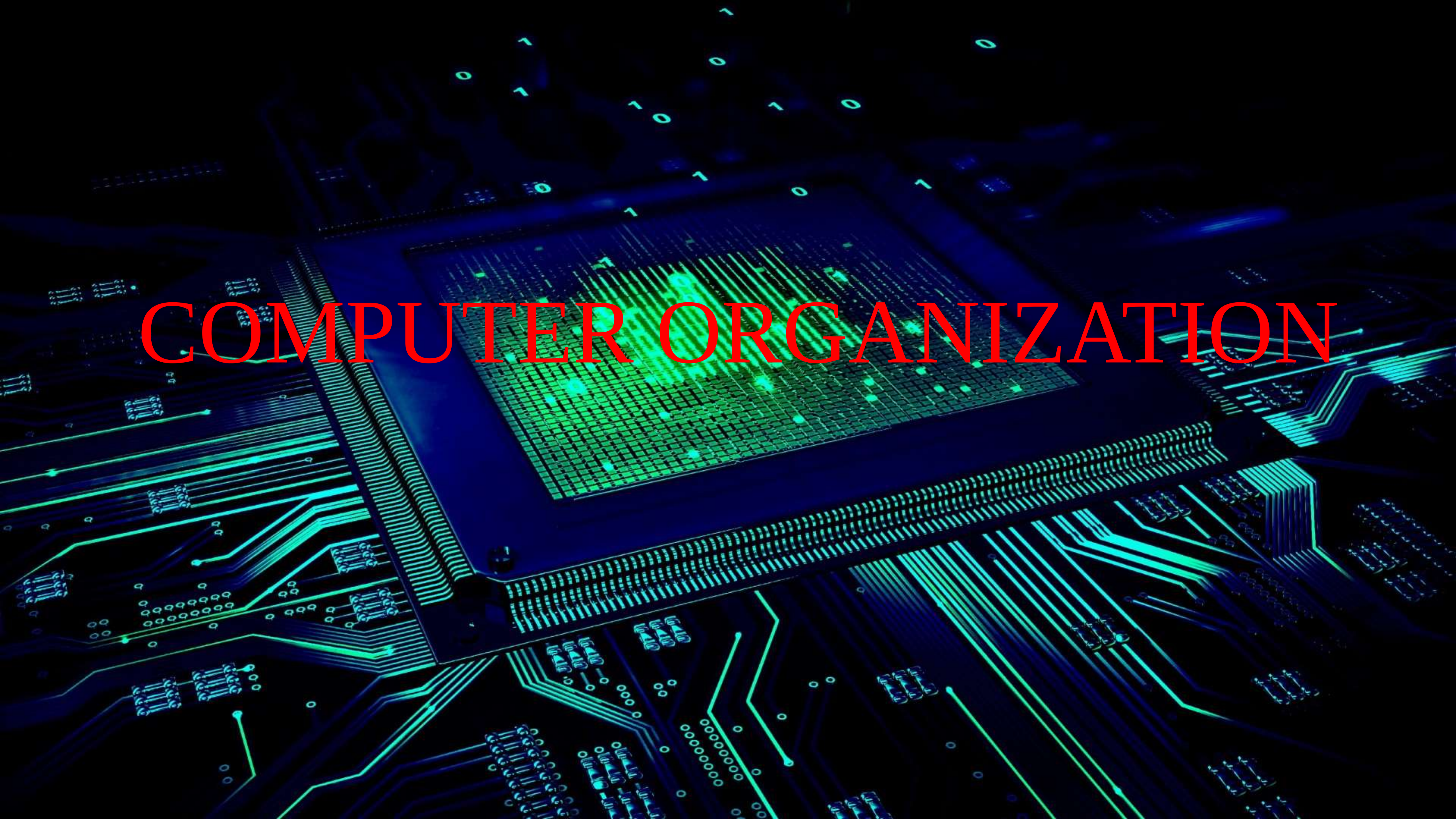


COMPUTER ORGANIZATION

Chapter 1 :

COMPUTER

Computer is an electronic device that takes inputs from the input devices and processes that Inputs and produces output by the output devices

Or

A computer is an electronic ,input processing out machine

Addition of two numbers

```
Void main()
{
int a,b,c;
clrscr();
printf(“enter a,b values”);
scanf(“%d %d”,&a,&b);
c=a+b;
printf(“%d”,c);
getch();
}
```

From the above example

input-a,b

Processing-addition

Output-c

Computer architecture

- The computer architecture concerned with the **structure** and behavior of the basic **hardware** we built

Computer organization

- The computer organization is concerned with the way that the **hardware components operate** and the way they are connected together to form the computer system

- Computer organization is deals with hardware that provides interface to the operating systems and compilers that are driven by the user or computer programmer | customers

Computer design

- The **computer design** is concerned with the hardware of the computer.
- It is concerned with determination of what hardware should be used and how the parts should be connected

Register transfer language

<u>BIT:</u>

A bit is a binary digit that is either 0 or 1

<u>FLIP FLOP:</u>

A flip flop is a memory device that stores one bit of binary information(i.e., it can store either a binary 0 or binary 1)

Register

- A register is a group of flip flops that is used to store binary information
- It is denoted by ‘R’

Example:program counter(PC),

memory address register(MAR),

memory data register(MDR),

instruction register(IR),

memory buffer register(MBR),

accumulator(AC) register.

Register transfer

- The process of transferring data from one register to another register is referred as

Register transfer

Eg: T:R1<- R2

Suppose R1=00000000

R2=00100100

COMPUTER ARCHITECTURE

computer architecture is a set of rules and methods that describe the functionality, organization, and implementation of computer systems

Computer Organization

The computer organization is concerned with the structure and behavior of digital computers. The main objective of this subject to understand the overall basic computer hardware structure, including the peripheral devices.

Computer Organization is realization of what is specified by the computer architecture.

It deals with how operational attributes are linked together to meet the requirements specified by computer architecture.

Some organizational attributes are hardware details, control signals, peripherals

BASIC STRUCTURE OF COMPUTERS

1.1 What is Computer

- A computer can be defined as a fast electronic calculating machine that accepts the (data) digitized input information process it as per the list of internally stored instructions and produces the resulting information.
- The list of instructions are called programs & internal storage is called computer memory.
- Since their introduction in the 1940s, digital computers have evolved into many different types that vary widely in size, cost, computational power, and intended use. Modern computers can be divided roughly into six general categories:

The different types of computers are

1. Embedded computers
2. Personal computers
3. Book computers
4. Work stations
5. Servers andEnterprise systems
6. Super computers

Embedded computers are purpose-built computing platforms, designed for a specific, software-controlled task.

An embedded computer is a microprocessor-based system, specially designed to perform a specific function and belong to a larger system. It comes with a combination of hardware and software to achieve a unique task and withstand different conditions.

Embedded computers are integrated into a larger device or system in order to automatically monitor and control a physical process or environment.

They are used for a specific purpose rather than for general processing tasks. Typical applications include industrial and home automation, appliances, telecommunication products, and vehicles. Users may not even be aware of the role that computers play in such systems.

Personal computers

This is the most common type found in homes, schools, Business offices etc.

A **personal computer** is a computer small and low cost, which is intended for personal use (or for use by a small group of individuals). The term “personal computer” is used to describe desktop computers (desktops). It is often shortened to the acronym PC or microcomputer, whose meaning in English is “personal computer”. It is a very common type of machines.

Personal Computer (acronym PC) consists of a central processing unit (CPU) contains the arithmetic, logic, and control circuitry on an single (IC) integrated circuit; two types of memory, main memory, such as RAM, and ROM, magnetic hard disks (HDD) and compact discs and various input/output devices, including a display screen, keyboard and mouse, modem, and printer.

Personal computers have achieved widespread use in homes, educational institutions, and business and engineering office settings, primarily for dedicated individual use.

They support a variety of applications such as general computation, document preparation, computer-aided design, audiovisual entertainment, interpersonal communication, and Internet browsing.

Note book computers

Notebook computers provide the basic features of a personal computer in a smaller lightweight package.

These are compact and portable versions of PC

A portable, compact computer that can run on power supply or a battery unit.

All components are integrated as one compact unit. It is generally more expensive than a comparable desktop. It is also called a Notebook.

Work stations

Workstation computers offer higher computational capacity and more powerful graphical display capabilities for engineering and scientific work

These have high resolution input/output (I/O) graphics capability, but with same dimensions as that of desktop computer.

These are used in engineering applications of interactive design work.

Servers and Enterprise systems

Servers and Enterprise systems are large computers that are meant to be shared by a potentially large number of users who access them from some form of personal computer over a public or private network.

Such computers may host large databases and provide information processing for a government agency or a commercial organization.

These are used for business data processing in medium to large corporations that require much more computing power and storage capacity than work stations.

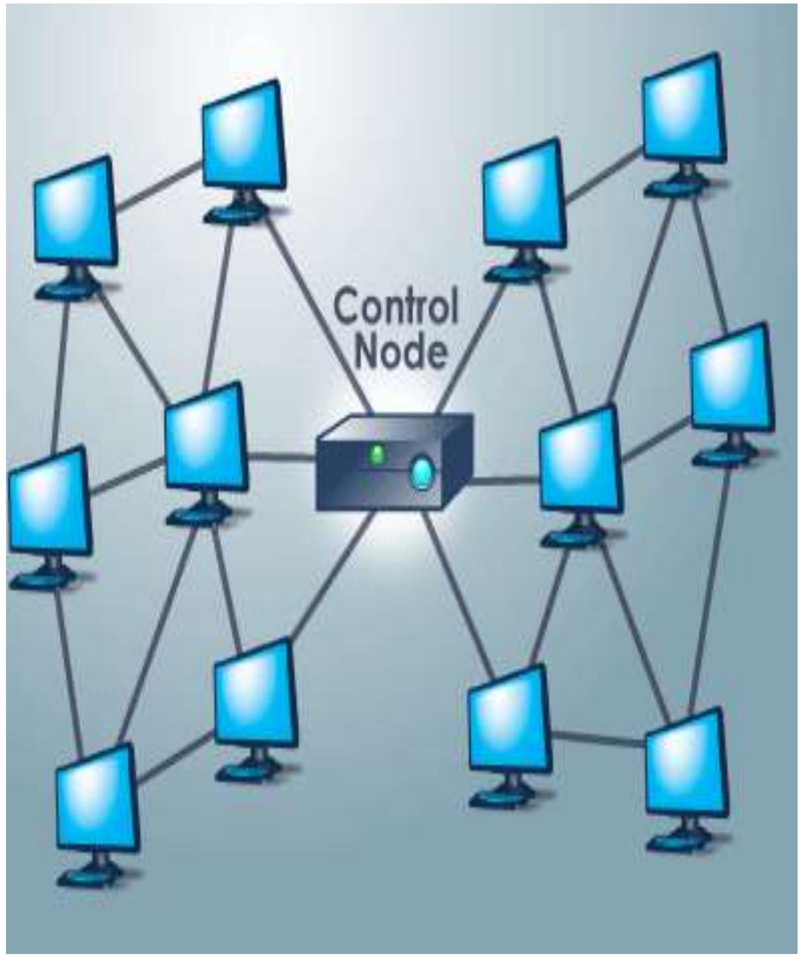
Super computers and Grid computers

A computer that is considered to be fastest in the world. Used to execute tasks that would take lot of time for other computers. For Ex: These are used for large scale numerical calculations required in the applications like weather forecasting etc.,

Supercomputers and Grid computers normally offer the highest performance. They are the most expensive and physically the largest category of computers. Supercomputers are used for the highly demanding computations needed in weather forecasting, engineering design and simulation, and scientific work and Modeling weather systems, genome sequence, etc. They have a high cost.

Grid computers provide a more cost-effective alternative. They combine a large number of personal computers and disk storage units in a physically distributed high-speed network, called a grid, which is managed as a coordinated computing resource.

By evenly distributing the computational workload across the grid, it is possible to achieve high performance on large applications ranging from numerical computation to information searching.



1.2 FUNCTIONAL UNITS

Functional units are a part of a CPU that performs the operations and calculations called for by the computer program.

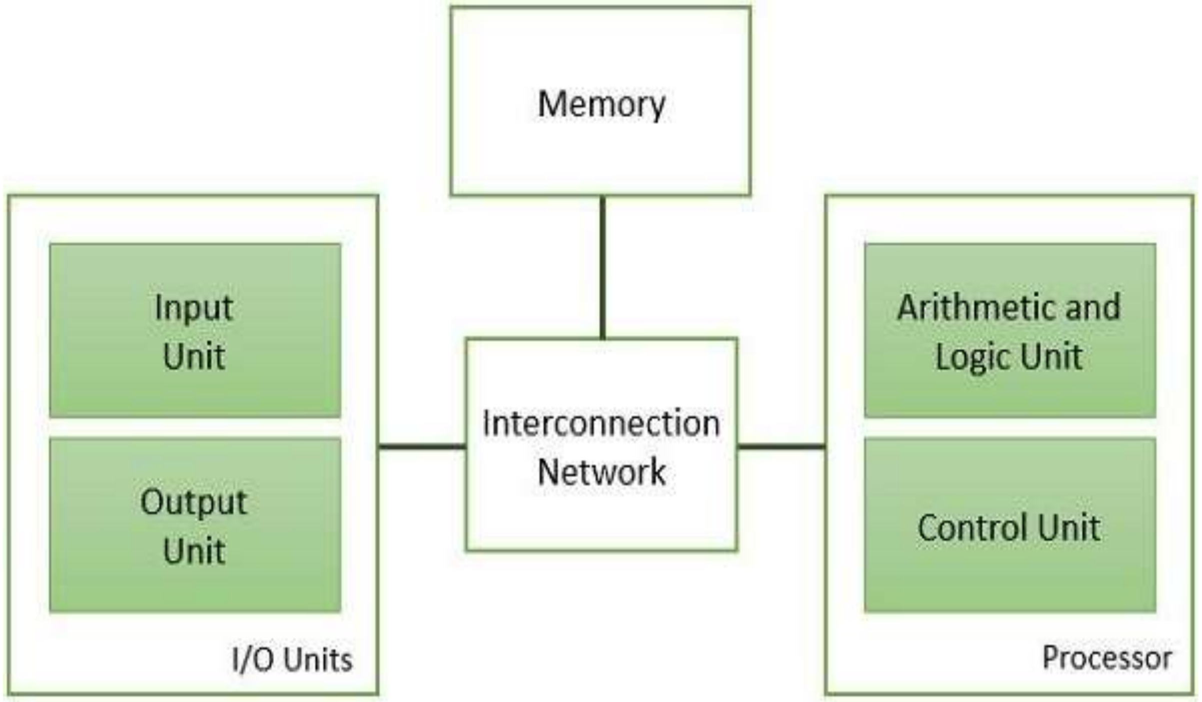


Figure 1. Functional Unit of Computers

- A computer consists of five functionally independent main parts: input, memory, arithmetic and logic, output, and control units,
- The input unit accepts coded information from human operators, from electromechanical devices such as key- boards, or from other computers over digital communication lines.
- The information received is either stored in the computer's memory for later reference or immediately used by the arithmetic and logic circuitry to perform the desired operations.
- The processing steps are determined by a program stored in the memory. Finally, the results are sent back to the outside world through the output unit. All of these actions are coordinated by the control unit.
- It is convenient to categorize this information as either instructions or data.
- Instructions, or machine instructions, are explicit commands that

Govern the transfer of information within a computer as well as between the computer and its I/O devices.

Specify the arithmetic and logic operations to be performed .

A program is a list of instructions which performs a task. Programs are stored in the memory. The processor fetches the program instructions from the memory, one after another, and performs the desired operations.

Data are numbers and characters that are used as operands by the instructions. Data are also stored in the memory.

Basic Functional Units

1. Input Unit
2. Memory Unit

    Primary Memory

    Cache Memory

    Secondary Storage

3. Arithmetic and Logic Unit
4. Output Unit
5. Control Unit

INPUT UNIT

- Computers accept coded information through input units, which read the data.
- The most well-known input device is the keyboard. Whenever a key is pressed, the corresponding letter or digit is automatically translated into its corresponding binary code and transmitted over a cable to either the memory or the processor.
- Many other kinds of input devices are available, including joysticks, trackballs, and mouse's, Light pen, Digitizer, Scanner.
- These are often used as graphic input devices in conjunction with displays.
- Microphones can be used to capture audio input which is then sampled and converted into digital codes for storage and processing.

MEMORY UNIT

- The function of the memory unit is to store programs and data.
- There are two classes of storage, called primary and secondary.

Primary memory:

- Primary memory is also known as the main memory or the random-access memory (RAM). It is the fastest accessible memory of the computer. If a program has to be executed it first needs to be placed in the primary memory.
- Then the instructions of the program are fetched one at a time by the processor for execution. It is the one exclusively associated with the processor and operates at the electronics speeds programs must be stored in this memory while they are being executed.
- The memory contains a large number of semiconductors storage cells.
- Each ALU Processor Control Unit capable of storing one bit of information. These are processed in a group of fixed site called word.
- The memory is organized in such a way that in one basic operation, one-word can be retrieved from the memory or one word can be stored to the memory. A word length could be 16, 32, or 64 bits.

- To provide easy access to a word in memory, a distinct address is associated with each word location. Addresses are numbers that identify memory location. Number of bits in each word is called word length of the computer.
- Programs must reside in the memory during execution. Instructions and data can be written into the memory or read out under the control of processor.
- Memory in which any location can be reached in a short and fixed amount of time after specifying its address is called random-access memory (RAM).

Cache Memory

- Cache memory can be accessed much faster as compared to primary memory and it is even smaller in size. It is stored with the data that is required frequently by the processor.
- As an adjunct to the main memory, a smaller, faster RAM unit, called a cache, is used to hold sections of a program that are currently being executed, along with any associated data.

The cache is tightly coupled with the processor and is usually contained on the same integrated-circuit chip. The purpose of the cache is to facilitate high instruction execution rates.

At the start of program execution, the cache is empty. All program instructions and any required data are stored in the main memory.

As execution proceeds, instructions are fetched into the processor chip, and a copy of each is placed in the cache.

When the execution of an instruction requires data located in the main memory, the data are fetched and copies are also placed in the cache.

Now, suppose a number of instructions are executed repeatedly as happens in a program loop. If these instructions are available in the cache, they can be fetched quickly during the period of repeated use.

Similarly, if the same data locations are accessed repeatedly while copies of their contents are available in the cache, they can be fetched quickly.

Secondary memory

- Is used where large amounts of data & programs have to be stored, particularly information that is accessed infrequently.
- The secondary memory is slower and less expensive as compared to primary memory. It doesn’t lose its contents even if the supply of power gets off.
  We require secondary memory to store a large volume of data or program permanently or the data that is less likely to be retrieved.
- Examples : Magnetic disks & tapes, optical disks (i.e.. CD-ROM’s), floppies etc.,

Arithmetic logic unit (ALU):

- Most of the computer operators are executed in ALU of the processor like addition, subtraction, division, multiplication, etc. the operands are brought into the ALU from memory and stored in high speed storage elements called register.

- For example, if two numbers located in the memory are to be added, they are brought into the processor, and the addition is carried out by the ALU. The sum may then be stored in the memory or retained in the processor for immediate use.
- When operands are brought into the processor, they are stored in high-speed storage elements called registers. Each register can store one word of data.
- Then according to the instructions the operation is performed in the required sequence. The control and the ALU are may times faster than other devices connected to a computer system.
- This enables a single processor to control a number of external devices such as keyboards, displays, magnetic and optical disks, sensors and other mechanical controllers.

Output Unit: Computer after computation returns the computed results, error messages, etc. via output unit.

The standard output device is a video monitor, LCD/TFT monitor. Other output devices are printers, plotters etc.

Control Unit:

- The memory, arithmetic and logic, and I/O units store and process information and perform input and output operations
- The operation of these units must be coordinated in some way. This is the responsibility of the control unit.
- The control unit is effectively the nerve center that sends control signals to other units and senses their states
- Control unit co-ordinates activities of all units by issuing control signals.
- Control signals issued by control unit govern the data transfers and then appropriate operations take place. Control unit interprets or decides the operation/action to be performed.

The operations of a computer can be summarized as follows:

1. A set of instructions called a program reside in the main memory of computer.

2. Information stored in the memory is fetched under program control into an arithmetic and logic unit, where it is processed.

3. Processed information leaves the computer through an output unit.

4. All activities in the computer are directed by the control unit.

Main Memory or primary memory

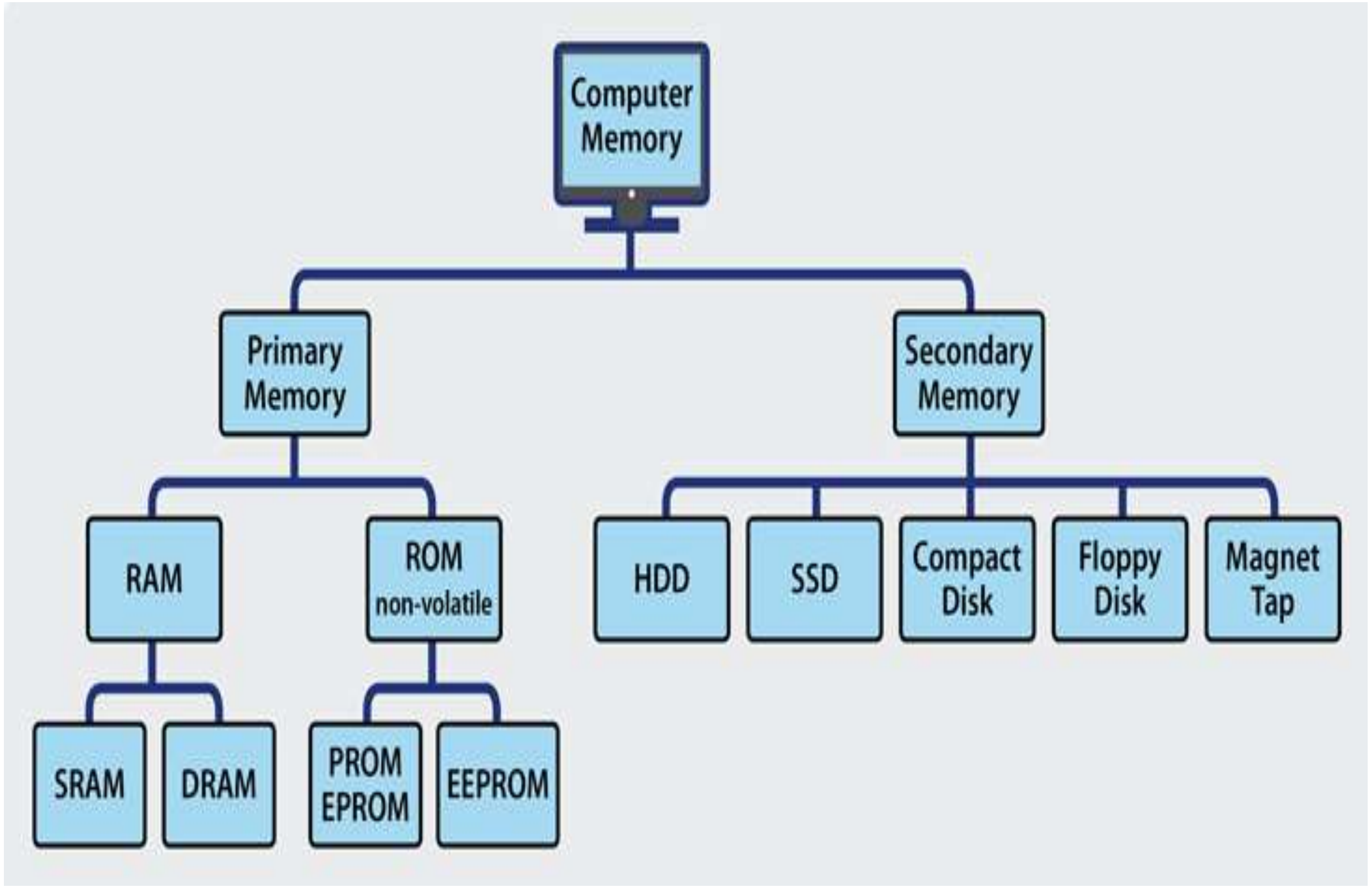
- The memory unit that communicates directly within the CPU, i/o processor and Cache memory, is called main memory.
- It is the central storage unit of the computer system. It is a large and fast memory used to store data

  memory/storage is classified into 2 categories:

  Volatile Memory: This loses its data, when power is switched off.

- **Non-Volatile Memory**: This is a permanent storage and does not lose any data when power is switched off.

Main memory

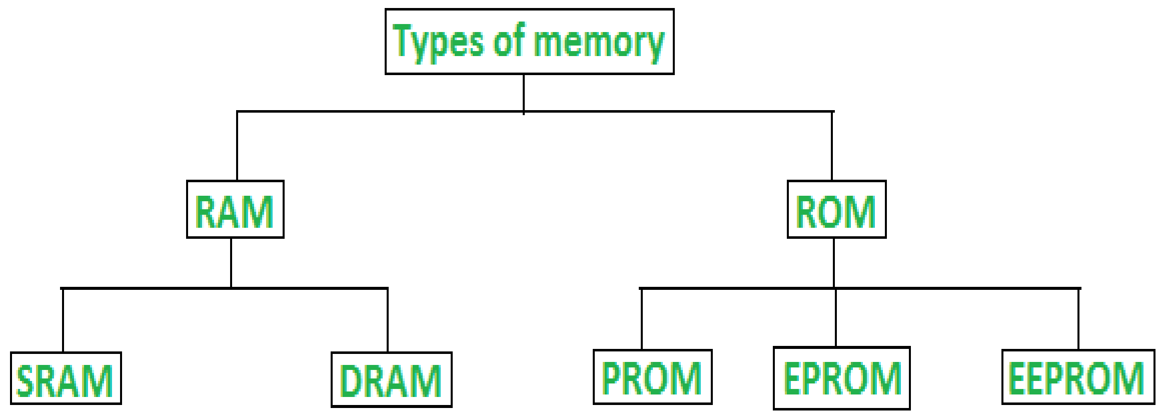


Solid State Drive(SSD)

Hard Disk Drives (HDD)

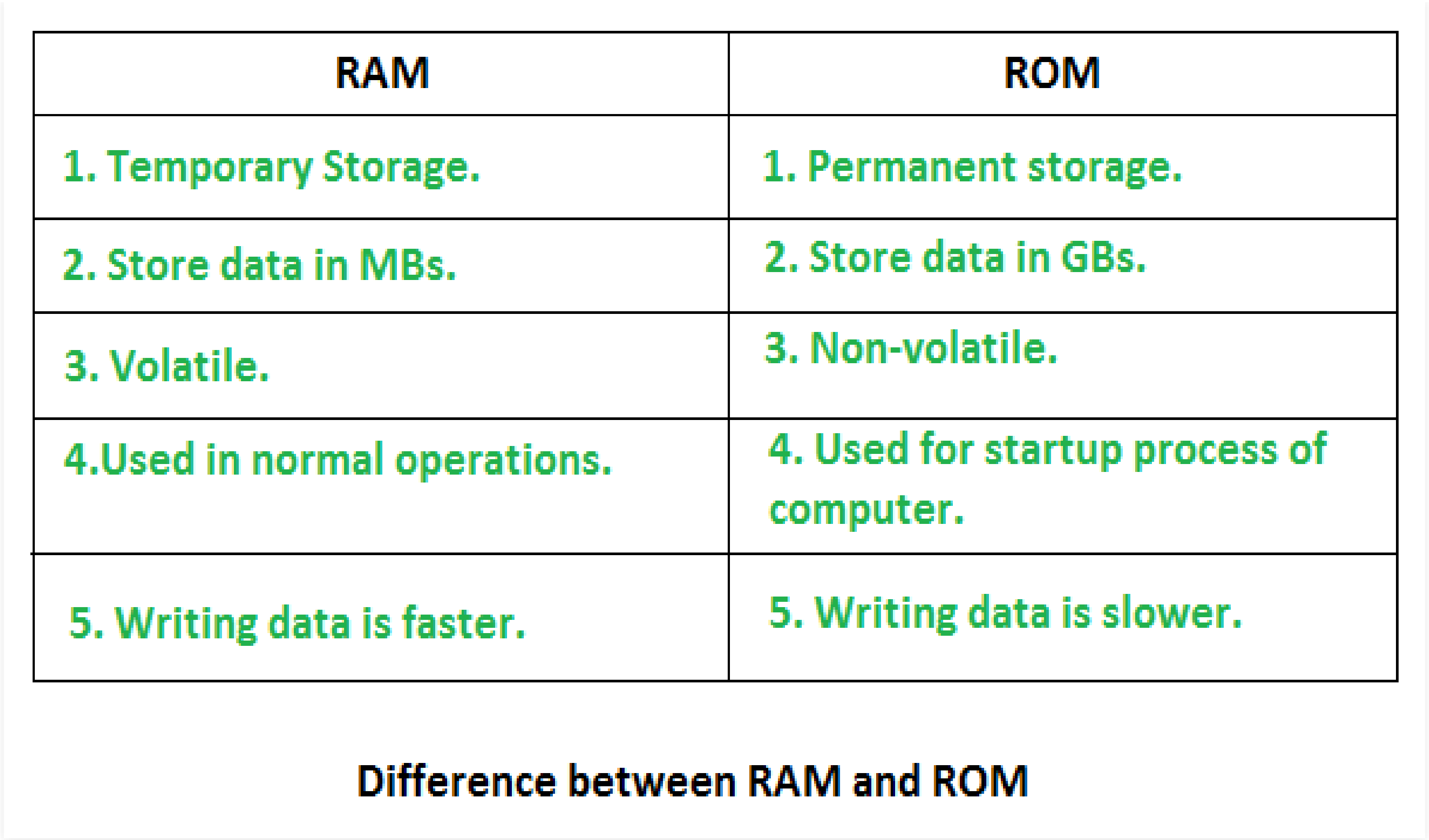
RAM(Random Access Memory (RAM)

- it is also called as *read write memory* or the *main memory* or the *primary memory*.
- it is a volatile memory as the data loses when the power is turned off.



Classification of computer memory

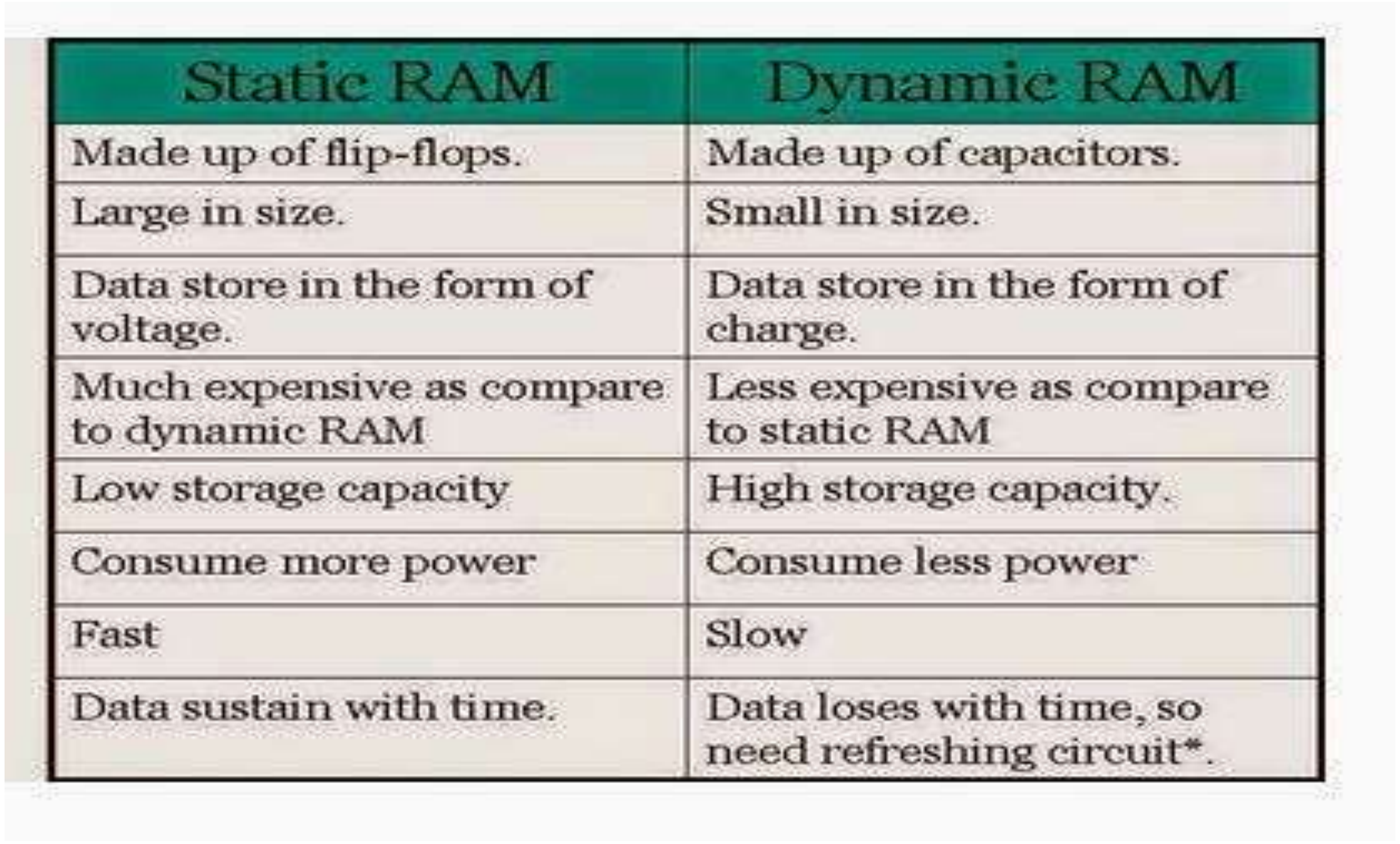
Difference between RAM and ROM



RAM	ROM
1. Temporary Storage.	1. Permanent storage.
2. Store data in MBs.	2. Store data in GBs.
3. Volatile.	3. Non-volatile.
4.Used in normal operations.	4. Used for startup process of computer.
5. Writing data is faster.	5. Writing data is slower.

Difference between RAM and ROM

Difference Between SRAM and DRAM



Static RAM	Dynamic RAM
Made up of flip-flops.	Made up of capacitors.
Large in size.	Small in size.
Data store in the form of voltage.	Data store in the form of charge.
Much expensive as compare to dynamic RAM	Less expensive as compare to static RAM
Low storage capacity	High storage capacity.
Consume more power	Consume less power
Fast	Slow
Data sustain with time.	Data loses with time, so need refreshing circuit*.

Types of Read Only Memory (ROM)

- **PROM (Programmable read-only memory)** – It can be programmed by user. Once programmed, the data and instructions in it cannot be changed.

- **EPROM (Erasable Programmable read only memory)** – It can be reprogrammed. To reprogram it, erase all the previous data.

- **EEPROM (Electrically erasable programmable read only memory)** – The data can be erased by applying electric field We can erase only portions of the chip.

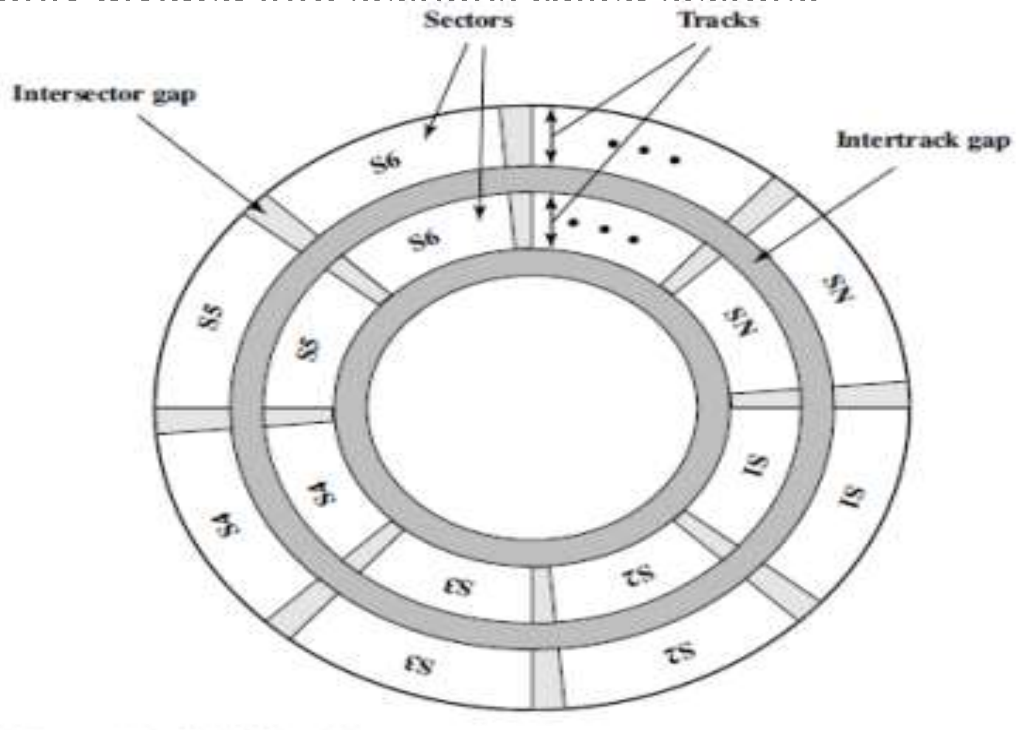
-

Auxiliary Memory:

- The most common auxiliary memory devices used in computer systems are **magnetic disks** and **tapes**.

Magnetic Disks:

- A magnetic disk is a circular plate constructed of metal or plastic coated with magnetized material.
- Bits are stored in the magnetized circles called tracks.
- The tracks are commonly divided into sections called sectors.



Magnetic Tapes:

- A magnetic tape consists of the electrical, mechanical, and electronic components to provide the parts and control mechanism for a magnetic tape unit.

- Magnetic tape units can be stopped, started to move forward or in reverse, or can be rewound.

- In magnetic tapes information is recorded in blocks referred as records. Records may be of fixed or variable length

1.3 Basic operational concepts

- To perform a given task an appropriate program consisting of a list of instructions is stored in the memory. Individual instructions are brought from the memory into the processor, which executes the specified operations. Data to be stored are also stored in the memory.

Examples: - Add LOCA, R0

- This instruction adds the operand at memory location LOCA, to operand in register R0 & places the sum into register.

This instruction requires the performance of several steps,

- 1. First the instruction is fetched from the memory into the processor.
- 2. The operand at LOCA is fetched and added to the contents of R0
- 3. Finally the resulting sum is stored in the register R0

- The preceding add instruction combines a memory access operation with an ALU operations. In some other type of computers, these two types of operations are performed by separate instructions for performance reasons.

Load LOCA, R_1

Add R_1, R_0

The steps to execute the instructions can be enumerated as below:

Step 1: Fetch the instruction from main memory into the processor

Step 2: Fetch the operand at location LOCA from main memory into the processor Register R_1

Step 3: Add the content of Register R_1 and the contents of register R_0

Step 4: Store the result (sum) in R_0.

- After operands have been loaded from memory into processor registers, arithmetic or logic operations can be performed on them.

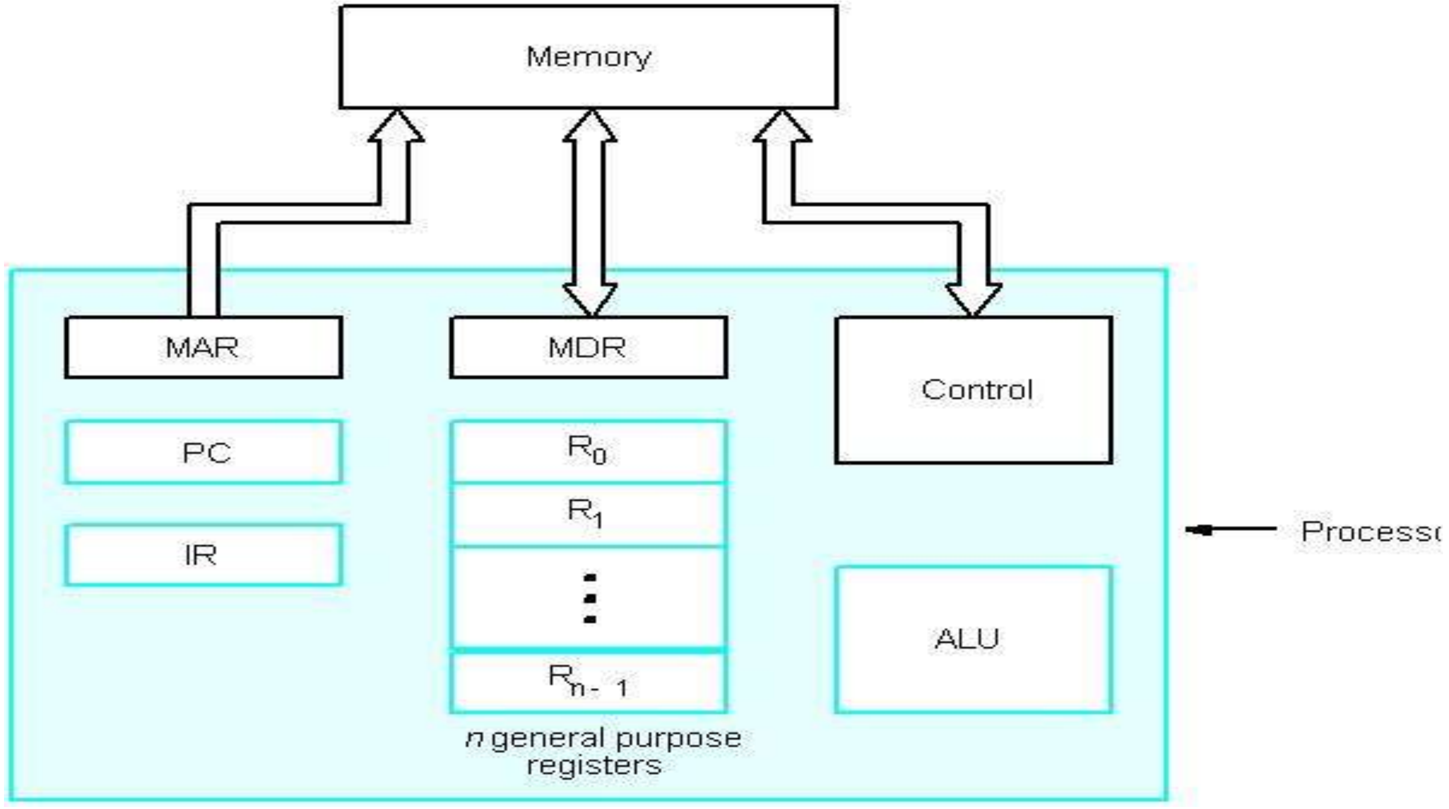
  For example, the instruction Add R4, R2, R3

- adds the contents of registers R2 and R3, then places their sum into register R4. The operands in R2 and R3 are not altered, but the previous value in R4 is overwritten by the sum.
- After completing the desired operations, the results are in processor registers. They can be transferred to the memory using instructions such as

  Store R4, LOC

- This instruction copies the operand in register R4 to memory location LOC. The original contents of location LOC are overwritten, but those of R4 are preserved.
- For Load and Store instructions, transfers between the memory and the processor are initiated by sending the address of the desired memory location to the memory unit and asserting the appropriate control signals. The data are then transferred to or from the memory.

- The below shows how the memory and the processor are connected. As shown in the diagram, in addition to the ALU and the control circuitry, the processor contains a number of registers used for several different purposes. The instruction register holds the instruction that is currently being executed.
- The program counter keeps track of the execution of the program. It contains the memory address of the next instruction to be fetched and executed. There are n general purpose registers R0 to Rn-1 which can be used by the programmers during writing programs.



- The fig shows how memory & the processor can be connected. In addition to the ALU & the control circuitry, the processor contains a number of registers used for several different purposes.

The instruction register (IR):- Holds the instructions that is currently being executed. Its output is available for the control circuits which generates the timing signals that control the various processing elements in one execution of instruction.

The program counter PC:-

- This is another specialized register that keeps track of execution of a program. It contains the memory address of the next instruction to be fetched and executed. Besides IR and PC, there are n-general purpose registers R0 through Rn-1.

The other two registers which facilitate communication with memory are: -

1. MAR – (Memory Address Register):- It holds the address of the location to be accessed.

2. MDR – (Memory Data Register):- It contains the data to be written into or readout of the address location.

Operating steps are

1. Programs reside in the memory & usually get these through the I/P unit.

2. Execution of the program starts when the PC is set to point at the first instruction of the program.

3. Contents of PC are transferred to MAR and a Read Control Signal is sent to the memory.

4. After the time required to access the memory elapses, the address word is read out of the memory and loaded into the MDR.

5. Now contents of MDR are transferred to the IR & now the instruction is ready to be decoded and executed.

6. If the instruction involves an operation by the ALU, it is necessary to obtain the required operands.

7. An operand in the memory is fetched by sending its address to MAR & Initiating a read cycle.

8. When the operand has been read from the memory to the MDR, it is transferred from MDR to the ALU.

9. After one or two such repeated cycles, the ALU can perform the desired operation.

10. If the result of this operation is to be stored in the memory, the result is sent to MDR.

11. Address of location where the result is stored is sent to MAR & a write cycle is initiated.

12. The contents of PC are incremented so that PC points to the next instruction that is to be executed.

- Normal execution of a program may be preempted (temporarily interrupted) if some devices require urgent servicing, to do this one device raises an Interrupt signal.
- An interrupt is a request signal from an I/O device for service by the processor.
- The processor provides the requested service by executing an appropriate interrupt service routine.
- The Diversion may change the internal stage of the processor its state must be saved in the memory location before interruption. When the interrupt-routine service is completed the state of the processor is restored so that the interrupted program may continue.

1.4 Number Representation and Arithmetic Operations

The most natural way to represent a number in a computer system is by a string of bits, called a binary number.

Integers

Integers are whole numbers or fixed-point numbers with the radix point fixed after the least-significant bit.

They are contrast to real numbers or floating-point numbers, where the position of the radix point varies.

It is important to take note that integers and floating-point numbers are treated differently in computers.

They have different representation and are processed differently (e.g., floating-point numbers are processed in a so-called floating-point processor).

Computers use a fixed number of bits to represent an integer. The commonly-used bit-lengths for integers are 8-bit, 16-bit, 32-bit or 64-bit.

There are two representation schemes for integers:

- *Unsigned Integers*: can represent zero and positive integers.
- *Signed Integers*: can represent zero, positive and negative integers. Three representation schemes had been proposed for signed integers

We need to represent both positive and negative numbers. Three systems are used for representing such numbers:

- Sign-and-magnitude
- 1’s-complement
- 2’s-complement

n-bit Unsigned Integers

Unsigned integers can represent zero and positive integers, but not negative integers. The value of an unsigned integer is interpreted as "the magnitude of its underlying binary pattern".

Example 1: Suppose that n=8 and the binary pattern is 0100 0001B, the value of this unsigned integer is 1×2^0 + 1×2^6 = 65D.

Example 2: Suppose that n=16 and the binary pattern is 0001 0000 0000 1000B, the value of this unsigned integer is 1×2^3 + 1×2^12 = 4104D.

Example 3: Suppose that n=16 and the binary pattern is 0000 0000 0000 0000B, the value of this unsigned integer is 0.

An n-bit pattern can represent 2^n distinct integers. An n-bit unsigned integer can represent integers from 0 to (2^n)-1, as tabulated below:

n	Minimum	Maximum
8	0	(2^8)-1 (=255)
16	0	(2^16)-1 (=65,535)
32	0	(2^32)-1 (=4,294,967,295) (9+ digits)
64	0	(2^64)-1 (=18,446,744,073,709,551,615) (19+ digits)

Signed Integers

Signed integers can represent zero, positive integers, as well as negative integers. Three representation schemes are available for signed integers:

Sign-Magnitude representation

1's Complement representation

2's Complement representation

In all the above three schemes, the most-significant bit (msb) is called the sign bit. The sign bit is used to represent the sign of the integer - with 0 for positive integers and 1 for negative integers.

The magnitude of the integer, however, is interpreted differently in different schemes.

n-bit Sign Integers in Sign-Magnitude Representation

In sign-magnitude representation:

The most-significant bit (msb) is the sign bit, with value of 0 representing positive integer and 1 representing negative integer.

The remaining n-1 bits represents the magnitude (absolute value) of the integer. The absolute value of the integer is interpreted as "the magnitude of the (n-1)-bit binary pattern".

Example 1: Suppose that n=8 and the binary representation is 0 100 0001B.

Sign bit is 0 ⇒ positive

Absolute value is 100 0001B = 65D

Hence, the integer is +65D

Example 2: Suppose that n=8 and the binary representation is 1 000 0001B.

Sign bit is 1 ⇒ negative

Absolute value is 000 0001B = 1D

Hence, the integer is -1D

Example 3: Suppose that n=8 and the binary representation is 0 000 0000B.

Sign bit is 0 ⇒ positive

Absolute value is 000 0000B = 0D

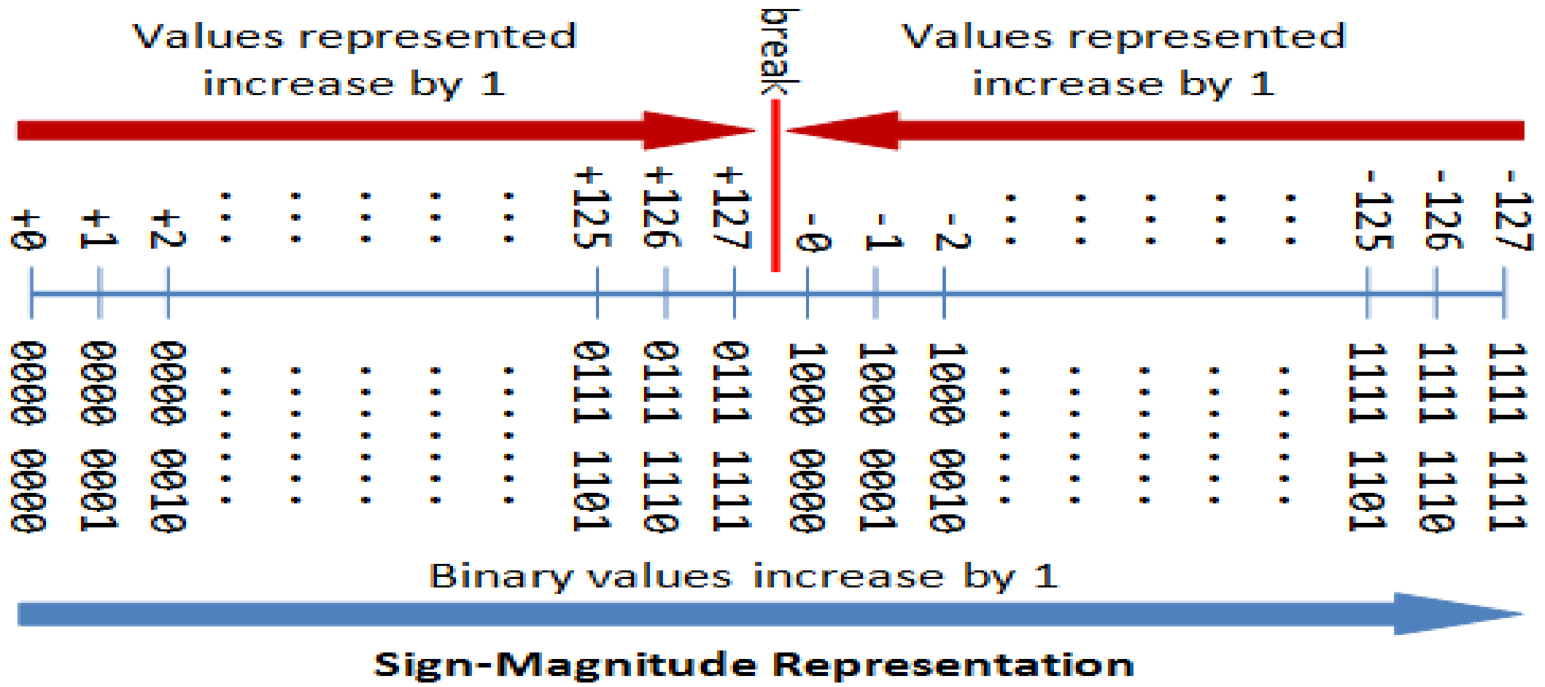
Hence, the integer is +0D

Example 4: Suppose that n=8 and the binary representation is 1 000 0000B.

Sign bit is 1 ⇒ negative

Absolute value is 000 0000B = 0D

Hence, the integer is -0D



The drawbacks of sign-magnitude representation are:

There are two representations (0000 0000B and 1000 0000B) for the number zero, which could lead to inefficiency and confusion.

Positive and negative integers need to be processed separately.

n-bit Sign Integers in 1's Complement Representation

In 1's complement representation:

Again, the most significant bit (msb) is the sign bit, with value of 0 representing positive integers and 1 representing negative integers.

The remaining n-1 bits represents the magnitude of the integer, as follows:

for positive integers, the absolute value of the integer is equal to "the magnitude of the (n-1)-bit binary pattern".

for negative integers, the absolute value of the integer is equal to "the magnitude of the complement (inverse) of the (n-1)-bit binary pattern" (hence called 1's complement).

Example 1: Suppose that n=8 and the binary representation 0 100 0001B.

Sign bit is 0 ⇒ positive

Absolute value is 100 0001B = 65D

Hence, the integer is +65D

Example 2: Suppose that n=8 and the binary representation 1 000 0001B.

Sign bit is 1 ⇒ negative

Absolute value is the complement of 000 0001B, i.e., 111 1110B = 126D

Hence, the integer is -126D

Example 3: Suppose that n=8 and the binary representation 0 000 0000B.

Sign bit is 0 ⇒ positive

Absolute value is 000 0000B = 0D

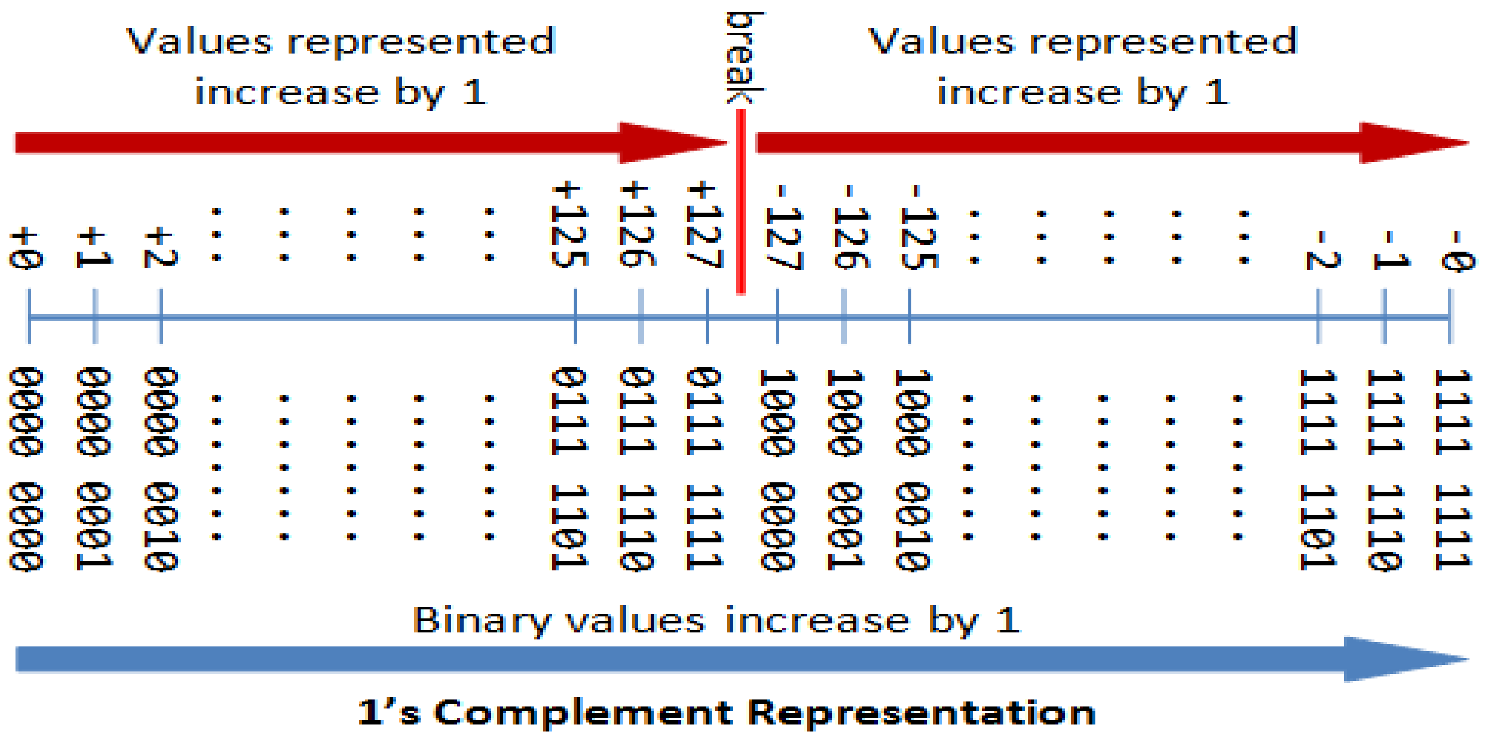
Hence, the integer is +0D

Example 4: Suppose that n=8 and the binary representation 1 111 1111B.

Sign bit is 1 ⇒ negative

Absolute value is the complement of 111 1111B, i.e., 000 0000B = 0D

Hence, the integer is -0D



Again, the drawbacks are:

There are two representations (0000 0000B and 1111 1111B) for zero.

The positive integers and negative integers need to be processed separately.

n-bit Sign Integers in 2's Complement Representation

In 2's complement representation:

Again, the most significant bit (msb) is the sign bit, with value of 0 representing positive integers and 1 representing negative integers.

The remaining n-1 bits represents the magnitude of the integer, as follows:

for positive integers, the absolute value of the integer is equal to "the magnitude of the (n-1)-bit binary pattern".

for negative integers, the absolute value of the integer is equal to "the magnitude of the complement of the (n-1)-bit binary pattern plus one" (hence called 2's complement).

Example 1: Suppose that n=8 and the binary representation 0 100 0001B.

Sign bit is 0 ⇒ positive

Absolute value is 100 0001B = 65D

Hence, the integer is +65D

Example 2: Suppose that n=8 and the binary representation 1 000 0001B.

Sign bit is 1 ⇒ negative

Absolute value is the complement of 000 0001B plus 1, i.e., 111 1110B + 1B = 127D

Hence, the integer is -127D

Example 3: Suppose that n=8 and the binary representation 0 000 0000B.

Sign bit is 0 ⇒ positive

Absolute value is 000 0000B = 0D

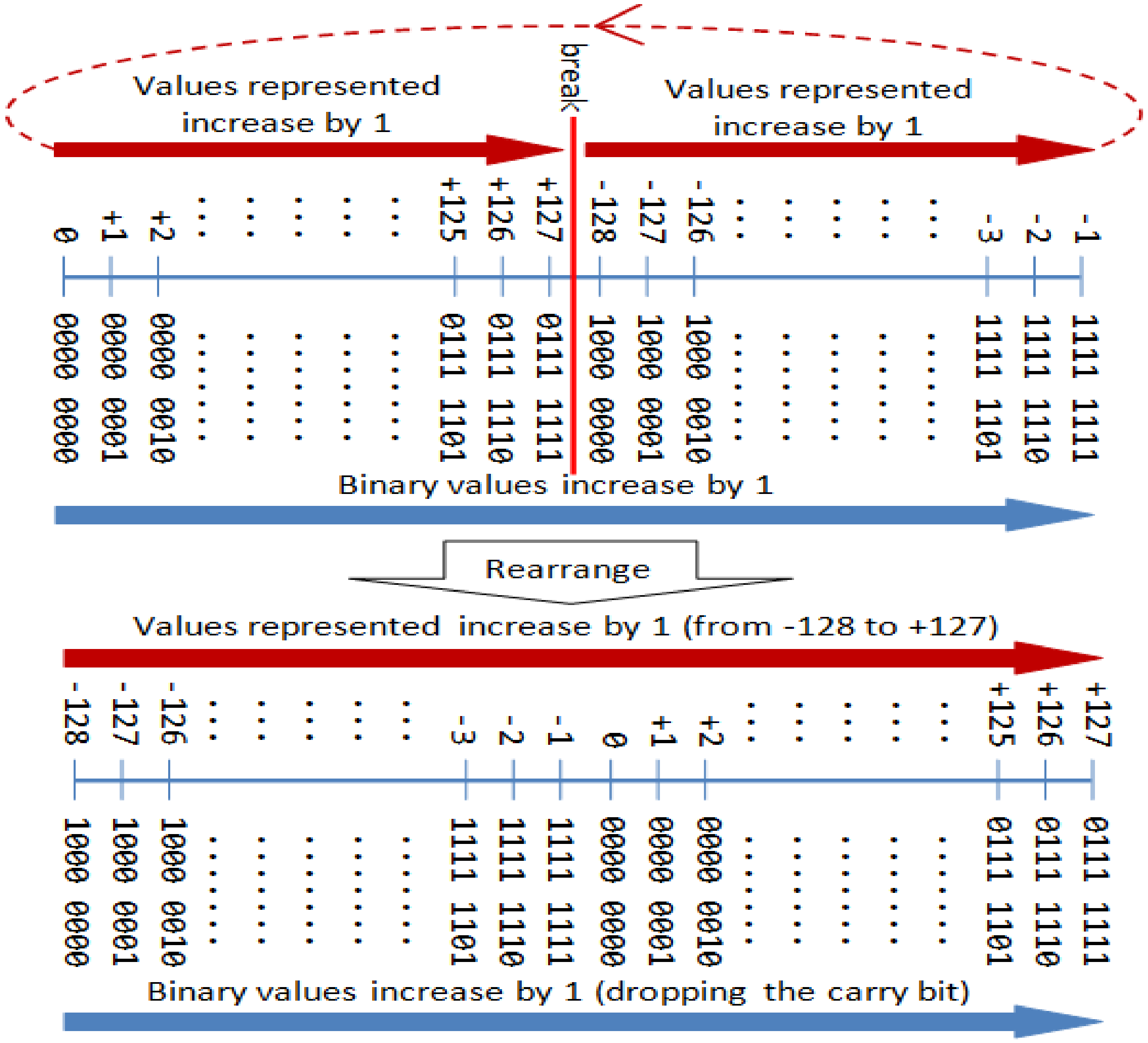
Hence, the integer is +0D

Example 4: Suppose that n=8 and the binary representation 1 111 1111B.

Sign bit is 1 ⇒ negative

Absolute value is the complement of 111 1111B plus 1, i.e., 000 0000B + 1B = 1D

Hence, the integer is -1D


break
Values represented increase by 1
Values represented increase by 1
0
+1
+2
+125
+126
+127
-128
-127
-126
-3
-2
-1
0000 0000
0000 0001
0000 0010
0111 1101
0111 1110
0111 1111
1000 0000
1000 0001
1000 0010
1111 1101
1111 1110
1111 1111
Binary values increase by 1
Rearrange
Values represented increase by 1 (from -128 to +127)
Binary values increase by 1 (dropping the carry bit)

2's Complement Representation

B	Values represented		
$b_3 b_2 b_1 b_0$	Sign and magnitude	1's complement	2's complement
0 1 1 1	+ 7	+ 7	+ 7
0 1 1 0	+ 6	+ 6	+ 6
0 1 0 1	+ 5	+ 5	+ 5
0 1 0 0	+ 4	+ 4	+ 4
0 0 1 1	+ 3	+ 3	+ 3
0 0 1 0	+ 2	+ 2	+ 2
0 0 0 1	+ 1	+ 1	+ 1
0 0 0 0	+ 0	+ 0	+ 0
1 0 0 0	- 0	- 7	- 8
1 0 0 1	- 1	- 6	- 7
1 0 1 0	- 2	- 5	- 6
1 0 1 1	- 3	- 4	- 5
1 1 0 0	- 4	- 3	- 4
1 1 0 1	- 5	- 2	- 3
1 1 1 0	- 6	- 1	- 2
1 1 1 1	- 7	- 0	- 1

Figure 2.1. Binary, signed-integer representations.

Addition of Unsigned Integers

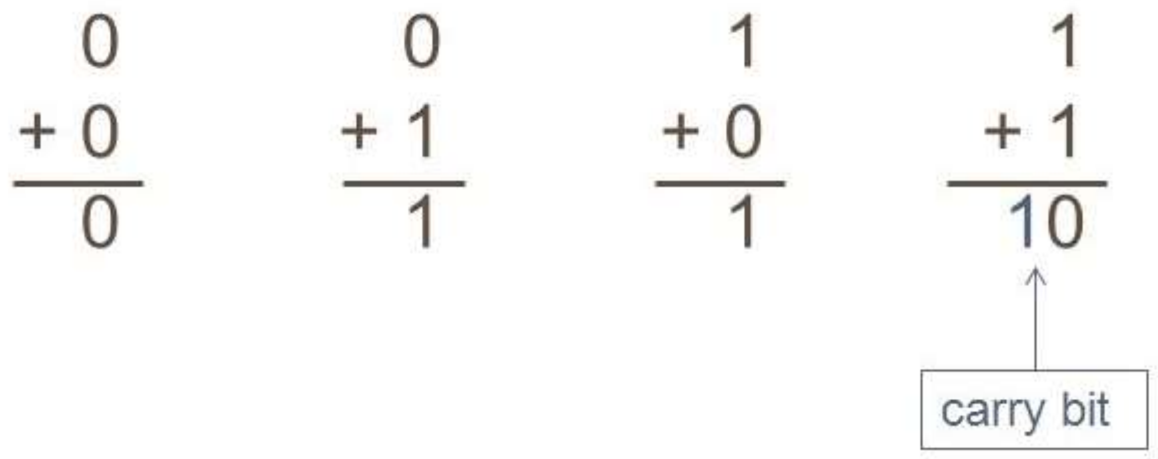
Addition of 1-bit numbers. The sum of 1 and 1 is the 2-bit vector 10, which represents the value 2. We say that the sum is 0 and the carry-out is 1.

In order to add multiple-bit numbers, we use a method analogous to that used for manual computation with decimal numbers.

We add bit pairs starting from the low-order (right) end of the bit vectors, propagating carries toward the high-order (left) end.

The carry-out from a bit pair becomes the carry-in to the next bit pair to the left. The carry-in must be added to a bit pair in generating the sum and carry-out at that position.

For example, if both bits of a pair are 1 and the carry-in is 1, then the sum is 1 and the carry-out is 1, which represents the value 3.

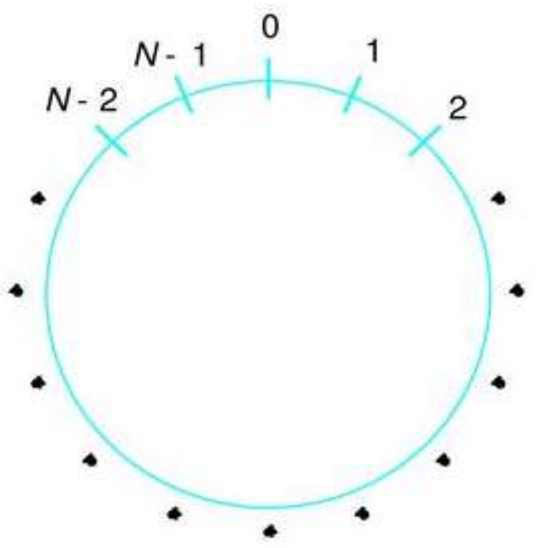


Addition and Subtraction of Signed Integers

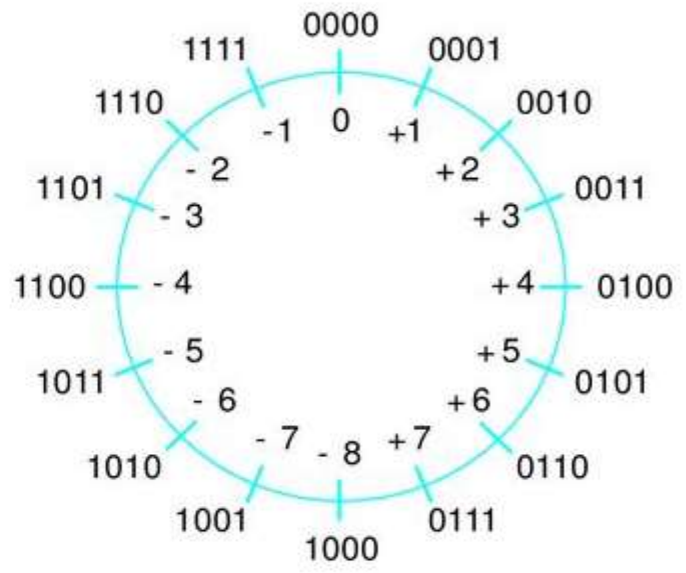
To add two numbers, add their n-bit representations, ignoring the carry-out bit from the most significant bit (MSB) position. The sum will be the algebraically correct value in 2’s-complement representation if the actual result is in the range −2n−1 through +2n−1 − 1.

To subtract two numbers X and Y , that is, to perform X − Y , form the 2’s-complement of Y , then add it to X using the add rule. Again, the result will be the algebraically correct value in 2’s-complement representation if the actual result is in the range −2n−1 through +2n−1 − 1

- Range is -2^{n-1} to $2^{n-1}-1$



(a) Circle representation of integers mod *N*



(b) Mod 16 system for 2's-complement numbers

(a) $$\begin{array}{r} 0\,0\,1\,0 \\ +\ 0\,0\,1\,1 \\ \hline 0\,1\,0\,1 \end{array} \qquad \begin{array}{r} (+2) \\ (+3) \\ \hline (+5) \end{array}$$

(b) $$\begin{array}{r} 0\,1\,0\,0 \\ +\ 1\,0\,1\,0 \\ \hline 1\,1\,1\,0 \end{array} \qquad \begin{array}{r} (+4) \\ (-6) \\ \hline (-2) \end{array}$$

(c) $$\begin{array}{r} 1\,0\,1\,1 \\ +\ 1\,1\,1\,0 \\ \hline 1\,0\,0\,1 \end{array} \qquad \begin{array}{r} (-5) \\ (-2) \\ \hline (-7) \end{array}$$

(d) $$\begin{array}{r} 0\,1\,1\,1 \\ +\ 1\,1\,0\,1 \\ \hline 0\,1\,0\,0 \end{array} \qquad \begin{array}{r} (+7) \\ (-3) \\ \hline (+4) \end{array}$$

(e) $$\begin{array}{r} 1\,1\,0\,1 \\ -\ 1\,0\,0\,1 \\ \hline \end{array} \qquad \begin{array}{r} (-3) \\ (-7) \\ \hline \end{array} \qquad \Longrightarrow \qquad \begin{array}{r} 1\,1\,0\,1 \\ +\ 0\,1\,1\,1 \\ \hline 0\,1\,0\,0 \end{array} \qquad \begin{array}{r} \\ \\ \hline (+4) \end{array}$$

(f) $$\begin{array}{r} 0\,0\,1\,0 \\ -\ 0\,1\,0\,0 \\ \hline \end{array} \qquad \begin{array}{r} (+2) \\ (+4) \\ \hline \end{array} \qquad \Longrightarrow \qquad \begin{array}{r} 0\,0\,1\,0 \\ +\ 1\,1\,0\,0 \\ \hline 1\,1\,1\,0 \end{array} \qquad \begin{array}{r} \\ \\ \hline (-2) \end{array}$$

(g) $$\begin{array}{r} 0\,1\,1\,0 \\ -\ 0\,0\,1\,1 \\ \hline \end{array} \qquad \begin{array}{r} (+6) \\ (+3) \\ \hline \end{array} \qquad \Longrightarrow \qquad \begin{array}{r} 0\,1\,1\,0 \\ +\ 1\,1\,0\,1 \\ \hline 0\,0\,1\,1 \end{array} \qquad \begin{array}{r} \\ \\ \hline (+3) \end{array}$$

(h) $$\begin{array}{r} 1\,0\,0\,1 \\ -\ 1\,0\,1\,1 \\ \hline \end{array} \qquad \begin{array}{r} (-7) \\ (-5) \\ \hline \end{array} \qquad \Longrightarrow \qquad \begin{array}{r} 1\,0\,0\,1 \\ +\ 0\,1\,0\,1 \\ \hline 1\,1\,1\,0 \end{array} \qquad \begin{array}{r} \\ \\ \hline (-2) \end{array}$$

(i) $$\begin{array}{r} 1\,0\,0\,1 \\ -\ 0\,0\,0\,1 \\ \hline \end{array} \qquad \begin{array}{r} (-7) \\ (+1) \\ \hline \end{array} \qquad \Longrightarrow \qquad \begin{array}{r} 1\,0\,0\,1 \\ +\ 1\,1\,1\,1 \\ \hline 1\,0\,0\,0 \end{array} \qquad \begin{array}{r} \\ \\ \hline (-8) \end{array}$$

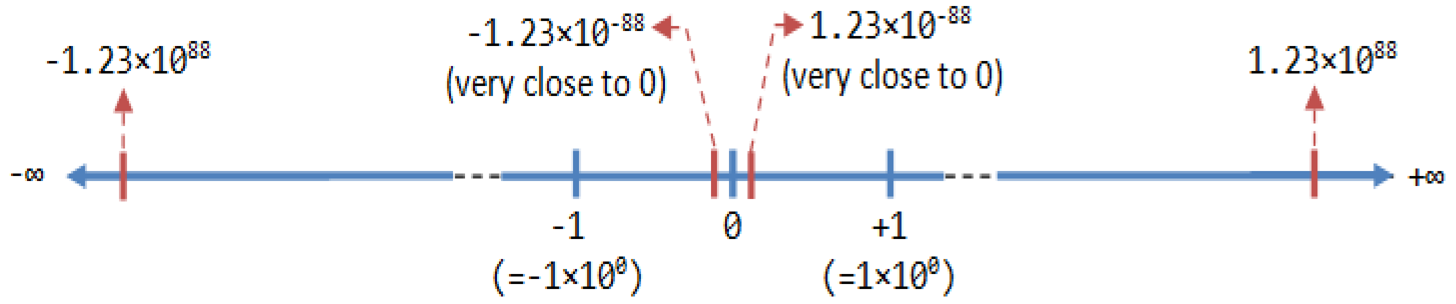
(j) $$\begin{array}{r} 0\,0\,1\,0 \\ -\ 1\,1\,0\,1 \\ \hline \end{array} \qquad \begin{array}{r} (+2) \\ (-3) \\ \hline \end{array} \qquad \Longrightarrow \qquad \begin{array}{r} 0\,0\,1\,0 \\ +\ 0\,0\,1\,1 \\ \hline 0\,1\,0\,1 \end{array} \qquad \begin{array}{r} \\ \\ \hline (+5) \end{array}$$

Figure 1.6 2's-complement Add and Subtract operations.

Floating-Point Numbers

A floating-point number (or real number) can represent a very large (1.23×10^88) or a very small (1.23×10^-88) value.

It could also represent very large negative number (-1.23×10^88) and very small negative number (-1.23×10^88), as well as zero, as :



Floating-point Numbers (Decimal)

A floating-point number is typically expressed in the scientific notation, with a fraction (F), and an exponent (E) of a certain radix (r), in the form of F×r^E.

Decimal numbers use radix of 10 (F×10^E); while binary numbers use radix of 2 (F×2^E).

Modern computers adopt IEEE 754 standard for representing floating-point numbers. There are two representation schemes: 32-bit single-precision and 64-bit double-precision.

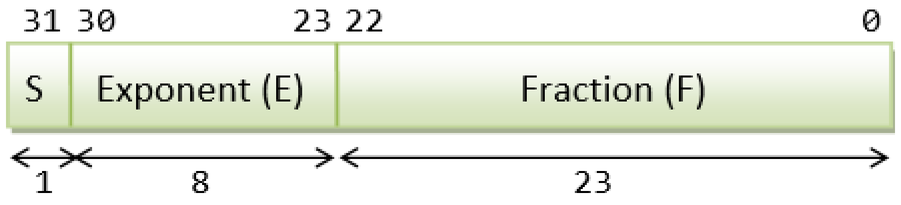
IEEE(Institute of Electrical and Electronics Engineers)-754 32-bit Single-Precision Floating-Point Numbers

In 32-bit single-precision floating-point representation:

The most significant bit is the sign bit (S), with 0 for positive numbers and 1 for negative numbers.

The following 8 bits represent exponent (E).

The remaining 23 bits represents fraction (F).



32-bit Single-Precision Floating-point Number

Normalized Form

Let's illustrate with an example, suppose that the 32-bit pattern is 1 1000 0001 011 0000 0000 0000 0000 0000, with:

S = 1

E = 1000 0001

F = 011 0000 0000 0000 0000 0000

In the normalized form, the actual fraction is normalized with an implicit leading 1 in the form of 1.F. In this example, the actual fraction is 1.011 0000 0000 0000 0000 0000 = 1 + 1×2^-2 + 1×2^-3 = 1.375D.

The sign bit represents the sign of the number, with S=0 for positive and S=1 for negative number. In this example with S=1, this is a negative number, i.e., -1.375D.

1.5 Character Representation

Everything represented by a computer is represented by binary sequences.

A common non-integer needed to be represented is characters.

We use standard encodings (binary sequences) to represent characters.

The most common encoding scheme for characters is ASCII (American Standard Code for Information Interchange).

In ASCII, there are 128 characters to be represented, so each character is stored in one byte. Here is one representation of the ASCII character set.

We can divide the character set into different classes:

Alphanumeric characters, operators, punctuation symbols, and control characters are represented by 7-bit codes as shown in below Table.

Alphabetic characters including: lower case letters - 'a'..'z' o upper case letters - 'A'..'Z‘.

Digit characters - '0'..'9‘.

Punctuation characters - . , : ; ‘ “ etc.

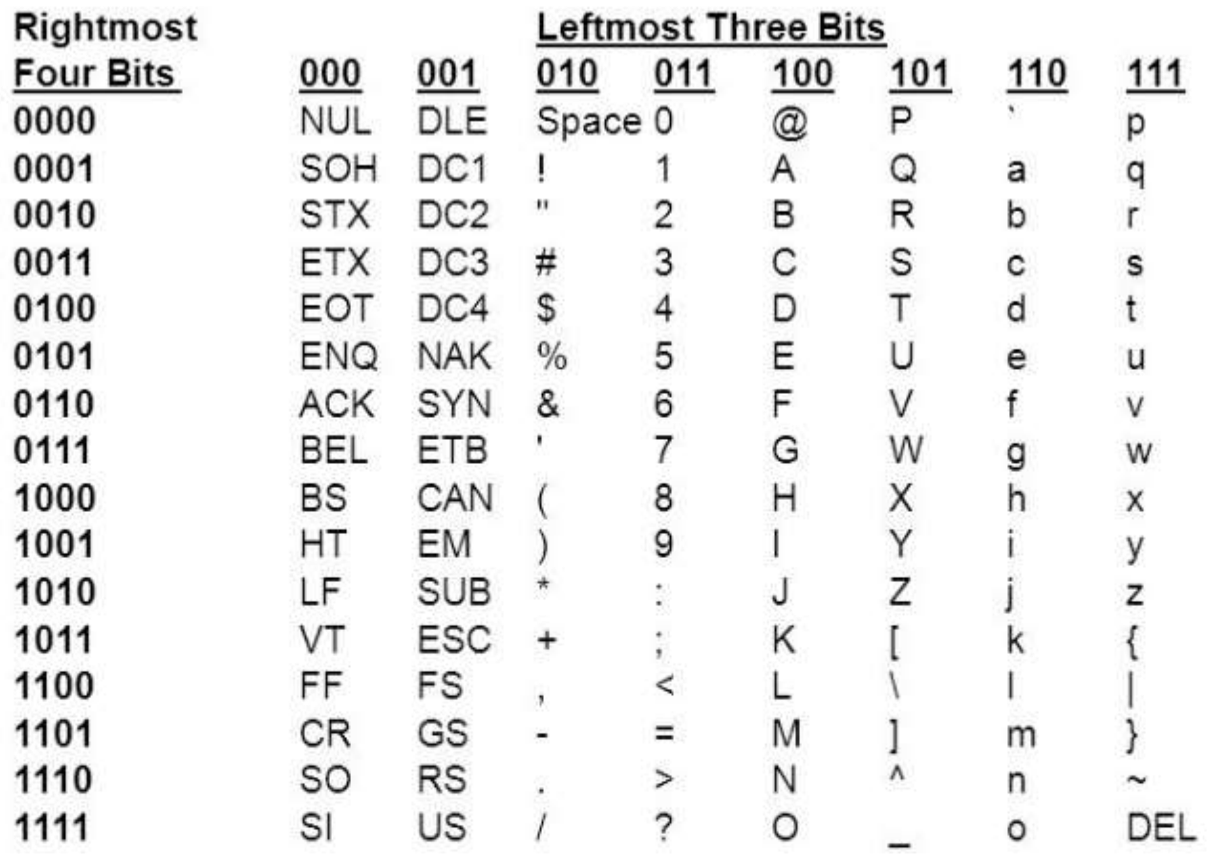
Special characters - '@',' < ', ' >', etc.

Control characters - '^A','^B', DC1 etc.

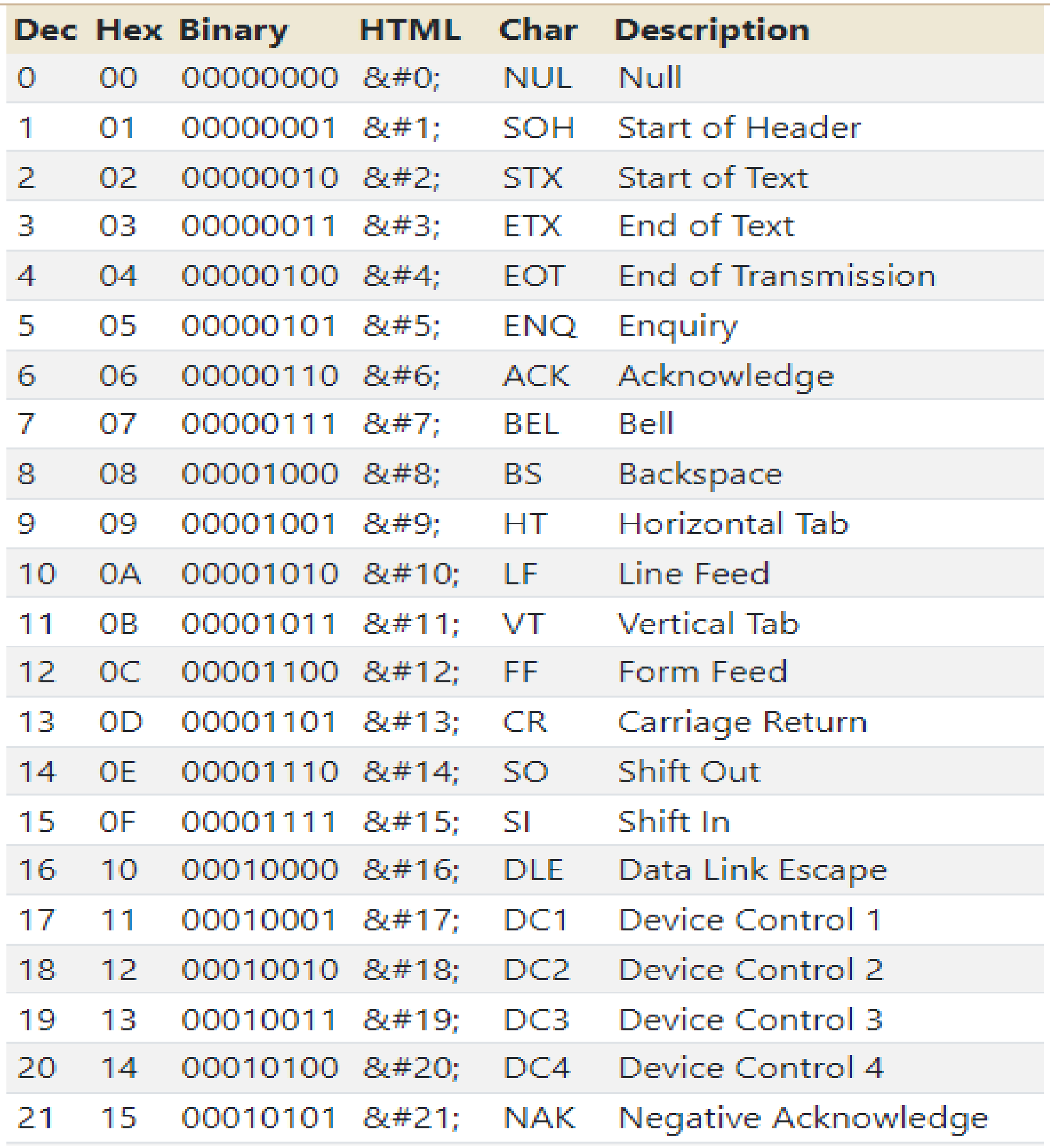
It is convenient to use an 8-bit byte to represent and store a character. The code occupies the low-order seven bits. The high-order bit is usually set to 0.

Table for ASCII values

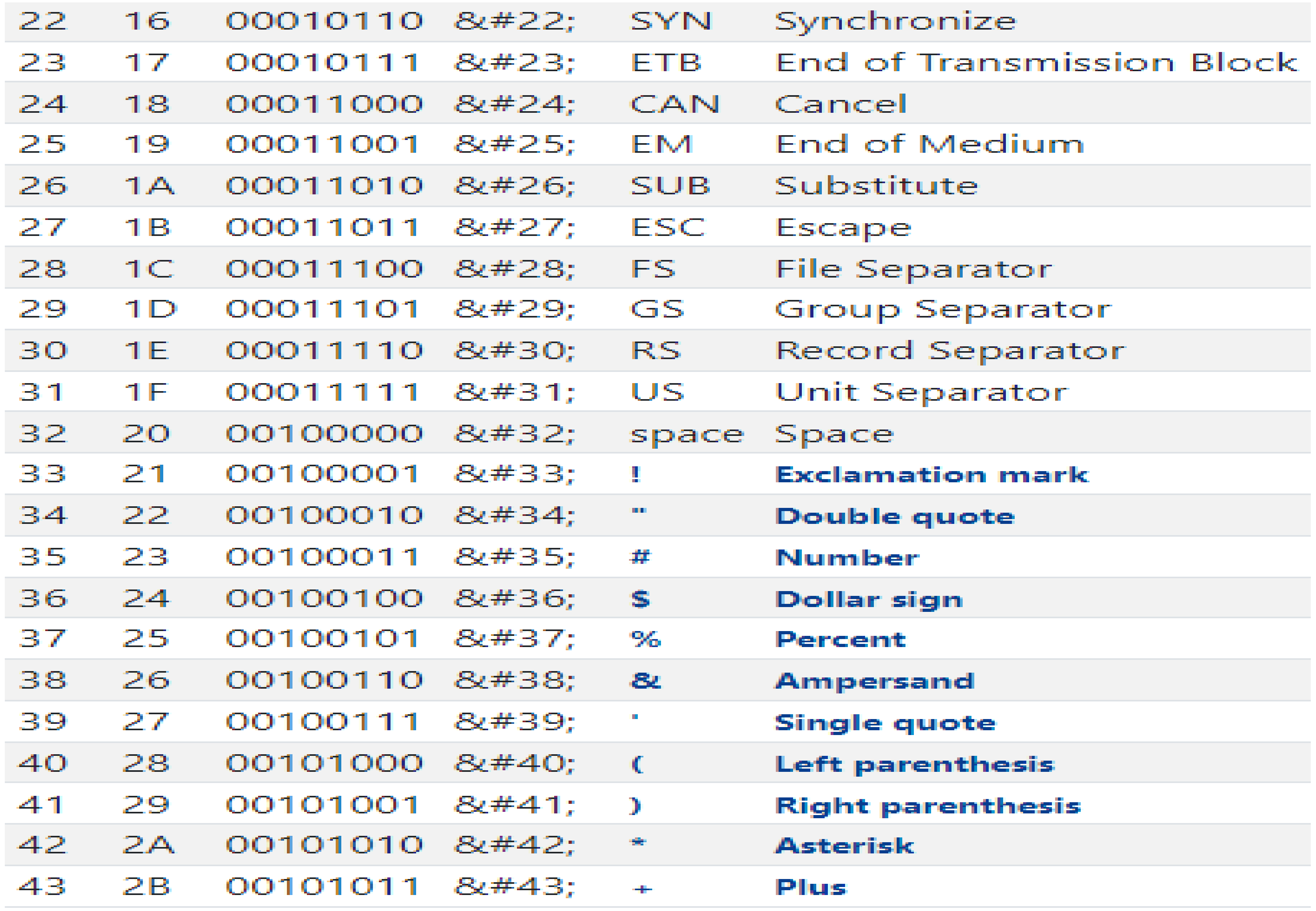
[..\..\ASCII rep.txt](..\..\ASCII rep.txt)



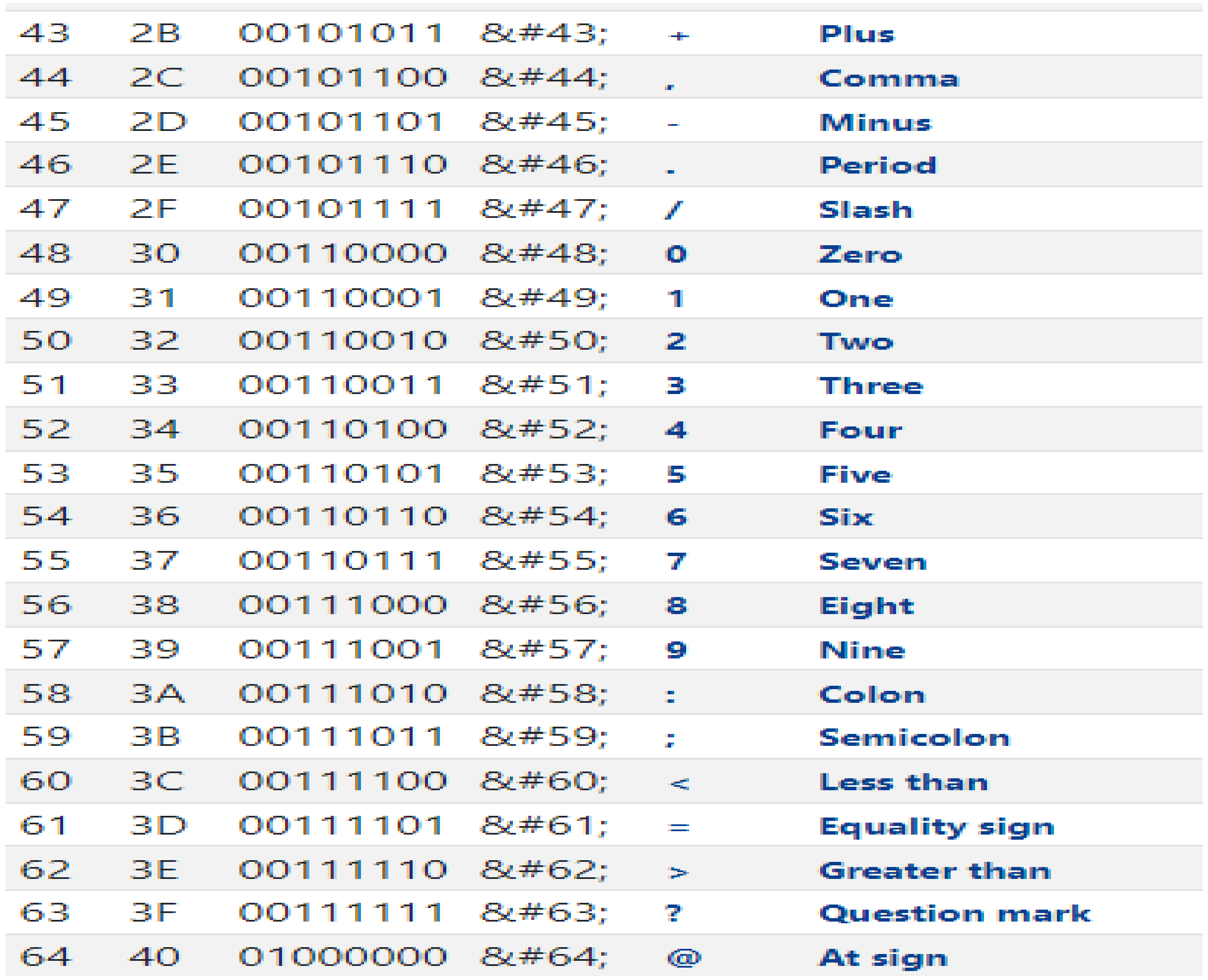
Rightmost			Leftmost Three Bits					
Four Bits	000	001	010	011	100	101	110	111
0000	NUL	DLE	Space	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL



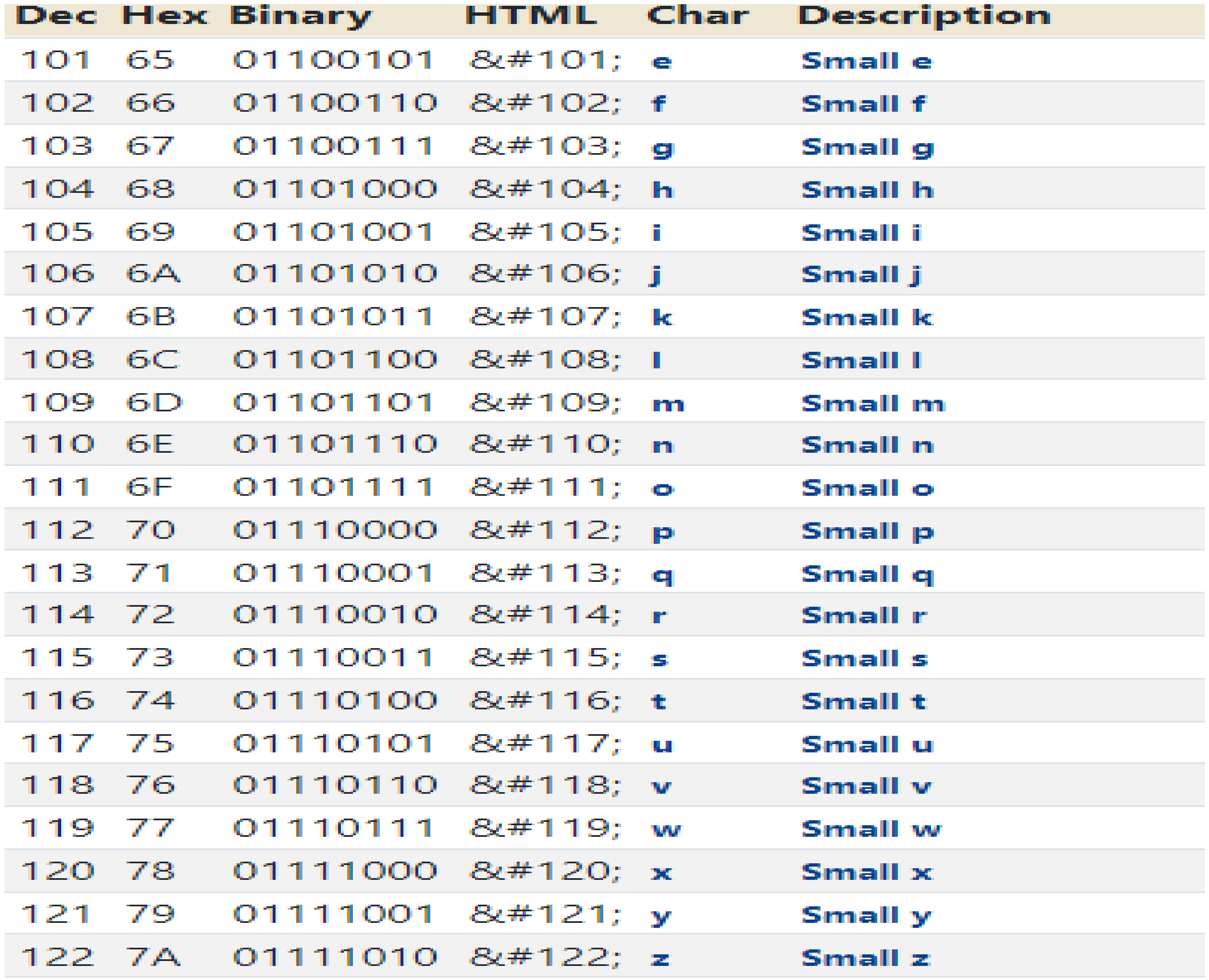
Dec	Hex	Binary	HTML	Char	Description
0	00	00000000	&#0;	NUL	Null
1	01	00000001	&#1;	SOH	Start of Header
2	02	00000010	&#2;	STX	Start of Text
3	03	00000011	&#3;	ETX	End of Text
4	04	00000100	&#4;	EOT	End of Transmission
5	05	00000101	&#5;	ENQ	Enquiry
6	06	00000110	&#6;	ACK	Acknowledge
7	07	00000111	&#7;	BEL	Bell
8	08	00001000	&#8;	BS	Backspace
9	09	00001001	&#9;	HT	Horizontal Tab
10	0A	00001010	&#10;	LF	Line Feed
11	0B	00001011	&#11;	VT	Vertical Tab
12	0C	00001100	&#12;	FF	Form Feed
13	0D	00001101	&#13;	CR	Carriage Return
14	0E	00001110	&#14;	SO	Shift Out
15	0F	00001111	&#15;	SI	Shift In
16	10	00010000	&#16;	DLE	Data Link Escape
17	11	00010001	&#17;	DC1	Device Control 1
18	12	00010010	&#18;	DC2	Device Control 2
19	13	00010011	&#19;	DC3	Device Control 3
20	14	00010100	&#20;	DC4	Device Control 4
21	15	00010101	&#21;	NAK	Negative Acknowledge



22	16	00010110	&#22;	SYN	Synchronize
23	17	00010111	&#23;	ETB	End of Transmission Block
24	18	00011000	&#24;	CAN	Cancel
25	19	00011001	&#25;	EM	End of Medium
26	1A	00011010	&#26;	SUB	Substitute
27	1B	00011011	&#27;	ESC	Escape
28	1C	00011100	&#28;	FS	File Separator
29	1D	00011101	&#29;	GS	Group Separator
30	1E	00011110	&#30;	RS	Record Separator
31	1F	00011111	&#31;	US	Unit Separator
32	20	00100000	&#32;	space	Space
33	21	00100001	&#33;	!	**Exclamation mark**
34	22	00100010	&#34;	"	**Double quote**
35	23	00100011	&#35;	#	**Number**
36	24	00100100	&#36;	$	**Dollar sign**
37	25	00100101	&#37;	%	**Percent**
38	26	00100110	&#38;	&	**Ampersand**
39	27	00100111	&#39;	'	**Single quote**
40	28	00101000	&#40;	(	**Left parenthesis**
41	29	00101001	&#41;	)	**Right parenthesis**
42	2A	00101010	&#42;	*	**Asterisk**
43	2B	00101011	&#43;	+	**Plus**



43	2B	00101011	&#43;	+	**Plus**
44	2C	00101100	&#44;	,	**Comma**
45	2D	00101101	&#45;	-	**Minus**
46	2E	00101110	&#46;	.	**Period**
47	2F	00101111	&#47;	/	**Slash**
48	30	00110000	&#48;	0	**Zero**
49	31	00110001	&#49;	1	**One**
50	32	00110010	&#50;	2	**Two**
51	33	00110011	&#51;	3	**Three**
52	34	00110100	&#52;	4	**Four**
53	35	00110101	&#53;	5	**Five**
54	36	00110110	&#54;	6	**Six**
55	37	00110111	&#55;	7	**Seven**
56	38	00111000	&#56;	8	**Eight**
57	39	00111001	&#57;	9	**Nine**
58	3A	00111010	&#58;	:	**Colon**
59	3B	00111011	&#59;	;	**Semicolon**
60	3C	00111100	&#60;	<	**Less than**
61	3D	00111101	&#61;	=	**Equality sign**
62	3E	00111110	&#62;	>	**Greater than**
63	3F	00111111	&#63;	?	**Question mark**
64	40	01000000	&#64;	@	**At sign**



Dec	Hex	Binary	HTML	Char	Description
101	65	01100101	&#101;	e	Small e
102	66	01100110	&#102;	f	Small f
103	67	01100111	&#103;	g	Small g
104	68	01101000	&#104;	h	Small h
105	69	01101001	&#105;	i	Small i
106	6A	01101010	&#106;	j	Small j
107	6B	01101011	&#107;	k	Small k
108	6C	01101100	&#108;	l	Small l
109	6D	01101101	&#109;	m	Small m
110	6E	01101110	&#110;	n	Small n
111	6F	01101111	&#111;	o	Small o
112	70	01110000	&#112;	p	Small p
113	71	01110001	&#113;	q	Small q
114	72	01110010	&#114;	r	Small r
115	73	01110011	&#115;	s	Small s
116	74	01110100	&#116;	t	Small t
117	75	01110101	&#117;	u	Small u
118	76	01110110	&#118;	v	Small v
119	77	01110111	&#119;	w	Small w
120	78	01111000	&#120;	x	Small x
121	79	01111001	&#121;	y	Small y
122	7A	01111010	&#122;	z	Small z

1.6 PERFORMANCE

- The most important measure of the performance of a computer is how quickly it can execute programs.
- The speed with which a computer executes programs is affected by the **design** of its hardware and its machine language instructions. Because programs are usually written in a high-level language.
- performance is also affected by the **compiler** that translates programs into machine language. For best performance, it is necessary to design the compiler, the machine instruction set, and the hardware in a coordinated way.
- The processor and a relatively small **cache memory** can be fabricated on a singie integrated circuit chip.
- The internal speed of performing the basic steps of instruction processing on such chips is very high and is considerably faster than the speed at which instructions and data can be fetched from the main memory.
- A program will be executed faster if the movement of instructions and data between the main memory an a the processor is minimized, which is achieved by using the cache.

- For example, Suppose a number of instructions are executed repeatedly over a short period of time, as happens in a program loop. If these instructions are available in the cache, they can be fetched quickly during the period of repeated use.

Technology

The technology of Very Large Scale Integration (VLSI) that is used to fabricate the electronic circuits for a processor on a single chip is a critical factor in the speed of execution of machine instructions.

The speed of switching between the 0 and 1 states in logic circuits is largely determined by the size of the transistors that implement the circuits.

Smaller transistors switch faster. Advances in fabrication technology over several decades have reduced transistor sizes dramatically.

This has two advantages: instructions can be executed faster, and more transistors can be placed on a chip, leading to more logic functionality and more memory storage capacity.

Parallelism

Parallelism refers to techniques to make programs faster by performing several computations at the same time.

This requires hardware with multiple processing units. In many cases the sub-computations are of the same structure, but this is not necessary.

Performance can be increased by performing a number of operations in parallel.

Parallelism can be implemented on many different levels.

Instruction-level Parallelism

Multicore Processors

Multiprocessors

Instruction-level Parallelism

The simplest way to execute a sequence of instructions in a processor is to complete all steps of the current instruction before starting the steps of the next instruction.

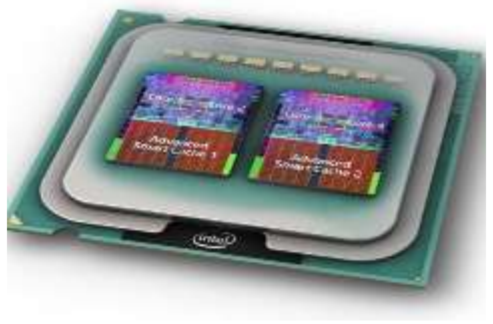
If we overlap the execution of the steps of successive instructions, total execution time will be reduced.

For example, the next instruction could be fetched from memory at the same time that an arithmetic operation is being performed on the register operands of the current instruction. This form of parallelism is called pipelining

Multicore Processors

A multicore processor is an integrated circuit that has two or more processor cores attached for enhanced performance and reduced power consumption. These processors also enable more efficient simultaneous processing of multiple tasks.

In technical literature, the term core is used for each of these processors. The term processor is then used for the complete chip. Hence, we have the terminology dual-core, quad-core, and octo-core processors for chips that have two, four, and eight cores, respectively.



Multiprocessors

A multiprocessor is a computer system with two or more central processing units (CPUs), with each one sharing the common main memory as well as the peripherals. This helps in simultaneous processing of programs.

Computer systems may contain many processors, each possibly containing multiple cores. Such systems are called multiprocessors.

These systems either execute a number of different application tasks in parallel, or they execute subtasks of a single large task in parallel.

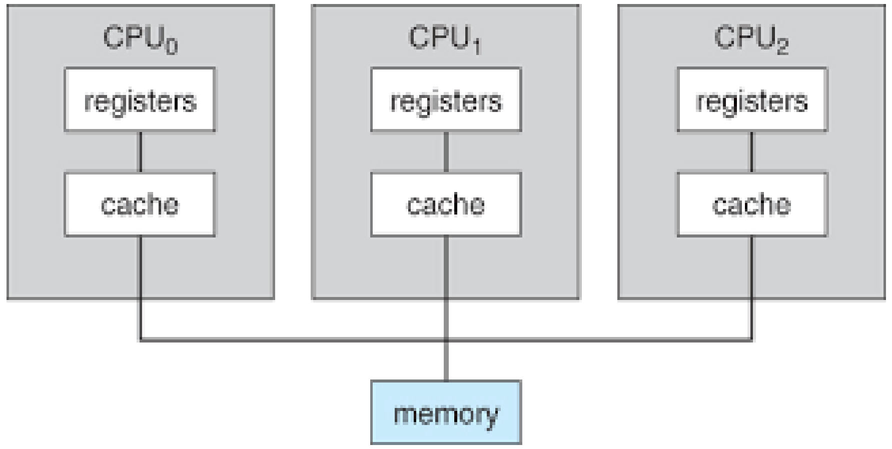
The high performance of these systems comes with much higher complexity and cost, arising from the use of multiple processors and memory units, along with more complex interconnection networks.

In contrast to multiprocessor systems, it is also possible to use an interconnected group of complete computers to achieve high total computational power.

The computers normally have access only to their own memory units.

When the tasks they are executing need to share data, they do so by exchanging messages over a communication network.

This property distinguishes them from shared-memory multiprocessors, leading to the name message passing multi-computers



Problem: Convert the following pairs of decimal numbers to 5-bit 2's-complement numbers, then perform addition and subtraction on each pair. Indicate whether or not overflow occurs for each case.

Example

(*a*) 7 and 13

(*b*) −12 and 9

Solution: The conversion and operations are:

(*a*) $7_{10} = 00111_2$ and $13_{10} = 01101_2$

Adding these two positive numbers, we obtain 10100, which is a negative number. Therefore, overflow has occurred.

To subtract them, we first form the 2's-complement of 01101, which is 10011. Then we perform addition with 00111 to obtain 11010, which is -6_{10}, the correct answer.

(*b*) $-12_{10} = 10100_2$ and $9_{10} = 01001_2$

Adding these two numbers, we obtain $11101 = -3_{10}$, the correct answer.

To subtract them, we first form the 2's-complement of 01001, which is 10111. Then we perform addition of the two negative numbers 10100 and 10111 to obtain 01011, which is a positive number. Therefore, overflow has occurred.

[M] Convert the following pairs of decimal numbers to 5-bit 2's-complement numbers, then add them. State whether or not overflow occurs in each case.

(*a*) 4 and 11

(*b*) 6 and 14

(*c*) −13 and 12

(*d*) −4 and 8

(*e*) −2 and −9

(*f*) −9 and −14