not produce as much heat as the first generation.
3. Assembly language and punch cards were used for input.
4. Low cost than first generation computers.
5. Better speed, calculate data in microseconds.
6. Better portability as compared to first generation

- ***Disadvantages:***

1. A cooling system was required.
2. Constant maintenance was required.
3. Only used for specific purposes.

3. THIRD GENERATION

- ***Introduction:***

1. 1965-1971 is the period of third generation computer.
2. These computers were based on Integrated circuits.
3. IC was invented by Robert Noyce and Jack KilbyIn 1958-1959.
4. IC was a single component containing number of transistors.

- ***Few Examples are:***

1. PDP-8
2. PDP-11
3. ICL 2900
4. IBM 360

5. IBM 370

... and many more

- *Advantages:*

1. These computers were cheaper as compared to second-generation computers.
2. They were fast and reliable.
3. Use of IC in the computer provides the small size of the computer.
4. IC not only reduce the size of the computer but it also improves the performance of the computer as compared to previous computers.
5. This generation of computers has big storage capacity.
6. Instead of punch cards, mouse and keyboard are used for input.
7. They used an operating system for better resource management and used the concept of time-sharing and multiple programming.
8. These computers reduce the computational time from microseconds to nanoseconds.

- *Disadvantages:*

1. IC chips are difficult to maintain.
2. The highly sophisticated technology required for the manufacturing of IC chips.
3. Air conditioning is required.

4. FOURTH GENERATION

- *Introduction:*

1. 1971-1980 is the period of fourth generation computer.
2. This technology is based on Microprocessor.
3. A microprocessor is used in a computer for any logical and arithmetic function to be performed in any program.
4. Graphics User Interface (GUI) technology was exploited to offer more comfort to users.

- *Few Examples are:*

1. IBM 4341
2. DEC 10
3. STAR 1000
4. PUP 11

... and many more

- *Advantages:*

1. Fastest in computation and size get reduced as compared to the previous generation of computer.

2. Heat generated is negligible.
3. Small in size as compared to previous generation computers.
4. Less maintenance is required.
5. All types of high-level language can be used in this type of computers.

- ***Disadvantages:***

1. The Microprocessor design and fabrication are very complex.
2. Air conditioning is required in many cases due to the presence of ICs.
3. Advance technology is required to make the ICs.

5. FIFTH GENERATION

- ***Introduction:***

1. The period of the fifth generation in 1980-onwards.
2. This generation is based on artificial intelligence.
3. The aim of the fifth generation is to make a device which could respond to natural language input and are capable of learning and self-organization.
4. This generation is based on ULSI(Ultra Large Scale Integration) technology resulting in the production of microprocessor chips having ten million electronic component.

- ***Few Examples are:***

1. Desktop
2. Laptop
3. NoteBook
4. UltraBook
5. Chromebook

... and many more

- ***Advantages:***

1. It is more reliable and works faster.
2. It is available in different sizes and unique features.
3. It provides computers with more user-friendly interfaces with multimedia features.

- ***Disadvantages:***

1. They need very low-level languages.

FIXED POINT REPRESENTATION OF NUMBER

There are various types of number representation techniques for digital number representation, for example: Binary number system, octal number system, decimal number system, and hexadecimal number system etc. But Binary number system is most relevant and popular for representing numbers in digital computer system.

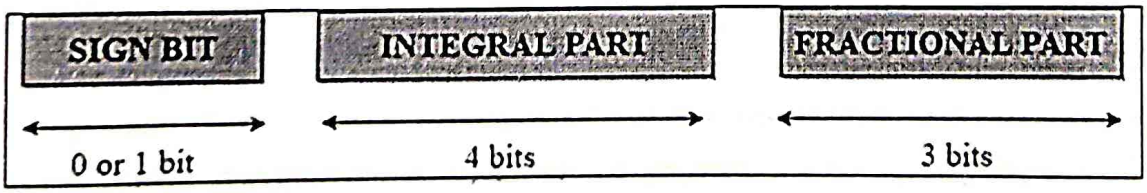
Fixed-point representation has a radix point known as decimal point. Fixed-point numbers having decimal points at the right end of the number are treated as integers because the fixed-point numbers having decimal points at the left end of the number are treated as fractions. In this method, the decimal point position is settled because the number saved in the memory is considered as an integer or as a fraction. The binary numbers that are unsigned are continually considered as positive integers and are defined as 0s in the MSB. The binary numbers that are registered contrast for negative numbers and are defined as 1s in the MSB.

There are two major approaches to store real numbers (i.e., numbers with fractional component) in modern computing. **These are (i) Fixed Point Notation and (ii) Floating Point Notation.** In fixed point notation, there are a fixed number of digits after the decimal point, whereas floating point number allows for a varying number of digits after the decimal point.

Fixed-Point Representation –

In computing, fixed-point number representation is a real data type for a number. With the help of fixed number representation, data is converted into binary form, and then data is processed, stored and used by the system.

Fixed point representation of data



Sign bit -The fixed-point numbers in binary uses a sign bit. A positive number has a sign bit 0, while a negative number has a sign bit 1.

Integral Part – The integral part is of different lengths at different places. It depends on the register's size, like in an 8-bit register, integral part is 4 bits.

Fractional part – Fractional part is also of different lengths at different places. It depends on the register's size, like in an 8-bit register, integral part is of 3 bits.

8 bits = 1Sign bit + 4 bits(integral) + 3bits (fractional part)

16 bits = 1Sign bit + 9 bits(integral) +6 bits (fractional part)

32 bits = 1Sign bit + 15 bits(integral) + 9 bits (fractional part)

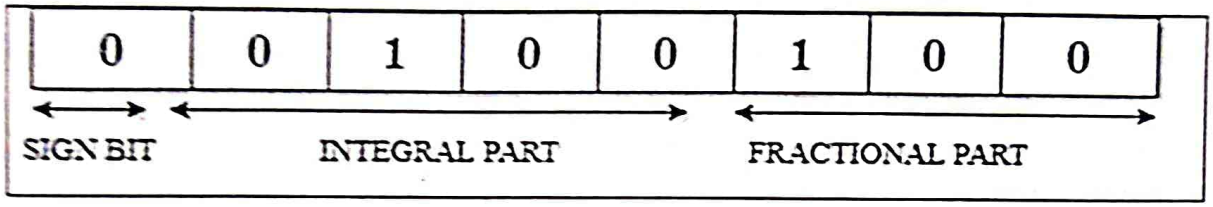
How to write the number in Fixed-point notation?

Number is 4.5

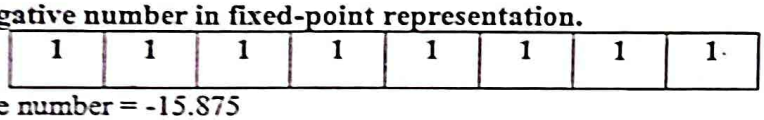
Step 1:- Convert the number into binary form.

4.5 = 100.1

Step 2:- Represent binary number in Fixed point notation

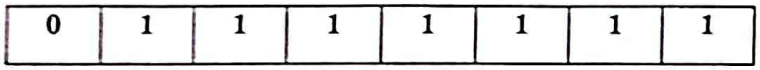


The smallest negative number in fixed-point representation.



Smallest negative number = -15.875

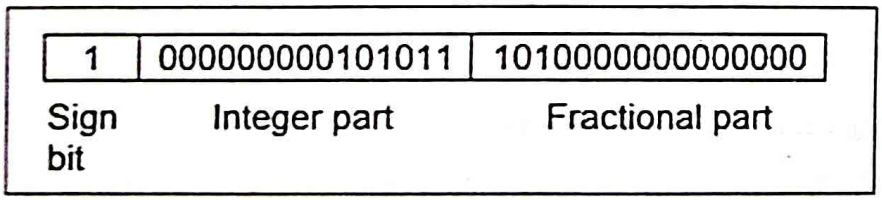
The largest number in fixed-point representation.



Larger number = +15.875

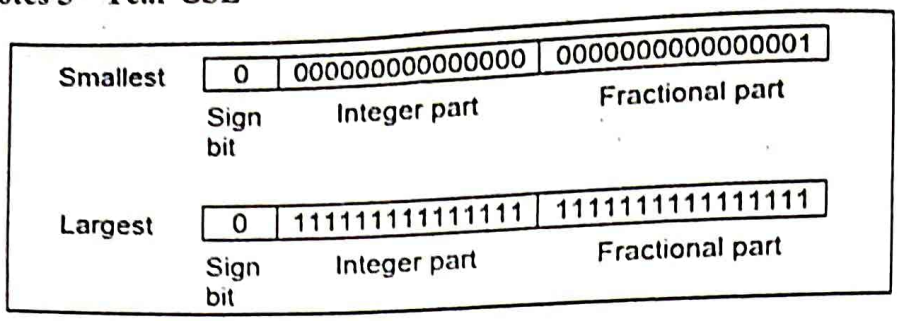
Example –Assume number is using 32-bit format which reserve 1 bit for the sign, 15 bits for the integer part and 16 bits for the fractional part.

Then, -43.625 is represented as following:



Where, 0 is used to represent + and 1 is used to represent. 000000000101011 is 15 bit binary value for decimal 43 and 1010000000000000 is 16 bit binary value for fractional 0.625.

The advantage of using a fixed-point representation is performance and disadvantage is relatively limited range of values that they can represent. So, it is usually inadequate for numerical analysis as it does not allow enough numbers and accuracy. A number whose representation exceeds 32 bits would have to be stored inexactly.



DIGITAL ARITHMETIC ALGORITHMS FOR ADDITION AND SUBTRACTION

An algorithm is a sequence of steps or instructions that outline how to solve a particular problem. One can think of an algorithm as a problem-solving formula or recipe. The term "**algorithm**" derives its name from al-Khwarizmi, an Arab mathematician who wrote a book on algebraic methods.

In the Hindu-Arabic number system, often referred to as the Arabic system, ten numerals or digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) are used in a sequence to form a number. Each digit , when the number is written in expanded form, represents a multiple of a power of ten; for example, 247 = 2 × 102 + 4 × 101 + 2 × 100. Therefore, this number system is called a base-10 , or decimal, number system.

An algorithm that uses ten to an integer power n, 10n, to perform calculations is a base-10 algorithm. Many of the rules that are used in fundamental arithmetic are actually algorithms for the base-10 system.

The base-10 number system is used every day and may seem ordinary, but it is a very powerful and elegant system with which to express numbers. Although most people may not be aware of it, when adding and subtracting numbers, for example, one is essentially using shortcut rules that are based on base-10 algorithms.

Addition Algorithms: Consider 448 + 246.

The base-10 algorithm for the same problem is shown explicitly below.

But 14 × 100 = (10 + 4) × 100 = 10 + 4 × 100. The 10 here is responsible for the 1 "carry-over" to 8, the first digit on the left. Therefore,

= 6 × 102 + (8 + 1) × 101 + 4 × 100

= 6 × 102 + 9 × 101 + 4 × 100 = 600 + 90 + 4 = 694.

Subtraction Algorithms

A base-10 algorithm for solving the subtraction problem 764 - 347 follows.

764 - 347 = (700 + 60 + 4) - (300 + 40 + 7)

Here the idea is to subtract corresponding numbers: 7 from 4, 40 from 60, and 300 from 700. It is known that 7 is greater than 4, and 7 from 4 is -3, but suppose nothing is known about negative numbers. In the first parentheses, write 60 as 50 + 10.

764 - 347 = (700 + 50 + 10 + 4) - (300 + 40 + 7)

In the first parenthesis, add 10 to 4, so 14 is greater than 7.

764 - 347 = (700 + 50 + 14) - (300 + 40 + 7)

So now the resulting problem is to subtract 7 from 14, 40 from 50, and 300 from 700.

764 - 347 = (700 - 300) + (50 - 40) + (14 - 7)

= 400 + 10 + 7 = 417

Another algorithm for subtraction is called "subtraction by taking complements." Consider 764 - 347 again. Adding the same number, c, to both the terms will not affect the answer, because c + (-c) = 0. So effectively the value does not change.

764 - 347 = (764 + c) - (347 + c)

Choose c so that 347 + c becomes a multiple of 10, say 1,000. Therefore, c = 653.

764 - 347 = (764 + 653) - (347 + 653)

= 1,417 - 1,000 = 417

Consider the following flow diagram which derives the algorithm for addition and subtraction of the fixed-point binary numbers:

Eight Conditions for Signed-Magnitude Addition/Subtraction

Operation	Add Magnitudes	SUBTRACT Magnitudes		
		A > B	A < B	A = B
(+A) + (+B)	+ (A + B)			
(+A) + (-B)		+ (A – B)	- (B – A)	+ (A – B)
(-A) + (+B)		- (A – B)	+ (B – A)	+ (A – B)
(-A) + (-B)	- (A + B)			
(+A) - (+B)		+ (A – B)	- (B – A)	+ (A – B)
(+A) - (-B)	+ (A + B)			
(-A) - (+B)	- (A + B)			
(-A) - (-B)		- (A – B)	+ (B – A)	+ (A – B)

As display in the table, the addition algorithm states that –

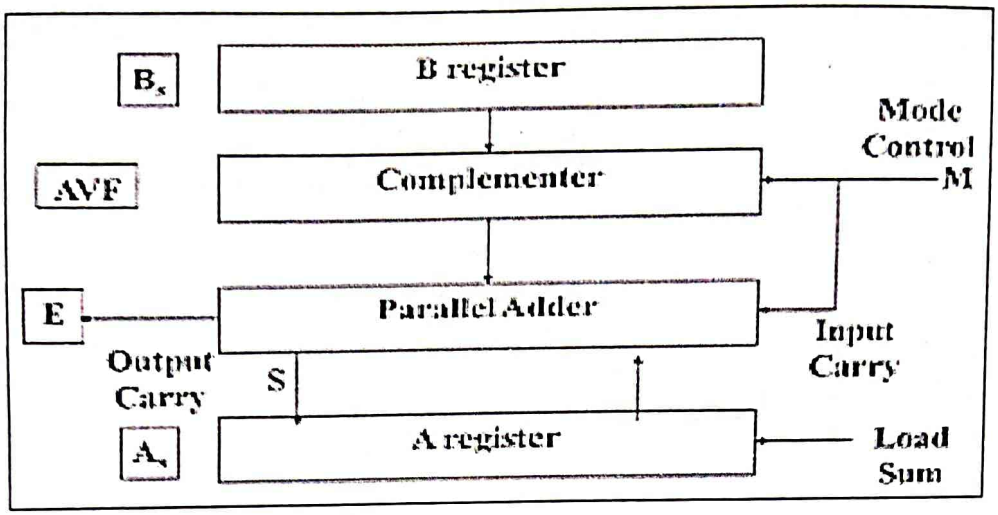
- When the signs of A and B are equal, add the two magnitudes and connect the sign of A to the output.
- When the signs of A and B are different, compare the magnitudes and subtract the smaller number from the greater number.
- The signs of the output have to be equal as A in case A > B or the complement of the sign of A in case A < B.
- When the two magnitudes are equal, subtract B from A and modify the sign of the output to positive.

The subtraction algorithm states that –

- When the signs of A and B are different, add the two magnitudes and connect the signs of A to the output.
- When the signs of A and B are the same, compare the magnitudes and subtract the smaller number from the greater number.
- The signs of the output have to be equal as A in case A > B or the complement of the sign of A in case A < B.
- When the two magnitudes are equal, subtract B from A and modify the sign of the output to positive.

The last column is required to avoid a negative Zero. In other words, when two equal numbers are subtracted the result should be +0 not -0.

Hardware for Singed Magnitude addition and subtraction:

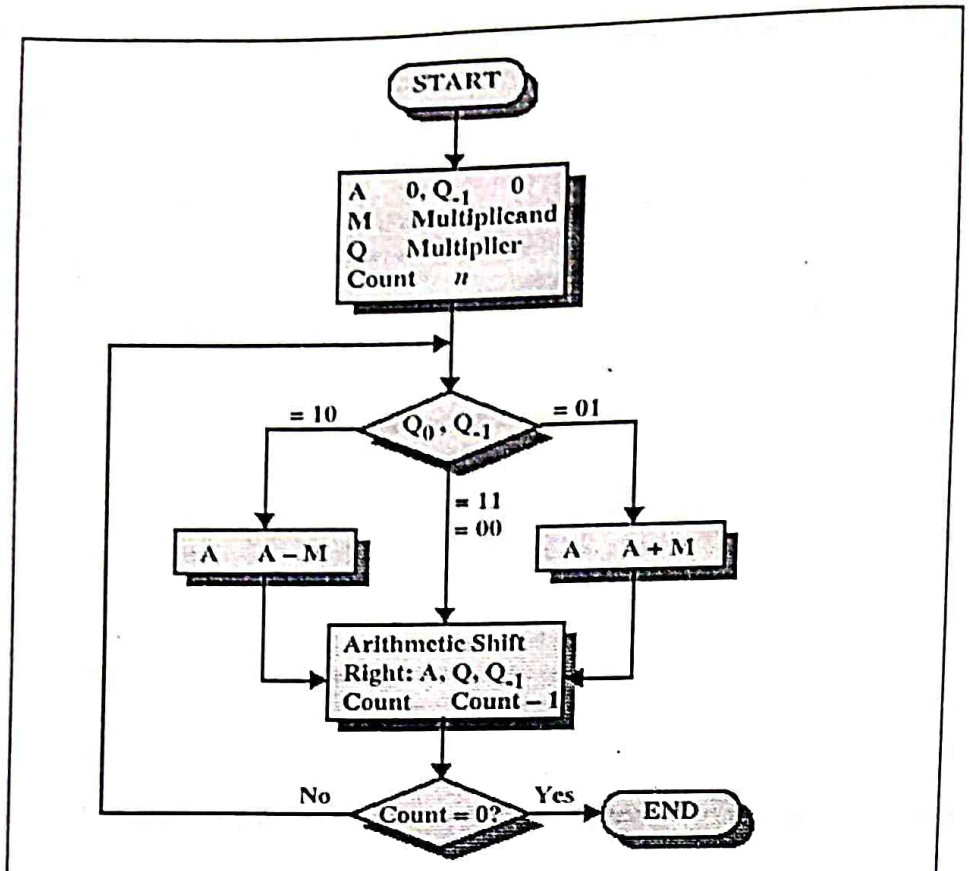


To implement the two arithmetic operations with hardware, we have to store numbers into two registers A and B. Let As and Bs be two flip-flops that hold the corresponding signs. The results are transferred to A and As. A and As together form an accumulator.

- Two registers A and B and sign flip-flops As and Bs.
- A magnitude comparator: to check if A>B, A<B and A=B.
- Hardware above consists of registers A and B and sign flip-flops As and Bs.
- The complementary provides an output of B or B' depending on mode input M.
- When M=0, the output of B is transferred to the adder, the input carry is 0 and thus output of adder is A+B. Output carry is transferred to flip-flop E, where it can be checked to determine overflow. When E=1, it is overflow, otherwise, not. The E=1 is transferred to the AVF (add-overflow-flip-flop) which holds the overflow bit when A ns B are added.
- When M=1, 1's complement of B is applied to the adder, input carry is 1 and output is S=A+B'+1 (i.e. A-B). Output carry is transferred to flip-flop E, where it can be checked to determine the relative magnitude of two numbers.

MULTIPLICATION USING BOOTH'S ALGORITHM

The booth algorithm is a multiplication algorithm that allows us to multiply the two signed binary integers in 2's complement, respectively. It is also used to speed up the performance of the multiplication process. It is very efficient too. It works on the string bits 0's in the multiplier that requires no additional bit only shift the right-most string bits and a string of 1's in a multiplier bit weight 2^k to weight 2^m that can be considered as $2^{k+1} - 2^m$.



Booth's Algorithm for Twos Complement Multiplication

1. Set the Multiplicand and Multiplier binary bits as M and Q, respectively.
2. Initially, we set the AC and Qn - 1 registers value to 0.
3. SC represents the number of Multiplier bits (Q), and it is a sequence counter that is continuously decremented till equal to the number of bits (n) or reached to 0.
4. A Qn represents the last bit of the Q, and the Qn-1 shows the incremented bit of Qn by 1.
5. On each cycle of the booth algorithm, Qn and Qn - 1 bits will be checked on the following parameters as follows:
   i. When two bits Qn and Qn - 1 are 00 or 11, we simply perform the **arithmetic shift right operation (ashr)** to the partial product AC. And the bits of Qn and Qn - 1 is incremented by 1 bit.
   ii. If the bits of Qn and Qn - 1 is shows to 01, the multiplicand bits (M) will be added to the AC (Accumulator register). After that, we perform the right shift operation to the AC and QR bits by 1.
   iii. If the bits of Qn and Qn - 1 is shows to 10, the multiplicand bits (M) will be subtracted from the AC (Accumulator register). After that, we perform the right shift operation to the AC and QR bits by 1.
6. The operation continuously works till we reached n - 1 bit in the booth algorithm.
7. Results of the Multiplication binary bits will be stored in the AC and QR registers.

Case 1:- When both multiplicand and multiplier are positive

Let, Multiplicand (M) = 7 Multiplier (Q) = 3

In binary, M = 0111 Q = 0011

-M = 2's compliment of M = 1001

Count (n)	*A*	*Q* $Q_3\ Q_2\ Q_1\ Q_0$	Q_{-1}	*Operation*
4	0 0 0 0	0 0 1 1	0	Initialization
	1 0 0 1	0 0 1 1	0	A ←A – M A = 0000 ÷ 1001 1001
	1 1 0 0	1 0 0 1	1	ASR A,Q,Q_{-1}
3	1 1 1 0	0 1 0 0	1	ASR A,Q,Q-1
2	0 1 0 1	0 1 0 0	1	A ←A ÷ M A = 1110 ÷ 0111 X→ 1 0101 Carry discarded
	0 0 1 0	1 0 1 0	0	ASR A,Q,Q-1
1	0 0 0 1	0 1 0 1	0	ASR A,Q,Q-1

Result = 0 0 0 10 1 0 1 = 21

If first bit is 0, then we get the decimal equivalent of the number to get result. But if first bit is 1, then it means result is negative and we have to do 2's complement of number to get the final result.

Hence, result of 7 x 3 = 21

Case 2:- When multiplicand is negative and multiplier is positive

Let, Multiplicand (M) = - 7 => -M = 7

Multiplier (Q) = 3

In binary, -M = 0111 Q = 0011

M = 1001 (2's compliment of -M)

Count (n)	A	Q $Q_3\ Q_2\ Q_1\ Q_0$	Q_{-1}	Operation
4	0 0 0 0	0 0 1 1	0	Initialization
	0 1 1 1	0 0 1 1	0	(i) $A \leftarrow A - M$ A = 0000 + 0111 0111
	0 0 1 1	1 0 0 1	1	(ii) ASR A, Q, Q_{-1}
3	0 0 1 1	1 1 0 0	1	(i) ASR A, Q, Q_{-1}
2	1 0 1 0	1 1 0 0	1	(i) $A \leftarrow A - M$ A = 0000 + 0111 0111
	1 1 0 1	0 1 1 0	0	(ii) ASR A, Q, Q_{-1}
1	1 1 1 0	1 0 1 1	0	ASR A, Q, Q-1

Here 1st bit is 1 which means result is negative.

To get the magnitude we have to find 2's complement of **1 1 1 0 1 0 1 1**

2's complement = 0 0 0 1 0 1 0 0

\+ 1

0 0 0 1 0 1 0 1 = 21(in decimal)

Hence, **result of -7 x 3 = -21**

Case 3:- When multiplicand is positive and multiplier is negative

Let, Multiplicand (M) = 7 Multiplier (Q) = - 3

In binary, M = 0111 -Q = 0011 Q = 1101 (2's compliment of -Q)

Count (n)	A	Q $Q_3\ Q_2\ Q_1\ Q_0$	Q_{-1}	Operation
4	0 0 0 0	1 1 0 1	0	Initialization
	1 0 0 1	1 1 0 1	0	(i) $A \leftarrow A - M$ A = 0000 + 1001 1001
	1 1 0 0	1 1 1 0	1	(ii) ASR A, Q, Q_{-1}
3	0 0 1 1	1 1 1 0	1	(i) $A \leftarrow A + M$ A= 1100 + 0111 1 0011 Carry discarded
	0 0 0 1	1 1 1 1	0	(ii) ASR A, Q, Q_{-1}
2	1 0 1 0	1 1 1 1	0	(i) $A \leftarrow A - M$ A = 0001 + 1001 1010
	1 1 0 1	0 1 1 1	1	(ii) ASR A, Q, Q_{-1}
1	1 1 1 0	1 0 1 1	1	ASR A, Q, Q-1

Here 1st bit is 1 which means result is negative.

To get the magnitude we have to find 2's complement of 1 1 1 0 1 0 1 1

2's complement = 0 0 0 1 0 1 0 0

\+ 1

0 0 0 1 0 1 0 1 = 21(in decimal)

Hence, result of -7 x 3 = -21

Case 4:- When both multiplicand and multiplier are negative

Let, Multiplicand (M) = -7 Multiplier (Q) = -3

In binary, -M = 0111&M = 1001 (2's compliment of -M)

-Q = 0011& Q = 1101 (2's compliment of -Q)

Count (n)	A	Q $Q_3\ Q_2\ Q_1\ Q_0$	Q_{-1}	Operation
4	0 0 0 0	1 1 0 1	0	Initialization
	0 1 1 1	1 1 0 1	0	(i) $A \leftarrow A - M$ A = 0000 + 0111 0111
	0 0 1 1	1 1 1 0	1	(ii) ASR A, Q, Q_{-1}
3	1 1 0 0	1 1 1 0	1	(i) $A \leftarrow A + M$ A= 0011 + 1001 1100
	1 1 1 0	0 1 1 1	0	(ii) ASR A, Q, Q_{-1}
2	0 1 0 1	0 1 1 1	0	(i) $A \leftarrow A - M$ A = 1110 + 0111 1 0101 Carry discarded
	0 0 1 0	1 0 1 1	1	(ii) ASR A, Q, Q_{-1}
1	0 0 0 1	0 1 0 1	1	ASR A, Q, Q-1

Result = 0 0 0 1 0 1 0 1

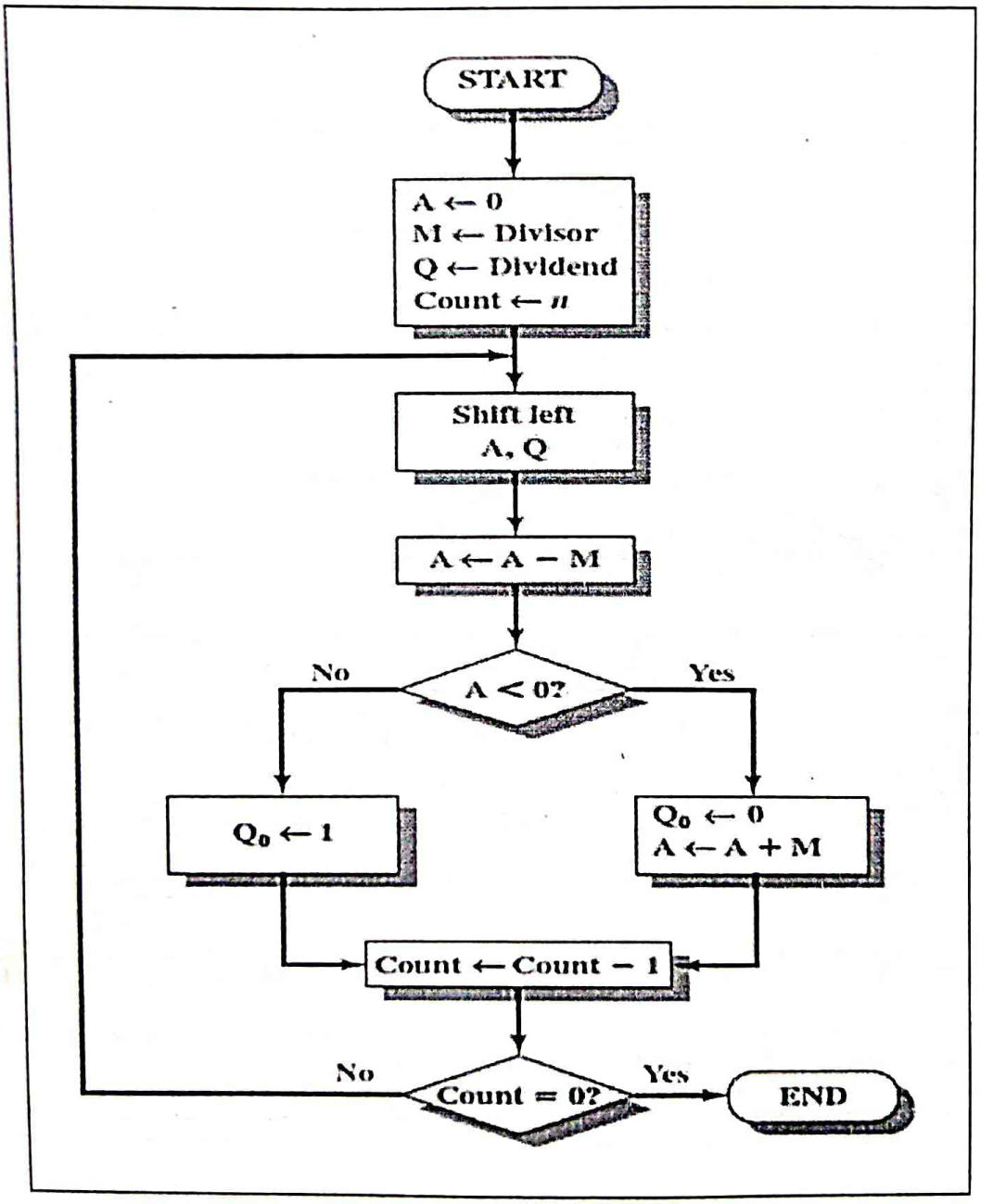
16 8 4 2 1 = 21

Here 1st bit is 0 which means result is positive.

Hence, result of -7 x -3 = 21.

BOOTH'S RESTORING DIVISION ALGORITHM

In a division algorithm there is a quotient and a remainder when we divide two number. Here, n-bit dividend is loaded in Q and divisor is loaded in M. **Value of Register is initially kept 0 and this is the register whose value is restored during iteration due to which it is named Restoring.** We are using restoring term because we know that the value of register A will be restored after each iteration. We will also try to solve this problem using the flow chart and apply bit operations. Here, register Q is used to contain the quotient, and register A is used to contain the remainder.



Example: 7/3 Q = Dividend = 0111

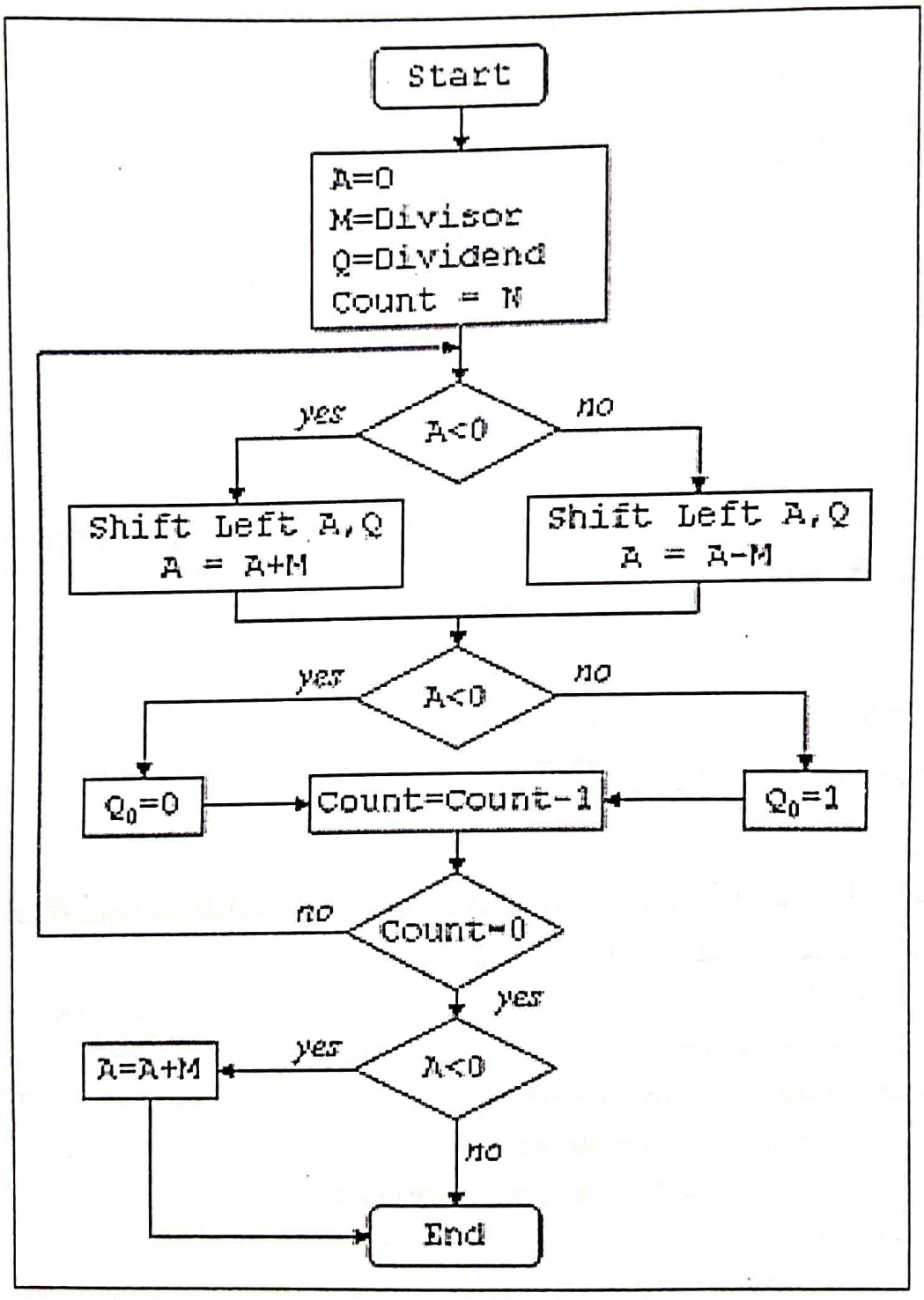
M = Divisor = 0011

-M=1101

A	Q	M	Count	Operation	
	$Q_3\ Q_2\ Q_1\ Q_0$				
0 0 0 0	0 1 1 1	0 0 1 1	4	Initialization	
0 0 0 0	1 1 1 0	0 0 1 1		Shift Left A, Q	
1 1 0 1	1 1 1 0	0 0 1 1		A=A-M	Restore
0 0 0 0	1 1 1 0	0 0 1 1	3	A=A+M	
0 0 0 1	1 1 0 0	0 0 1 1		Shift Left A, Q	
1 1 1 0	1 1 0 0	0 0 1 1		A=A-M	Restore
0 0 0 1	1 1 0 0	0 0 1 1	2	A=A+M	
0 0 1 1	1 0 0 0	0 0 1 1		Shift Left A, Q	
0 0 0 0	1 0 0 0	0 0 1 1		A=A-M	
0 0 0 0	1 0 0 1	0 0 1 1	1	$Q_0 = 1$	
0 0 0 1	0 0 1 0	0 0 1 1		Shift Left A, Q	
1 1 1 0	0 0 1 0	0 0 1 1		A=A-M	
0 0 0 1	0 0 1 0	0 0 1 1	0	A=A+M	Restore
↑ Remainder	↑ Quotient				

BOOTH'S NON-RESTORING DIVISION ALGORITHM

The non-restoring division algorithm is more complex as compared to the restoring division algorithm. But when we implement this algorithm in hardware, it has an advantage, i.e., it **contains only one decision and addition/subtraction per quotient bit.**



Example: 8/3 Q = Dividend = 1000 M = Divisor = 0011 -M=1101

A	Q	M	Count	Operation
	$Q_3 Q_2 Q_1 Q_0$			
0 0 0 0	1 0 0 0	0 0 1 1	4	Initialization
0 0 0 1	0 0 0 0	0 0 1 1		Shift Left A, Q
1 1 1 0	0 0 0 0	0 0 1 1		A=A-M
1 1 1 0	0 0 0 0	0 0 1 1	3	$Q_0 = 0$
1 1 0 0	0 0 0 0	0 0 1 1		Shift Left A, Q
1 1 1 1	0 0 0 0	0 0 1 1		A=A+M
1 1 1 1	0 0 0 0	0 0 1 1	2	$Q_0 = 0$
1 1 1 0	0 0 0 0	0 0 1 1		Shift Left A, Q
0 0 0 1	0 0 0 0	0 0 1 1		A=A+M
0 0 0 1	0 0 0 1	0 0 1 1	1	$Q_0 = 1$
0 0 1 0	0 0 1 0	0 0 1 1		Shift Left A, Q
1 1 1 1	0 0 1 0	0 0 1 1		A=A-M
1 1 1 1	0 0 1 0	0 0 1 1	0	$Q_0 = 0$
0 0 1 0	0 0 1 0	0 0 1 1		A=A+M
↑ Remainder	↑ Quotient			

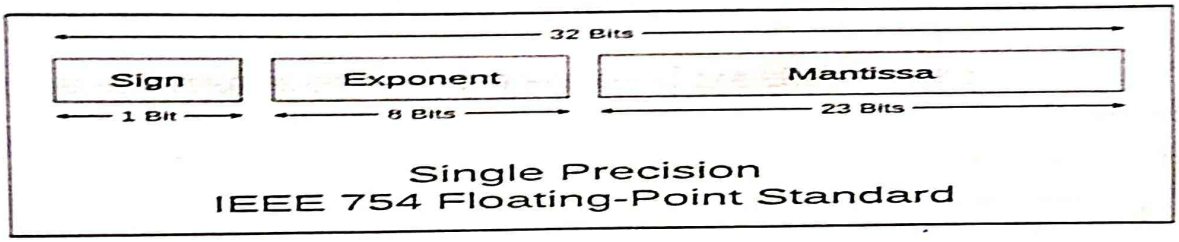
FLOATING POINT REPRESENTATION WITH IEEE STANDARDS AND ITS ARITHMETIC OPERATIONS

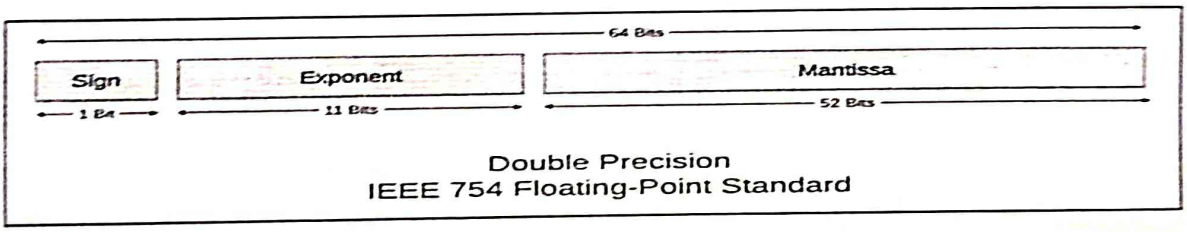
The IEEE Standard for Floating-Point Arithmetic (IEEE 754) is a technical standard for floating-point computation which was established in 1985 by the Institute of Electrical and Electronics Engineers (IEEE). The standard addressed many problems found in the diverse floating point implementations that made them difficult to use reliably and reduced their portability. IEEE Standard 754 floating point is the most common representation today for real numbers on computers, including Intel-based PC's, Macs, and most Unix platforms.

There are several ways to represent floating point number but IEEE 754 is the most efficient in most cases. IEEE 754 has 3 basic components:

1. **The Sign of Mantissa –**
   This is as simple as the name. 0 represents a positive number while 1 represents a negative number.
2. **The Biased exponent –**
   The exponent field needs to represent both positive and negative exponents. A bias is added to the actual exponent in order to get the stored exponent.
3. **The Normalized Mantissa –**
   The mantissa is part of a number in scientific notation or a floating-point number, consisting of its significant digits. Here we have only 2 digits, i.e. O and 1. So a normalized mantissa is one with only one 1 to the left of the decimal.

IEEE 754 numbers are divided into two based on the above three components: single precision and double precision.





Numerical 1: Represent $(1259.125)_{10}$ in single and double precision format.

Step1: Convert decimal to binary.

$(1259)_{10} = (10011101011)_2$ $(.125)_{10} = (001)_2$

$(1259.125)_{10} = (10011101011.001)_2$

Step2: Normalize the number

Single Precision: $(1.N)2^{E-127}$

Double Precision: $(1.N)2^{E-1023}$

$(10011101011.001)_2 = 1.0011101011001*2^{10}$

Step3: Single Precision format (32 bits)

$(1.N)2^{E-127} = 1.0011101011001*2^{10}$

E-127=10

$E=137=(1001001)_2$

0	1001001	0011101011001.....
Sign Bit (1 bit)	Exponent (8 bits)	Mantissa (23 bits)

Step4: Double Precision format (64 bits)

$(1.N)2^{E-1023} = 1.0011101011001*2^{10}$

E-1023=10

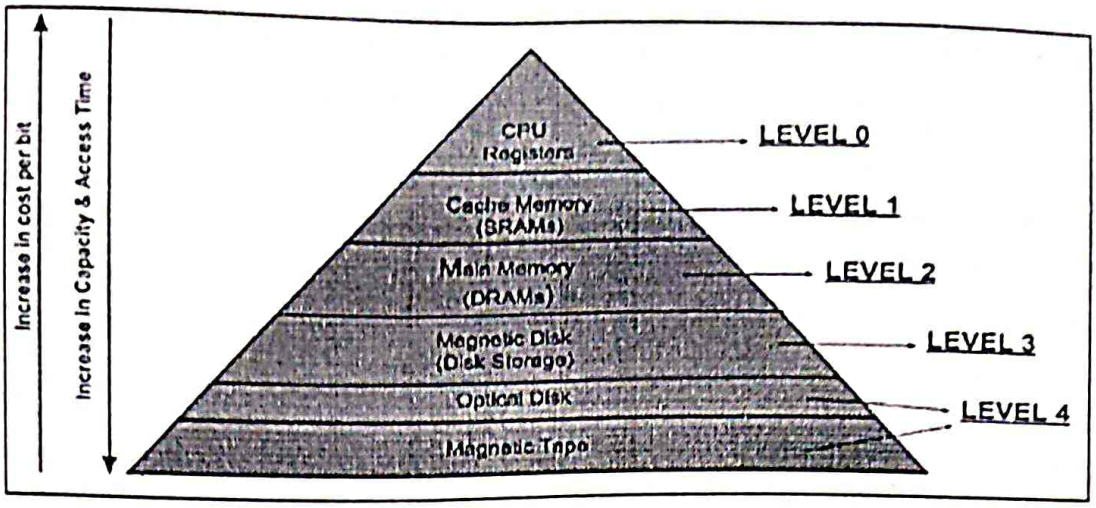
$E=1033=(10000001001)_2$

0	10000001001	0011101011001.....
Sign Bit (1 bit)	Exponent (11 bits)	Mantissa (52 bits)

MEMORY ORGANIZATION: MEMORY HIERARCHY →

Memory Hierarchy is a feature in computer system design that helps to organize memory such that access time is reduced. A processor, as well as a huge number of memory devices, were used in the computer system architecture. However, the fundamental issue is that these components are costly. As a result, memory hierarchy can be used to organize the system's memory. It has many memory levels with varying performance rates. However, any of these can provide a precise objective, reducing the access time. The memory hierarchy was created in response to the program's activity. The memory in a computer can be divided into hierarchy based on 03 key characteristics of memory.

1. Capacity **2. Access time** **3. Cost**



- The memory unit is used for storing programs and data. It fulfills the need for storage of information.
- The additional storage with main memory capacity enhances the performance of the general-purpose computers and makes them efficient.
- Only those programs and data, which are currently needed by the processor, reside in the main memory. Information can be transferred from auxiliary memory to main memory when needed.

A variety of technologies are used to implement memory systems, and across this spectrum of technologies the following relationship hold:

- Faster access time, greater cost per bit.
- Greater capacity, smaller cost per bit.
- Greater Capacity, slower access time.

As one goes down the hierarchy, the following occur :

a. Decreasing cost per bit

b. Increasing capacity

C. Increasing access time

d. Decreasing frequency of access time of the memory by the processor

Thus, smaller, more expensive, faster memories are supplemented by larger, cheaper, slower memories.

1. PRIMARY MEMORY → It is also known as Internal memory, and this is access by the processor straightly. This memory includes main, cache and CPU registers. Primary memory is the computer memory that is directly accessible by CPU. It provides the actual working space to the processor. It holds the data and instructions that the processor is currently

working on. The earliest memory devices were electro-mechanical switches, or relays (*computers: The first computer*), and electron tubes (*computers: The first stored-program machines*). In the late 1940s the first stored-program computers used ultrasonic waves in tubes of mercury or charges in special electron tubes as main memory. The latter were the first random-access memory (RAM). RAM contains storage cells that can be accessed directly for read and write operations, as opposed to serial access memory, such as magnetic tape, in which each cell in sequence must be accessed till the required cell is located.

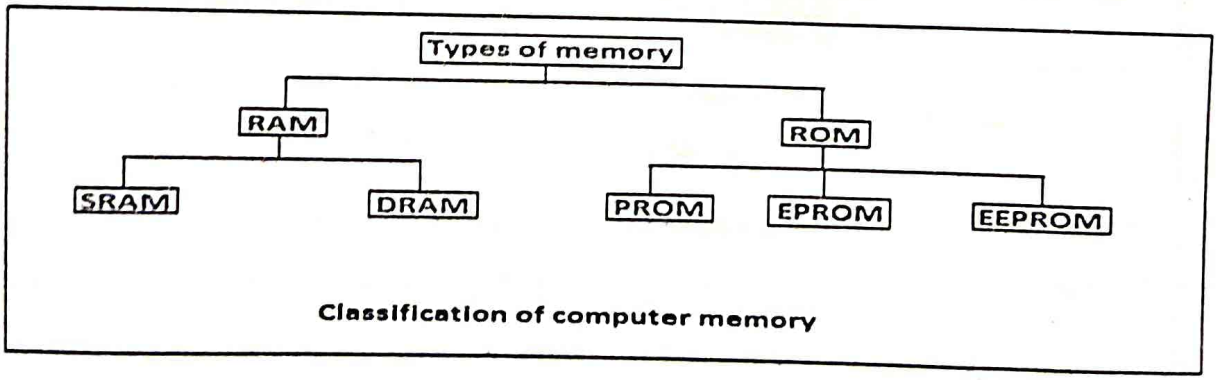
Need of Primary Memory:

In order to enhance the efficiency of the system, memory is organized in such a way that access time for the ready process is minimized. The following approach is followed to minimize access time for the ready process.

- All programs, files, and data are stored in secondary storage that is larger and hence has greater access time.
- Secondary memory cannot be accessed directly by a CPU or processor.
- In order, to execute any process operating system loads the process in primary memory which is smaller and can be accessed directly by the CPU.
- Since only those processes are loaded in primary memory which is ready to be executed, the CPU can access those processes efficiently and this optimizes the performance of the system.

Classification of Primary Memory

Primary memory can be broadly classified into two parts:



Classification of computer memory

Read-Only Memory:

Any data which need not be altered are stored in ROM. ROM includes those programs which run on booting of the system *(know as* a *bootstrap program* that initializes *OS)* along

with data like algorithm required by OS. Anything stored in ROM cannot be altered or changed.

Types of ROM:

ROM can be broadly classified into 4 types based on their behavior:

- **MROM:** *Masked ROM* are hardwired and pre-programmed ROM. Any content that is once written cannot be altered anyhow.
- **PROM:** *Programmable ROM* can be modified once by the user. The user buys a blank PROM and writes the desired content but once written content cannot be altered.
- **EPROM:** *Erasable and Programmable ROM* Content can be changed by erasing the initial content which can be done by exposing EPROM to UV radiation. This exposure to ultra-violet light dissipates the charge on ROM and content can be rewritten on it.
- **EEPROM:** *Electrically Erasable and Programmable ROM* Content can be changed by erasing the initial content which could be easily erased electrically. However, one byte can be erased at a time instead of deleting in one go. Hence, reprogramming of EEPROM is a slow process.

Random Access Memory (RAM)

Any process in the system which needs to be executed is loaded in RAM which is processed by the CPU as per Instructions in the program. Like if we click on applications like Browser, firstly browser code will be loaded by the Operating system into the RAM after which the CPU will execute and open up the Browser.

Types of RAM:

RAM can be broadly classified into SRAM (Static RAM) and DRAM (Dynamic RAM) based on their behavior:

- **DRAM:** Dynamic RAM or DRAM needs to periodically refresh in few milliseconds to retain data. DRAM is made up of capacitors and transistors and electric charge leaks from capacitors and DRAM needs to be charged periodically. DRAM is widely used in home PCs and servers as it is cheaper than SRAM.
- **SRAM:** Static RAM or SRAM keeps the data as long as power is supplied to the system. SRAM uses Sequential circuits like a flip-flop to store a bit and hence need not be periodically refreshed. SRAM is expensive and hence only used where speed is the utmost priority.

Primary Memory is volatile in nature?

Content of primary memory may or may not vanish when power is lost depending on if it is stored in RAM or ROM.

- Content of ROM is non-volatile in nature, they are stored even when power is lost.
- Content of RAM is volatile in nature, it vanishes when power is lost.

2. AUXILIARY MEMORY/SECONDARY MEMORY :-

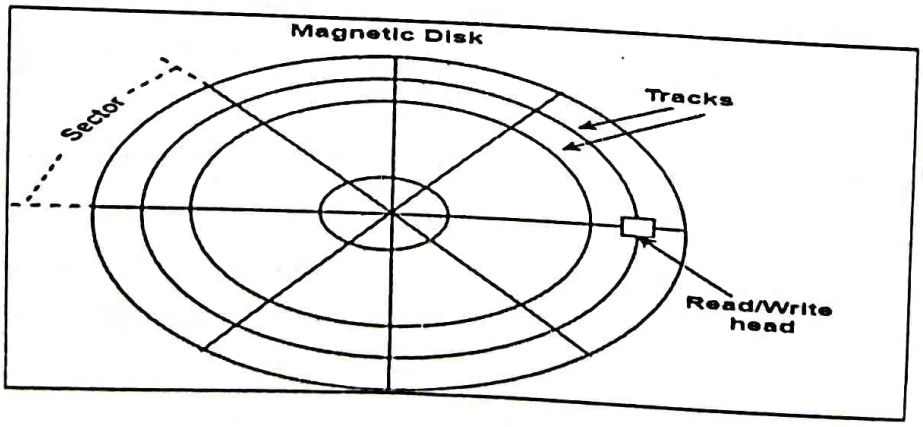
An Auxiliary memory is referred to as the lowest-cost, highest-space, and slowest-approach storage in a computer system. It is where programs and information are preserved for long-term storage or when not in direct use. The most typical auxiliary memory devices used in computer systems are magnetic disks and tapes. The contents of the secondary memory first get transferred to the primary memory and then are accessed by the processor, this is because the processor does not directly interact with the secondary memory. This is also known as external memory, and this is accessible by the processor through an Input/output module.

Magnetic Disks

A magnetic disk is a round plate generated of metal or plastic coated with magnetized material. There are both sides of the disk are used and multiple disks can be stacked on one spindle with read/write heads accessible on each surface.

All disks revolve together at high speed and are not stopped or initiated for access purposes. Bits are saved in the magnetized surface in marks along concentric circles known as tracks. The tracks are frequently divided into areas known as sectors.

In this system, the lowest quantity of data that can be sent is a sector. The subdivision of one disk surface into tracks and sectors is displayed in the figure.



Magnetic Tape

Magnetic tape transport includes the robotic, mechanical, and electronic components to support the methods and control structure for a magnetic tape unit. The tape is a layer of plastic coated with a magnetic documentation medium.

Bits are listed as a magnetic stain on the tape along various tracks. There are seven or nine bits are recorded together to form a character together with a parity bit. Read/write heads are mounted one in each track therefore that information can be recorded and read as a series of characters.

Magnetic tape units can be stopped, initiated to move forward, or in the opposite, or it can be reversed. However, they cannot be initiated or stopped fast enough between single characters. For this reason, data is recorded in blocks defined as records. Gaps of unrecorded tape are added between records where the tape can be stopped.

The tape begins affecting while in a gap and achieves its permanent speed by the time it arrives at the next record. Each record on tape has a recognition bit design at the starting and end. By reading the bit design at the starting, the tape control recognizes the data number.

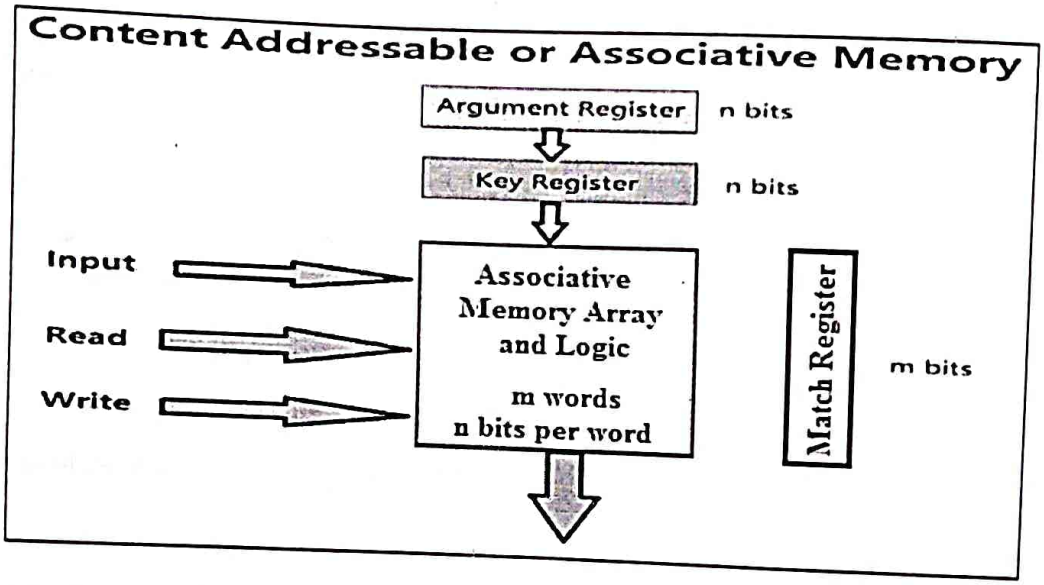
Now, Let's see the difference between Primary memory and Secondary memory:

Primary memory	Secondary memory
Primary memory is temporary.	Secondary memory is permanent.
Primary memory is directly accessible by Processor/CPU.	Secondary memory is not directly accessible by the CPU.
Nature of Parts of Primary memory varies, RAM- volatile in nature. ROM- Non-volatile.	It's always Non-volatile in nature.
Primary memory devices are more expensive than secondary storage devices.	Secondary memory devices are less expensive when compared to primary memory devices.
The memory devices used for primary memory are semiconductor memories.	The secondary memory devices are magnetic and optical memories.
Primary memory is also known as Main	Secondary memory is also known as

Primary memory	Secondary memory
memory or Internal memory.	External memory or Auxiliary memory.
Examples: RAM, ROM, Cache memory, PROM, EPROM, Registers, etc.	Examples: Hard Disk, Floppy Disk, Magnetic Tapes, etc.

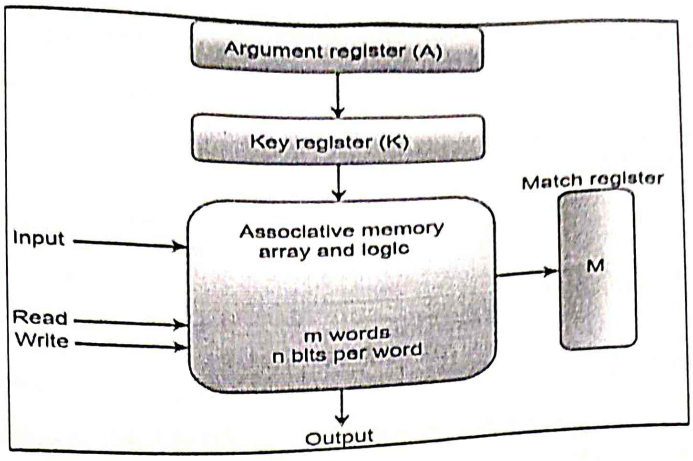
ASSOCIATIVE MEMORY:

The number of access to memory depends on the location of item and the efficiency of search algorithm. Many search algorithms have been used to minimize the number of accesses while searching for an item in random or sequential access memory. The time required to find an item stored in memory can be reduced considerably if stored data can be identified for access by the content of data itself rather than by the address. A memory unit accessed by content is called Associative memory or content addressable memory (CAM).



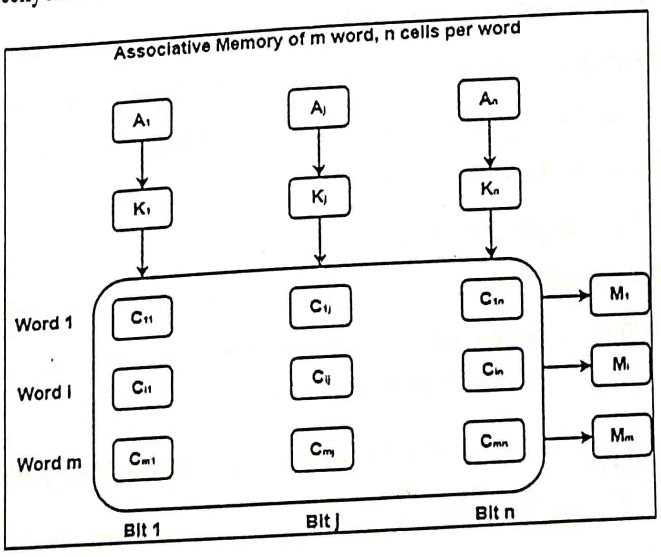
Hardware organization → The block diagram of associative memory consists of a memory array and logic for **m words** with **n bits** per word. The **Argument Register (A)** and **Key Register (K)** have n bits, one for each bit of a word. The **Match Register M** has **m bits**, one for each memory word. Each word in memory is compared in parallel with the content of **Argument Register**. The word that match the bits of arguments register set a corresponding bit in the match register.

After the matching process, those bits in match register that have been set indicate to one. Then reading is accomplished by a sequential access to memory for those words whose corresponding bits in the match register have been set to 1.



A	101	111100	
K	111	000000	
Word 1	100	111100	no match
Word 2	101	000001	match

The Key Register provide a mask for choosing a particular field or Key in the Argument word. The entire argument is compared with each memory word if the key register contains all 1's. Otherwise, only those bits in the argument register that have 1's in their corresponding position of the a Key register are compared. The key provides mask or identifying piece of information which specify how the reference to memory is made.



The cells in the array are considered by the letter C with two subscripts. The first subscript provides the word number and the second determines the bit position in the word. Therefore cell C_{ij} is the cell for bit j in word i.

A bit in the argument register is compared with all the bits in column j of the array supported that $K_j = 1$. This is completed for all columns $j = 1, 2 \ldots, n$.

If a match appears between all the unmasked bits of the argument and the bits in word i, the equivalent bit M_i in the match register is set to 1. If one or more unmasked bits of the argument and the word do not match, M_i is cleared to 0.

CACHE MEMORY

Analysis of a large number of typical programs has shown that the reference to memory at any given interval of time tend to be confined a few localized areas in memory. **This phenomenon is known as the property of Reference Property of locality of Reference.** If the active portion of the program and data are placed in a fast small memory, the average memory access time can be reduced . Such a fast small memory is referred to as a **cache memory.**

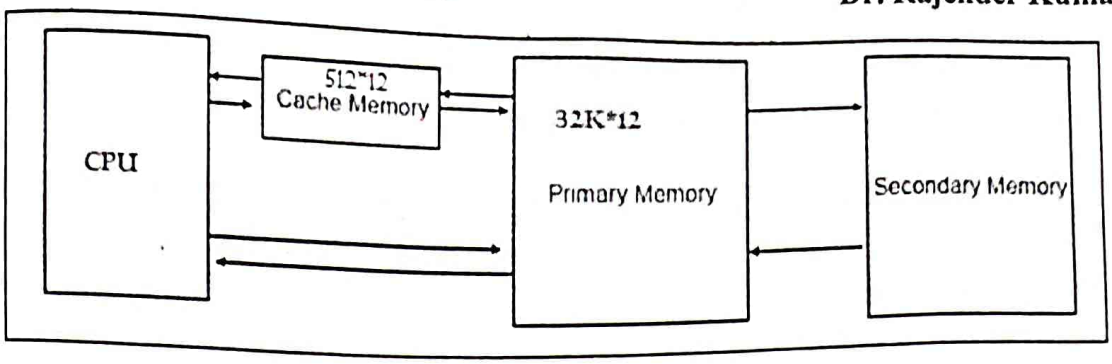
The Cache memory access time is less than the access time of main memory by a factor of 5 to 10. The Cache is the fastest Component in the memory hierarchy. The basic operation of the cache is that when the CPU needs to access memory, the cache is examined. If the word found in cache, it is read from the fast memory.

The performance of cache memory is frequently measured in terms of a quantity, called **hit ratio.** When the CPU refers to memory and find the word in cache, it is said to be produce a hit.

- If the word is not found in cache, it is main memory and it in counts as a miss. The ratio of the number of hits divided by the total CPU references to memory **(hit plus miss)** is the hit ratio. Hit ratio of 0.9 and higher have been reported. **This high ratio verifies the validity of the locality of reference property.**
- The basic characteristic of Cache memory is its fast access time. Therefore, very little or no time very must be wasted when searching for words in the cache. The transformation of data from main memory to cache memory is referred to as **mapping process.**

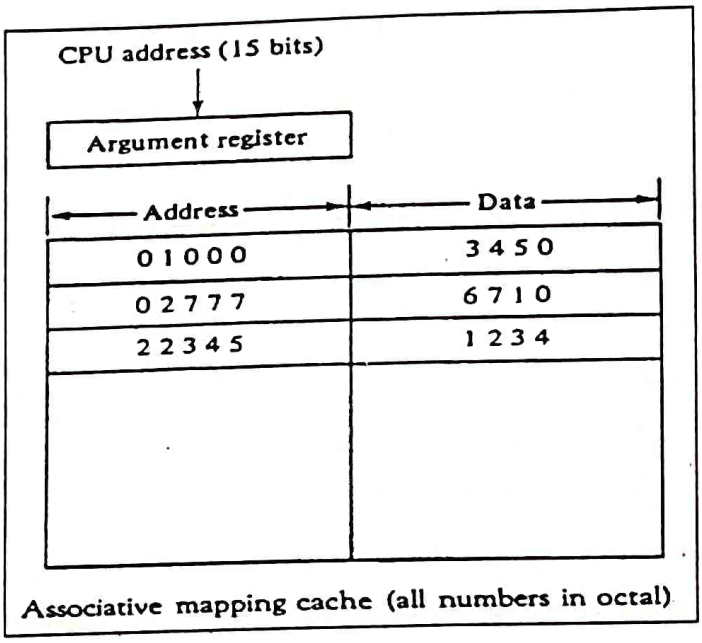
Three types of mapping procedures are of practical interest when considering the organization of Cache memory:

1. Associative mapping
2. Direct mapping
3. Set -associative mapping



The main can store 32k words of 12 bits each. The cache is capable of storing 512 of these woods any given time. For every word stored in cache, there is a. duplicate copy in main memory. The CPU Communicate with both memories. It first sends a 15-bit address to cache. If there is a hit, the CPU accepts the 12 bits data from cache. If there is a miss, the CPU reads the word from main memory and the word is then transferred to cache.

1. **Associative mapping:** → The fastest and most flexible cache organization uses an associative memory. The associative memory stores both the address and content (data) of memory word. This permits any location in cache to store any word from main memory.



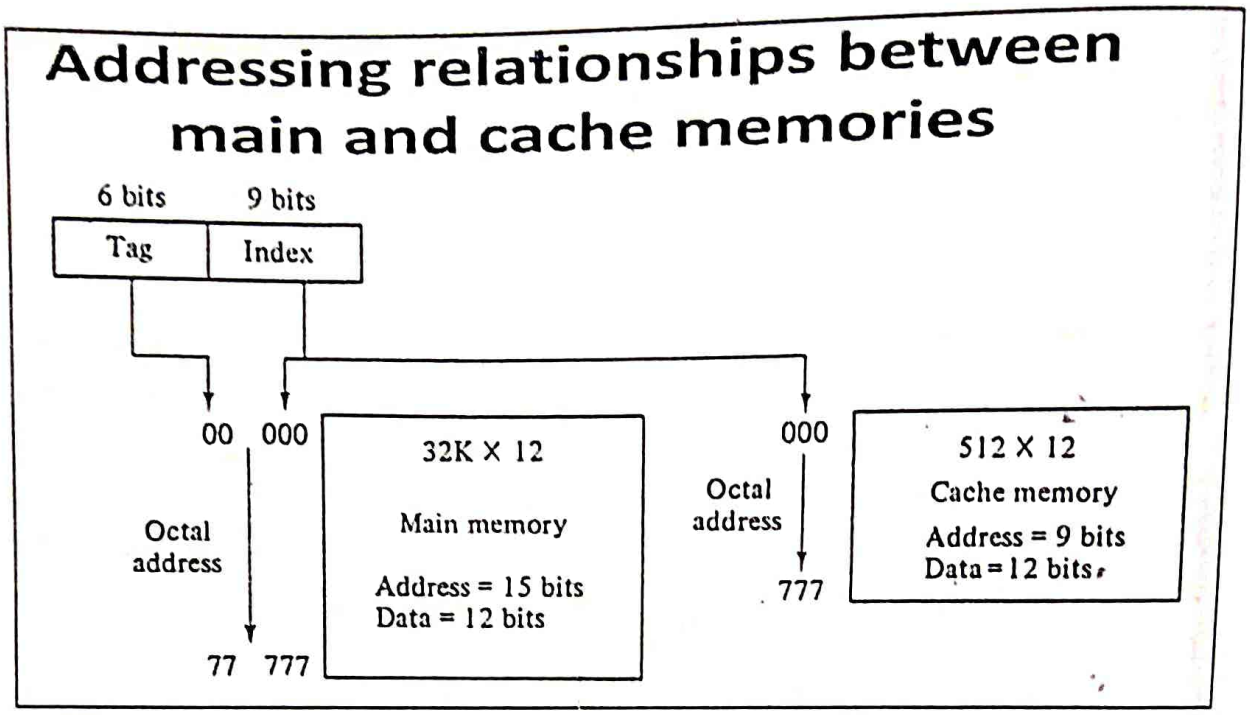
Associative mapping cache (all numbers in octal)

The address values of 15 bits is shown as a 5-digit octal number and its corresponding 12-bit word is shown as 4-bit octal number. A CPU address of 15-bits is placed in the augment register and associative memory is searched for a matching address.

If the address is found, data is read and sent the corresponding 12-bit data is read and sent to the CPU, If no match occurs, the main memory is searched and accessed for the word. The address-data pair is then transferred to the associative cache memory. If the cache is full, an address-data pail must be displaced to make room for a pair that is needed and not presently in

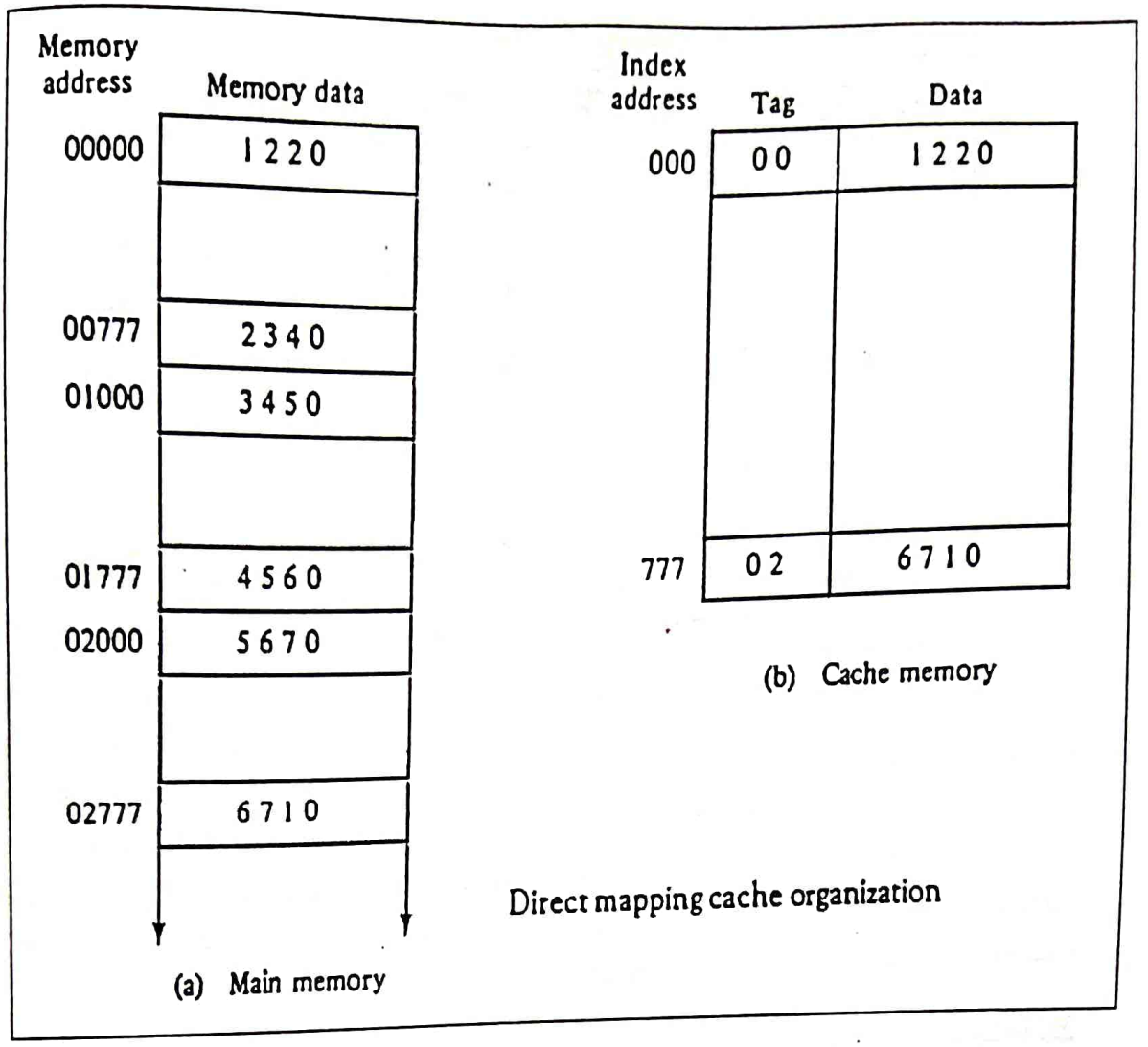
the cache. The decision as to what pair is replaced is determined from the replacement algorithm that the designer chooses for the cache.

2. **Direct mapping :** → Associative memories are expensive as compared to random- access memories because the logic added with each cell. In direct mapping, the CPU address of 15-bits is divided into 2-fields. The 09 least significant bits constitutes the **Index field** and remaining 06-bits formed the **tag field.**



The number of bits in the index field is equal to number of address bits required to access the cache memory. In the general case, there are 2^k **words** in cache and 2^n **words** in main memory. The n-bit memory address is divided into **02-fields K bits for the index field** and **n-k bits for the tag field.** The direct mapping cache organization uses the n-bit address to access the main memory and the k-bit index to access the cache.

To see how the direct mapping organization operates, consider the numerical example. The word at address **ZERO** is presently stores in the cache (index=000, tag=00, data =1220). Suppose that the CPU now wants to access the word at address 02000. The address is 000, So it is used to access the cache. The 02 tags are then compared.



The cache tag is 000 but the address tag is 02, which does not produce a match. Therefore, the main memory is accessed and data word 5670 is transferred to the CPU. The cache word at index address 000 is then replaced with a tag of 02 and data of 5670.

3. **Set-Associative mapping :** → The disadvantage of direct mapping is that 02-words with the same index in their address but with different tag values cannot reside in cache memory at the same time. A **third type of cache organization, called set-associative mapping,** is an improvement over the direct mapping organization in that each word of cache can store 02 or more words of memory under the same index address.

Index	Tag	Data	Tag	Data
000	01	3450	02	5670
777	02	6710	00	2340

Two-way set-associative mapping cache

The words stored at addresses 01000 and 02000 of main memory are stored in cache memory at index address 000. Similarly, the words at addresses 02777 and 00777 stored in cache at index address 777. The **comparison logic** is done by an **associative search** of the tags in the set similar to an **associative memory search**: thus the name "**set-associative**". When a miss occurs in a set-associative cache and set is full, it is necessary to replace one of tag-data items with a new value. The most common replacement algorithms are: random, FIFO, LRU.

Writing into cache → An important aspect of cache is concerned with memory write requests. When the CPU finds a word in cache during a read operation, the main memory is not involved in the transfer. However, if the operation is a write, there are 02 ways that the system can proceed.

1. Write through → The simplest and most commonly used procedure is to update main memory with every memory write operation, with cache being updated in parallel if it contains the word at the specified address. This is called the write through method. This method has the advantage that main memory always contains the same data as the cache.

2. Write-back → The second procedure is known as write-back method. In this method only the cache updated during a write operation. The location is then marked by a flag so that later when the word is removed from the cache it is copied into main memory.

Numerical. The access time of **Cache memory** is **100 ns** and that of **main memory 1000 ns**. It is estimated that 80 percent of memory requests are for read and remaining 20 percent for write. The hit ratio for read access is 0.9. A write-through procedure is used.

(i) what is the average access time considering only read cycle?

(ii) what is average access time of system for both read and write requests?

(iii) what is hit ratio taking into consideration the write cycle?

Answer (i) only for read operations,

effective read time = (hit ratio for cache) (cache access time) +(cache miss time) (cache time + M.M time)

= 0.9*100+ (1 - 0.9) (100 + 1000)

= 0.9 *100 + (0.1) (1100)

= 90 + 110=200 ns

Answer (ii) Similarly, only for write time = 1000 ns (because it's the write through cache and largest of cache and M.M time is considered)

→ Effective access time for read and write ⇒ (Probability of read operation) * (effective read operation time) +(Probability of write operation) * (effective write operation time)

= 0.8 * 200+ 0.2*1000 = 160+ 200=360 ns

Answer (iii) Hit ratio considering write cycle =

= 0·9*80 + 1*20 (since there will be hit on write)

=72+20= 92%

VIRTUAL MEMORY:

Virtual memory is a concept used in some large computer systems that permit the user to construct programs as though a large memory space were available, equal to the totality of auxiliary memory. Each address that is referenced by the CPU goes through an address mapping from the so-called virtual address to a physical address in main memory. Virtual memory is used to give programmers the illusion that they have a very large memory at their disposal, even though computer actually has small main memory.

A virtual memory system provides a mechanism for translating program-generated address into correct main memory locations. This is done dynamically, while programs are being executed in the CPU.

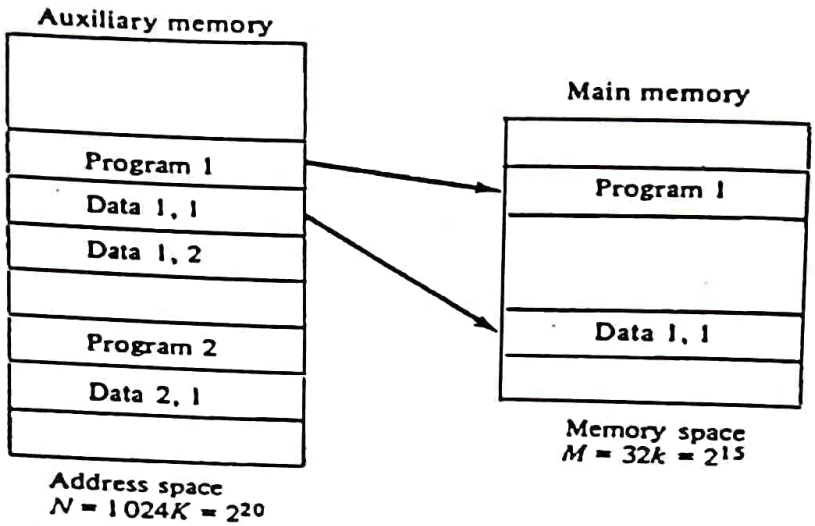
Address space and Memory space →

An address used by a programmer will be called a **virtual address**, and the **set of such addresses as address space**. An address in main memory is called a location or **physical**

address. The set of such locations is called the memory space. In most computers, the address and memory spaces are identical. The address space is allowed to be larger than the memory space in computer with virtual memory.

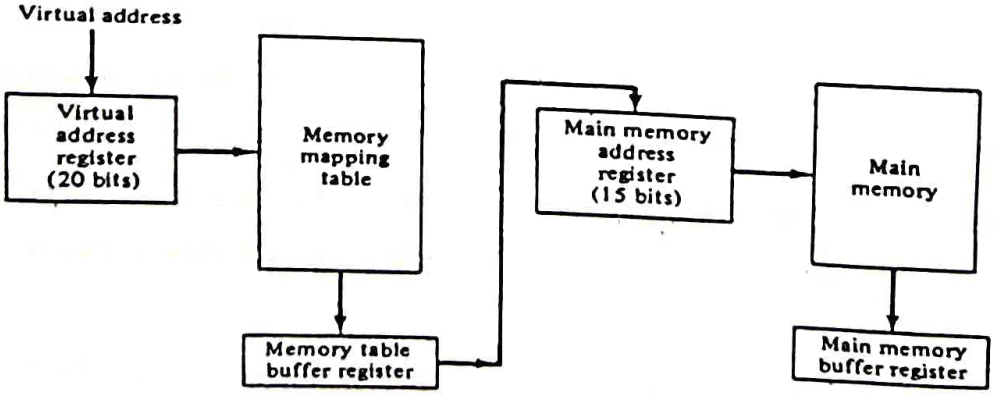
As an example, consider a computer with a main memory capacity of 32K words (k=1024). 15 bits are needed to specify a physical address in memory since $32k = 2^{15}$. Suppose that the computer has available auxiliary memory for storing 2^{20}=1024k words. Thus, auxiliary memory has a capacity for storing information equivalent to the capacity of 32 main memories.

Denoting the address space by N and the memory space by M, we then have an example of N=1024k and M=32k.



Relation between address and memory space in a virtual memory system

A table a is needed to map a virtual address of 20 bits to a physical address of 15 bits. The mapping is dynamic operation, which means that every address is by translated immediately as a word is referenced by CPU.

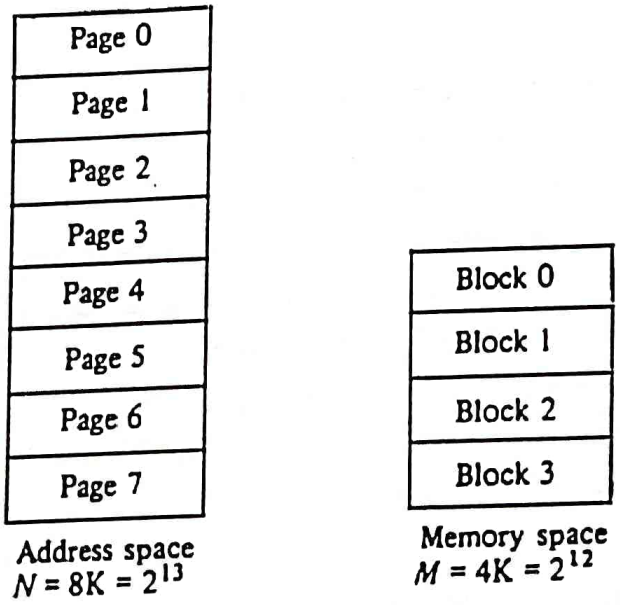


Memory table for mapping a virtual address

The mapping table may be stored ma separate memory or the main memory. In the first case, an additional memory unit is required as well as one extra memory access time. In the second case, the table takes space from main memory and two accesses to memory are required with the program running at half speed. A third alternative is to use an associative memory.

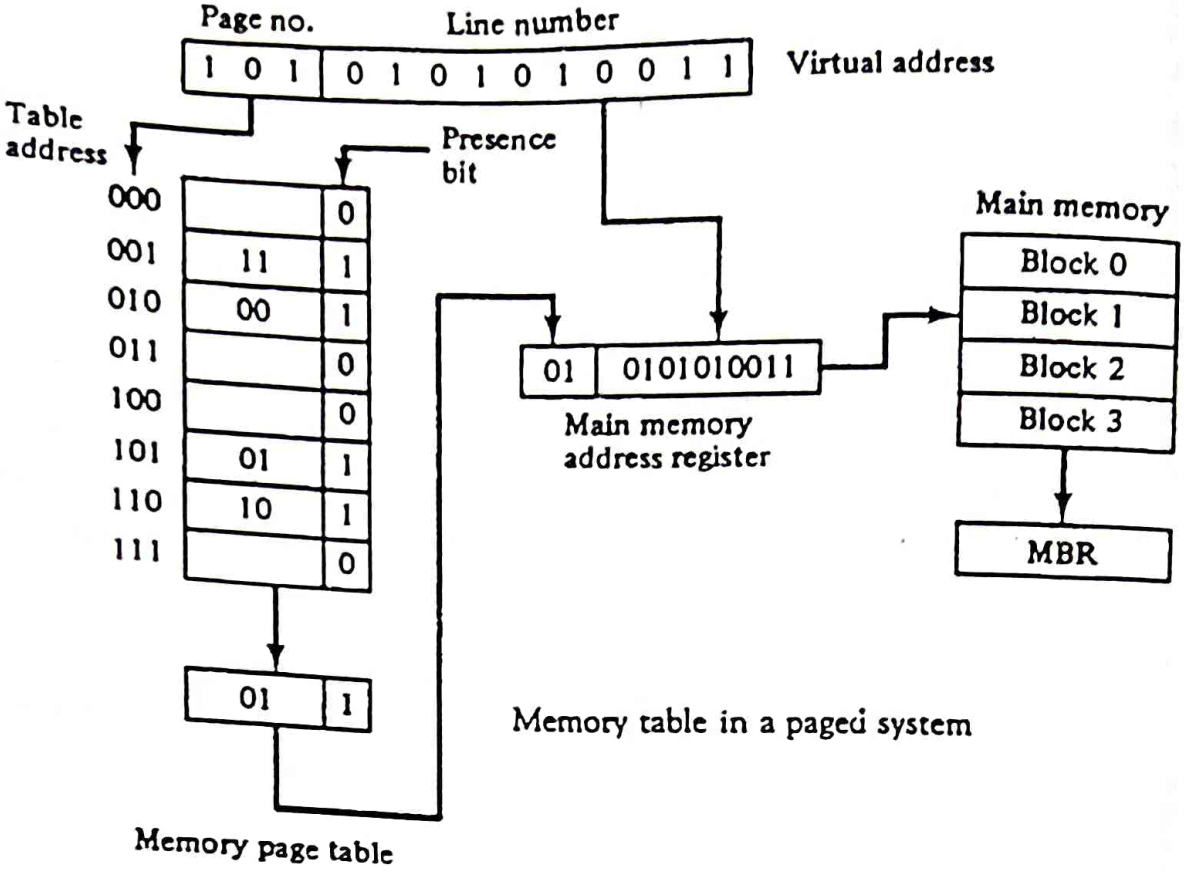
Address mapping using Pages → The table implementation of the address mapping is simplified if the information in the address space and the memory space are each divided into groups of fixed size. The physical memory broken down into **groups of equal size is called blocks**, which may range from 64 to 4096 words each. The term **page refers to group of address space of same size**. The term "**page frame**" is sometimes used to denote a **block.**

Consider a computer with an **address space of 8k** and a **memory space of 4k.** If we split each into of **1k words**, we **obtain 8 pages and 4 blocks.** At any given time up to 4 pages of **address space** may reside in main memory in any one of the **04 Blocks.**



Address space and memory space split into groups of 1K words

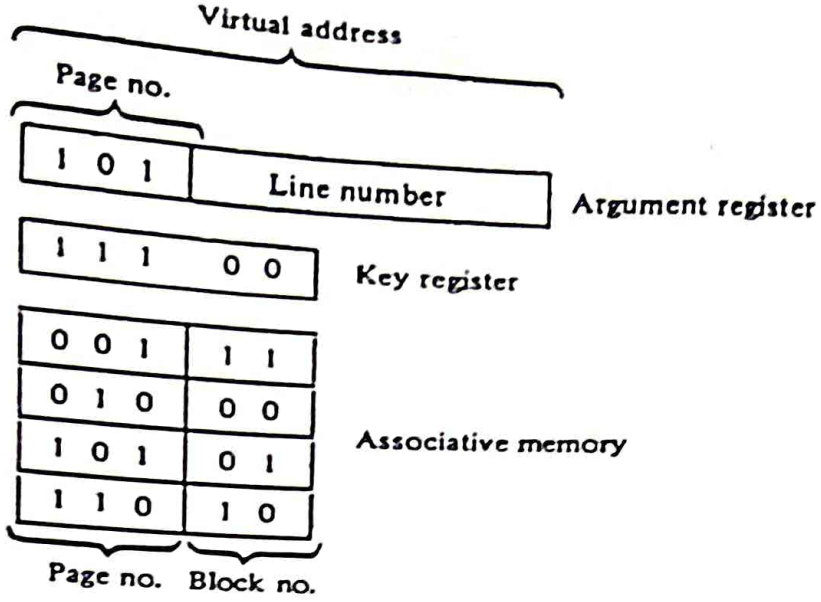
The **mapping from address space to memory space** is facilitated if each virtual address is considered to be represented by **02 numbers: a page number address** and a **line within the page.** In a compete with 2^P words per page, p bits are used to specify a line address and the remaining high order bits of the virtual address specify the page number.



Memory table in a paged system

The Memory-Page table consists of 08 words, one for each page. The address in the page table denotes the page number and the contents of the words gives black number where that page is shared in main memory. The table shows that pages. 1, 2, 5 and 6 are now available in main memory in blocks 3, 0, 1 and 2 respectively. A presence bit in each location indicates whether the page has been transferred from auxiliary memory into main memory. A, 0 in the presence bit indicate that this page is not available in main memory. The CPU reference a word in main memory with a virtual address of 13 bits. The 03 high order bits of the virtual address specify a page number and also an address for the memory-page table.

Associative Memory page table →A random access memory page table is inefficient with respect to storage utilization. A more efficient way to organize the page table would be to construct it with a number of words equal to the way number of blocks in main memory. In this the size of the memory is reduced and each location is fully utilized. This method can be implemented by means of an associate memory with each word in memory containing a page number together with its corresponding block number.



An associative memory page table

The Page field in each word is compared with the Page No. in the virtual address. If a match occurs, the word is read from memory and its corresponding block number is extracted.

Page Replacement → The memory management software system handles all the software operations for the efficient utilization of memory space.

It must decide:

(1) Which page in main memory ought to be removed to make room for a new page,

(2) When a new page is to be transferred from auxiliary memory to main memory, and

(3) Where the page is to be placed in main memory.

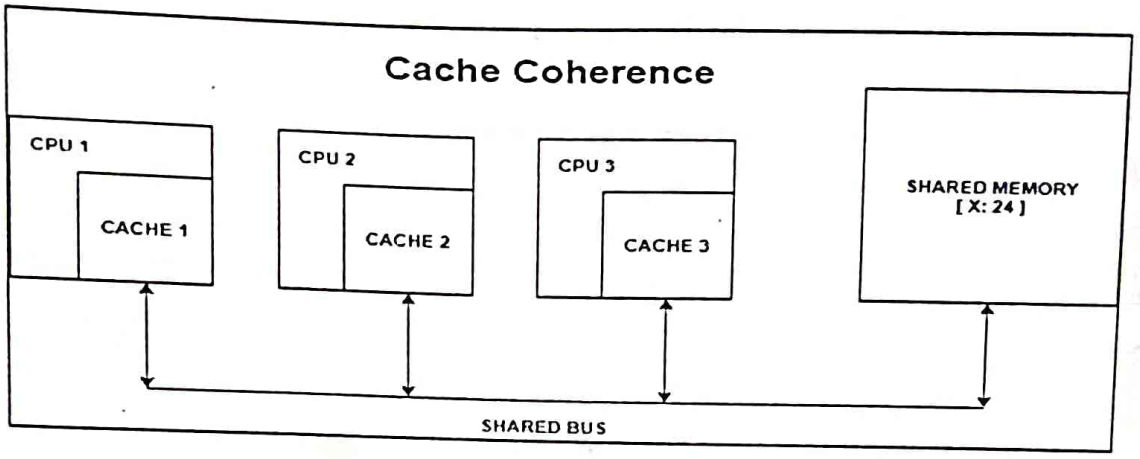
When a program starts execution, one or more pages are transferred into main memory and the page table is set to indicate their position. The program is executed from main memory until it attempts to reference a page that is still in auxiliary memory. This condition is called page fault. When the page fault occurs, the execution of the present program is suspended until the required page is brought into main memory. The policy for choosing pages to remove is determined from the replacement algorithm that is used. The goal of replacement policy is to try to remove the page least likely to be referenced in the immediate future. Two of the most common replacement algorithms used are, the first in first out (FIFO) and the least recently used (LRU).

CACHE COHERENCE AND SYNCHRONIZATION MECHANISMS

In a multiprocessor system, data inconsistency may occur among adjacent levels or within the same level of the memory hierarchy. For example, the cache and the main memory may have inconsistent copies of the same object.

As multiple processors operate in parallel, and independently multiple caches may possess different copies of the same memory block, this creates cache coherence problem. Cache coherence schemes help to avoid this problem by maintaining a uniform state for each cached block of data.

Example : Cache and the main memory may have inconsistent copies of the same object.



Suppose there are three processors, each having cache. Suppose the following scenario:-

- **Processor 1 read X :** obtains 24 from the memory and caches it.
- **Processor 2 read X :** obtains 24 from memory and caches it.
- **Again, processor 1 writes as X : 64,** Its locally cached copy is updated. Now, processor 3 reads X, what value should it get?
- Memory and processor 2 thinks it is 24 and processor 1 thinks it is 64.

As multiple processors operate in parallel, and independently multiple caches may possess different copies of the same memory block, this creates a cache coherence problem. Cache coherence is the discipline that ensures that changes in the values of shared operands are propagated throughout the system in a timely fashion. There are three distinct level of cache coherence :-

1. Every write operation appears to occur instantaneously.
2. All processors see exactly the same sequence of changes of values for each separate operand.
3. Different processors may see an operation and assume different sequences of values; this is known as non-coherent behavior.

There are various Cache Coherence Protocols in multiprocessor system. These are :-

1. MSI protocol (Modified, Shared, Invalid)
2. MOSI protocol (Modified, Owned, Shared, Invalid)
3. MESI protocol (Modified, Exclusive, Shared, Invalid)
4. MOESI protocol (Modified, Owned, Exclusive, Shared, Invalid)

These important terms are discussed as follows:

- **Modified** – It means that the value in the cache is dirty, that is the value in current cache is different from the main memory.
- **Exclusive** – It means that the value present in the cache is same as that present in the main memory, that is the value is clean.
- **Shared** – It means that the cache value holds the most recent data copy and that is what shared among all the cache and main memory as well.
- **Owned** – It means that the current cache holds the block and is now the owner of that block, that is having all rights on that particular blocks.
- **Invalid** – This states that the current cache block itself is invalid and is required to be fetched from other cache or main memory.

For detail on above protocol, refer :- Cache coherence protocol

Coherency mechanisms : There are three types of coherence :

1. **Directory-based** – In a directory-based system, the data being shared is placed in a common directory that maintains the coherence between caches. The directory acts as a filter through which the processor must ask permission to load an entry from the primary memory to its cache. When an entry is changed, the directory either updates or invalidates the other caches with that entry.
2. **Snooping** – First introduced in 1983, snooping is a process where the individual caches monitor address lines for accesses to memory locations that they have cached. It is called a write invalidate protocol. When a write operation is observed to a location that a cache has a copy of and the cache controller invalidates its own copy of the snooped memory location.
3. **Snarfing** – It is a mechanism where a cache controller watches both address and data in an attempt to update its own copy of a memory location when a second master modifies a location in main memory. When a write operation is observed to a location that a cache has a copy of the cache controller updates its own copy of the snarfed memory location with the new data.