

## 1. Introductions to computers

- Computer is an electrical device that receives input stores or processes the input as per user instruction and provides output desired format.
- Computer input is called data.
- Output obtained after processing data based on user instruction is called information.
- Raw facts and figures which be processed using arithmetic and logic operations to obtain information are called data
- Process that can be applied to data are two types:
  - Arithmetic operation
  - Logic operation

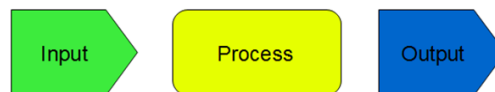


FIGURE (1)

## 2. Data processing

The activity of processing data using a computer is called data processing

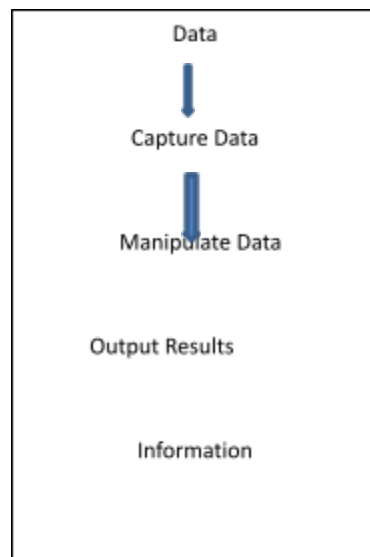


FIGURE (2)

**Data** is raw material used as input and information is processed data obtained as output of data processing

### 3. Characteristics of Computers

1. **Automatic:** Given a job, computer can work on it automatically without human interventions
2. **Speed:** Computer can perform data processing jobs very fast, usually measured in microseconds (10<sup>-6</sup>), nanoseconds (10<sup>-9</sup>), and picoseconds (10<sup>-12</sup>)
3. **Accuracy:** Accuracy of a computer is consistently high and the degree of its accuracy depends upon its design. Computer errors caused due to incorrect input data or unreliable programs are often referred to as GarbageIn-Garbage-Out (GIGO).
4. **Diligence:** Computer is free from monotony, tiredness, and lack of concentration. It can continuously work for hours without creating any error and without grumbling
5. **Versatility:** Computer is capable of performing almost any task, if the task can be reduced to a finite series of logical steps
6. **Power of Remembering:** Computer can store and recall any amount of information because of its secondary storage capability. It forgets or loses certain information only when it is asked to do so
7. **No I.Q.:** A computer does only what it is programmed to do. It cannot take its own decision in this regard
8. **No Feelings:** Computers are devoid of emotions. Their judgement is based on the instructions given to them in the form of programs that are written by us (human beings)

### 4. Booting

Starting a computer or computer-embedded device is called booting.

Booting steps:

- Power on the power supply.
- Loading operating system into computer main memory.
- Keep all applications in a state of readiness in case needed by the user.

The first program or set of instructions running when the computer is switched on is called **Bios** or a basic input-output system.

There are two types of Booting:

- A. Hard Booting: System is started by switching on the power supply.
- B. Soft Booting: The system is already running and needs to be restarted or rebooted. (Already Operating System (OS) has loaded).

## **What is the Master Boot Record (MBR)?**

The Master Boot Record (MBR) is the information in the first sector of a hard disk or a removable drive. It identifies how and where the system's operating system (OS) is located in order to be booted (loaded) into the computer's main storage or random access memory (RAM).

### **5. Main computer part and computer type:**

1. Motherboard
2. CPU
3. RAM
4. Hard-drive
5. Other memory (CD, DVD, Floppy)
6. Expansion cards (video, audio, NIC, etc)

### **1. Motherboard:**

A motherboard is the central or primary circuit board making up a complex electronic system, such as a computer. The motherboard contains the connectors for attaching additional boards.

- The motherboard = your spine.
- The motherboard acts as a connecting point between all other parts of your computer.
- Expansion cards plug into the motherboard.

- Hard drive and other drives connect to the motherboard.
- Power supply connects to the motherboard.
- CPU and RAM connect to the motherboard.

## 2. CPU:

The CPU or simply processor is the component in a computer that interprets instructions and processes data

- Central Processing Unit (CPU)
- CPU = your brain
- This is the component that is responsible for making decisions (processing) in your computer.
- It makes calculations
- Tells other components (ie: RAM) what to do.

## 3. RAM:

The area of the computer's memory that can be read from and written to (changed). All RAM locations are equally accessible at any time in any order. The components of RAM are erased when the computer is turned off.

- Random Access Memory (RAM)
- RAM = nervous system?

- RAM temporarily stores information
- Electronic memory, so it is very fast.
- Allows for multiple applications to be open at once.
- Memory is lost when power is shut off.

#### 4. Hard drive:

The hard drive is the primary storage unit of the computer. It is where the operating system, applications, and files are kept. Hard drive space is typically measured in Gigabytes (GB); the larger the number, the more programs, etc. you can hold on your computer.

- Permanent storage device for your computer
- Hard drive = brain (the storage centre, not decision- making)
- Data is stored on a magnetic disk that spins (much like a CD) to access the information.
- Most hard drives spin at about 7200 rpm, so is relatively slow compared to RAM.

#### 5. Expansion Cards:

- **Audio** – card that allows you to use audio devices (earphones, speakers)

- **Video** – card that allows you to use video on a computer (now, usually, built into the motherboard). The better the video card, the better the graphics on your computer.
- **NIC** – Network Interface Card. This allows you to connect to a network. It allows you to communicate with other computers.

## 6. BIOS:

The Basic Input/output System is installed on the computer's motherboard. It controls the most basic operations and is responsible for starting your computer up and initializing the hardware.

- Checks hardware on startup
- Loads operating system

## Basic organization of a computer

### 1- Basic operation of computer system

The five basic operations of a computer system:-

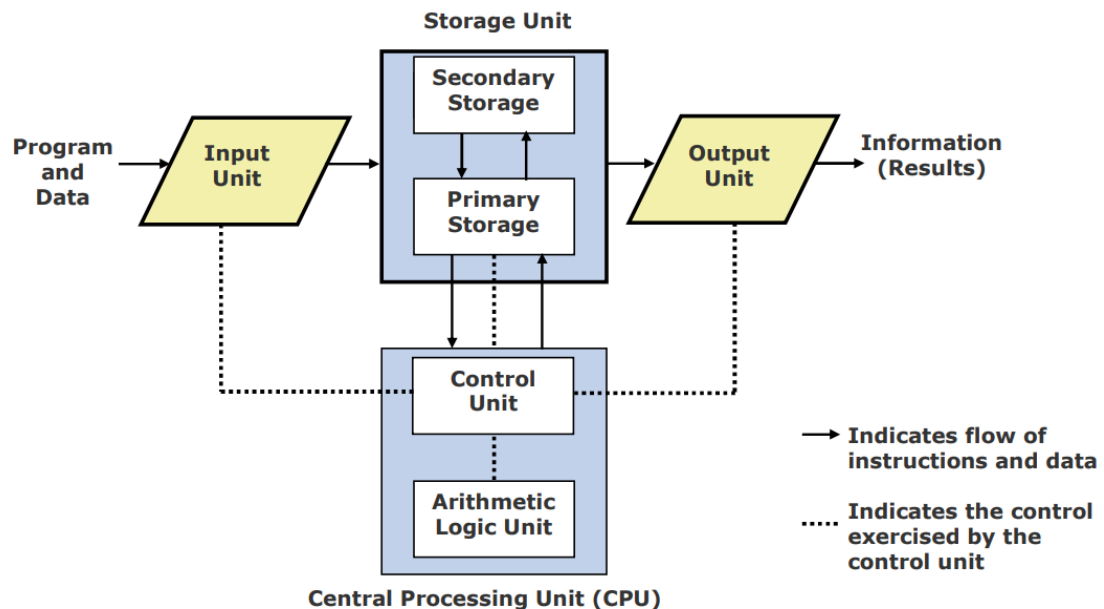
**Inputting:** - The process of entering data and instructions into the computer system.

**Storing:** - Saving data and instructions to make them readily available for initial or additional processing whenever required.

**Processing:** - Performing arithmetic operations (add, subtract, multiply, divide, etc.) or logical operations (comparisons like equal to, less than, greater than, etc.) on data to convert them into useful information.

**Outputting:** - The process of producing useful information or results for the user such as a printed report or visual display.

**Controlling:** - Directing the manner and sequence in which all of the above operations are performed.



### 2- input unit

An input unit of a computer system performs the following functions:

1. It accepts (or reads) instructions and data from outside world
2. It converts these instructions and data in computer acceptable form

3. It supplies the converted instructions and data to the computer system for further processing

### 3- output unit

An output unit of a computer system performs the following functions:

1. It accepts the results produced by the computer, which are in coded form and hence, cannot be easily understood by us
2. It converts these coded results to human acceptable (readable) form
3. It supplies the converted results to outside world

### 4- storage unit

The storage unit of a computer system holds (or stores) the following :

1. Data and instructions required for processing (received from input devices)
2. Intermediate results of processing
3. Final results of processing, before they are released to an output device

Two types of storage: -

#### ➤ **Primary storage**

- Used to hold running program instructions
- Used to hold data, intermediate results, and results of ongoing processing of job(s)
- Fast in operation
- Small Capacity
- Expensive & Volatile (loses data on power dissipation)

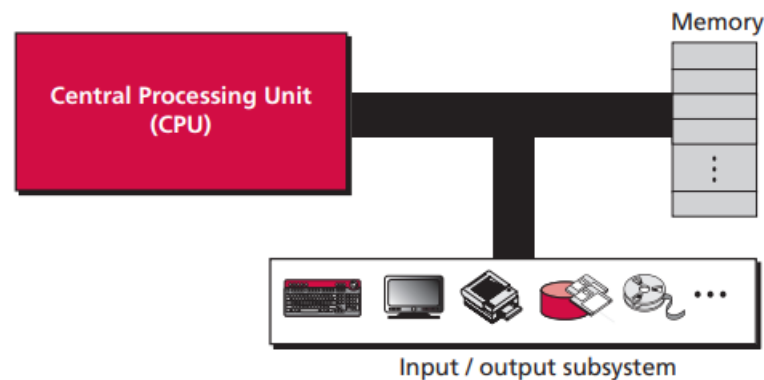
#### ➤ **Secondary storage**

- Used to hold stored program instructions
- Used to hold data and information of stored jobs
- Slower than primary storage
- Large Capacity
- Lot cheaper than primary storage
- Retains data even without power



## Central processing unit

The central processing unit (CPU) performs operations on data. In most architectures it has three parts: an arithmetic logic unit (ALU), a control unit, and a set of registers, figure (2.1).



### 1- Arithmetic Logic Unit(ALU)

Arithmetic Logic Unit of a computer system is the place where the actual executions of instructions take place during processing operation. Its performs:

- ✓ **Logic operations** We discussed several logic operations, such as NOT, AND, OR, and XOR. These operations treat the input data as bit patterns and the result of the operation is also a bit pattern.
- ✓ **Shift operations** We discussed two groups of shift operations on data. logical shift operations and arithmetic shift operations. Logical shift operations are used to shift bit patterns to the left or right, while arithmetic operations are applied to integers. Their main purpose is to divide or multiply integers by two.
- ✓ **Arithmetic operation** We discussed some arithmetic operations on integers. We mentioned that some operations can be implemented more efficiently in hardware.

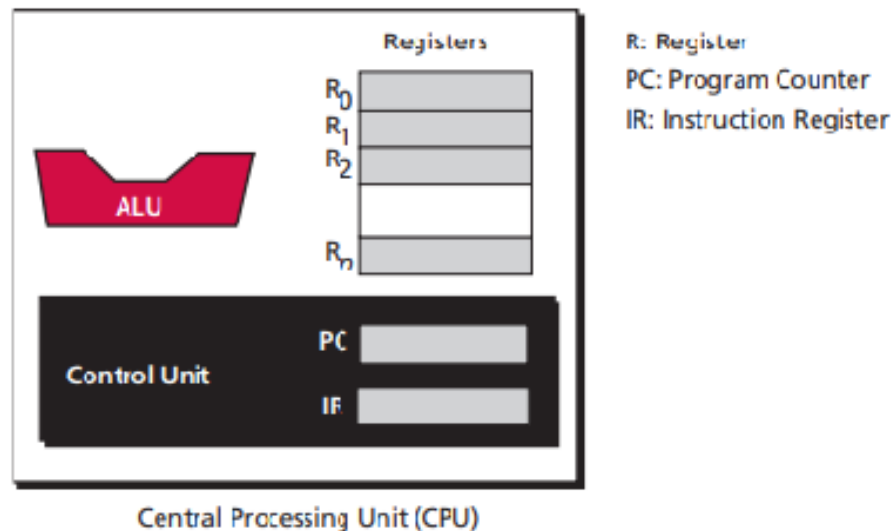
## 2- Registers

Registers are fast stand-alone storage locations that hold data temporarily. Multiple registers are needed to facilitate the operation of the CPU. Some of these registers are shown in Figure 2.2.

**Data registers** In the past computers had only a few data registers to hold the input data and the result of the operations. Today, computers use dozens of registers inside the CPU to speed up their operations, because complex operations are done using hardware instead of software. These require several registers to hold the intermediate results. Data registers are named R1 to Rn in Figure 2.2.

**Instruction registers** Today computers store not only data, but also programs, in their memory. The CPU is responsible for fetching instructions one by one from memory, storing them in the instruction register (IR in Figure 2.2.), decoding them, and executing them. We will discuss this issue later in the chapter. Program counter Another common register in the CPU is the program counter (PC in Figure 2.2.).

**The program counter** keeps track of the instruction currently being executed. After execution of the instruction, the counter is incremented to point to the address of the next instruction in memory



### **3- Control Unit(CU)**

The third part of any CPU is the control unit. The control unit controls the operation of each subsystem. Controlling is achieved through signals sent from the control unit to other subsystems.

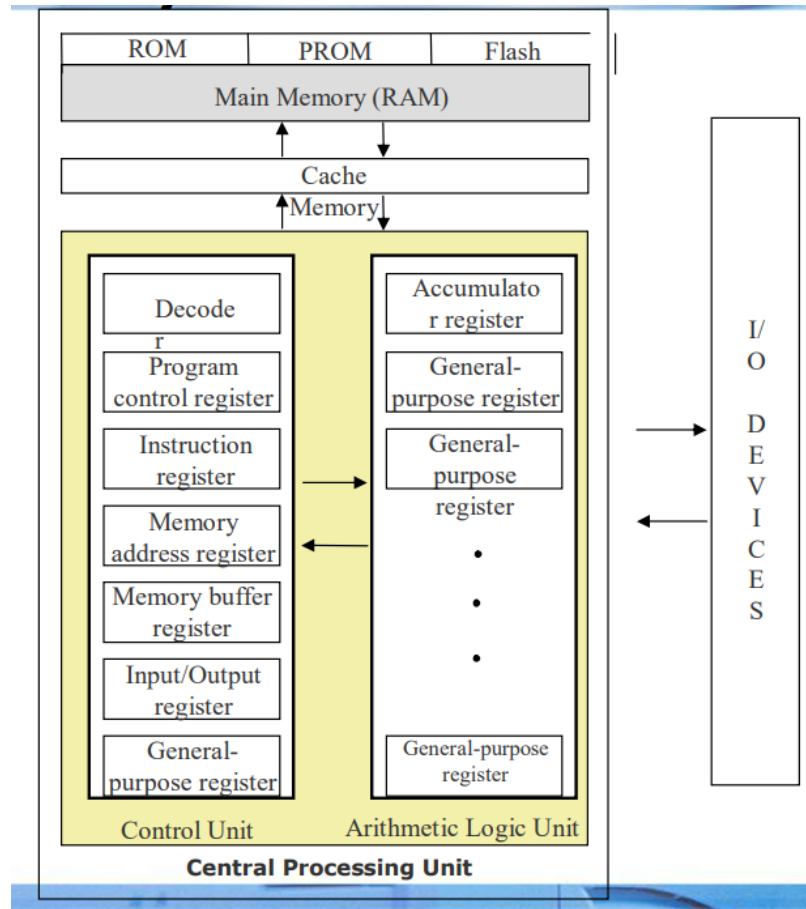
#### **The system concept**

A system has following three characteristics:

1. A system has more than one element
2. All elements of a system are logically related
3. All elements of a system are controlled in a manner to achieve the system goal

## Basic Processor of computer

**Central Processing Unit (CPU)** The CPU is the brain of a computer, containing all the circuitry needed to process input, store data, and output results. The CPU is constantly following instructions of computer programs that tell it which data to process and how to process it.



### Control Unit (CU)

- ❖ One of the two basic components of CPU
- ❖ Acts as the central nervous system of a computer system
- ❖ Selects and interprets program instructions, and coordinates execution
- ❖ Has some special purpose registers and a decoder to perform these activities

### Arithmetic Logic Unit (ALU)

- ❖ One of the two basic components of CPU.
- ❖ Actual execution of instructions takes place in ALU

- ❖ Has some special purpose registers
- ❖ Has necessary circuitry to carry out all the arithmetic and logic operations included in the CPU instruction set

### ***Instruction Set***

- ❖ CPU has built-in ability to execute a particular set of machine instructions, called its instruction set
- ❖ Most CPUs have 200 or more instructions (such as add, subtract, compare, etc.) in their instruction set
- ❖ CPUs made by different manufacturers have different instruction sets
- ❖ Manufacturers tend to group their CPUs into “families” having similar instruction sets
- ❖ New CPU whose instruction set includes instruction set of its predecessor CPU is said to be ***backward compatible*** with its predecessor

### ***Registers***

1. Special memory units, called registers, are used to hold information on a temporary basis as the instructions are interpreted and executed by the CPU
2. Registers are part of the CPU (not main memory) of a computer
3. The length of a register, sometimes called its word size, equals the number of bits it can store
4. With all other parameters being the same, a CPU with 32-bit registers can process data twice as fast as one with 16-bit registers
5. a twice larger than one with 16-bit registers

### **Function of commonly used registers**

<b>SQ</b>	<b>Name of Register</b>	<b>Function</b>
1	Memory Address (MAR)	Holds address of the active memory location
2	Memory Buffer (MBR)	Holds contents of the accessed (read/written) memory word
3	Program Control (PC)	Holds address of the next instruction to be executed
4	Accumulator (A)	Holds data to be operated upon, intermediate results, and the results

5	Instruction (I)	Holds an instruction while it is being executed
6	Input/Output (I/O)	Used to communicate with the I/O devices

### *Processor Speed*

- ❖ Computer has a built-in system clock that emits millions of regularly spaced electric pulses per second (known as clock cycles)
- ❖ It takes one cycle to perform a basic operation, such as moving a byte of data from one memory location to another
- ❖ Normally, several clock cycles are required to fetch, decode, and execute a single program instruction
- ❖ Hence, shorter the clock cycle, faster the processor
- ❖ Clock speed (number of clock cycles per second) is measured in Megahertz ( $10^6$  cycles/sec) or Gigahertz ( $10^9$  cycles/sec)

### *Types of Processor*

Type of Architecture	Features	Usage
CISC (Complex Instruction Set Computer)	<ul style="list-style-type: none"> <li>§ Large instruction set</li> <li>§ Variable-length instructions</li> <li>§ Variety of addressing modes</li> <li>§ Complex &amp; expensive to produce</li> </ul>	Mostly used in personal computers
RISC (Reduced Instruction Set Computer)	<ul style="list-style-type: none"> <li>§ Small instruction set</li> <li>§ Fixed-length instructions</li> <li>§ Reduced references to memory to retrieve operands</li> </ul>	Mostly used in workstations
EPIC (Explicitly Parallel Instruction Computing)	<ul style="list-style-type: none"> <li>§ Allows software to communicate explicitly to the processor when operations are parallel</li> <li>§ Uses tighter coupling between the compiler and the processor</li> <li>§ Enables compiler to extract maximum parallelism in the</li> </ul>	Mostly used in high-end servers and workstations

	original code, and explicitly describe it to the processor	
Multi-Core Processor	<ul style="list-style-type: none"> <li>§ Processor chip has multiple cooler-running, more energy efficient processing cores</li> <li>§ Improve overall performance by handling more work in parallel</li> <li>§ can share architectural components, such as memory elements and memory management</li> </ul>	Mostly used in high-end servers and workstations

### ***Main Memory***

§ Every computer has a temporary storage built into the computer hardware

§ It stores instructions and data of a program mainly when the program is being executed by the CPU.

§ This temporary storage is known as main memory, primary storage, or simply memory.

§ Physically, it consists of some chips either on the motherboard or on a small circuit board attached to the motherboard of a computer

§ It has random access property.

§ It is volatile.

## Memory Architecture of a Computer

### Main memory

**Main memory** is the second major subsystem in a computer (Figure 4.1). It consists of a collection of storage locations, each with a unique identifier, called an address. Data is transferred to and from memory in groups of bits called words. A word can be a group of 8 bits, 16 bits, 32 bits, or 64 bits (and growing). If the word is 8 bits, it is referred to as a byte. The term ‘byte’ is so common in computer science that sometimes a 16-bit word is referred to as a 2-byte word, or a 32-bit word is referred to as a 4-byte word.

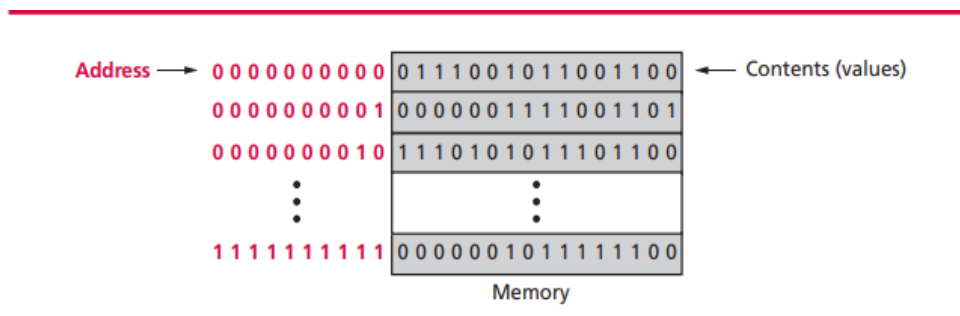


Figure (4.1) Main Memory

### Address space

To access a word in memory requires an identifier. Although programmers use a name to identify a word (or a collection of words), at the hardware level each word is identified by an address. The total number of uniquely identifiable locations in memory is called the address space. For example, a memory with 64 kilobytes and a word size of 1 byte has an address space that ranges from 0 to 65535. Table 4.1 shows the units used to refer to memory. Note that the terminology is misleading: it approximates the number of bytes in powers of 10, but the actual number of bytes is in powers of 2. Units in powers of 2 facilitates addressing.



Table (4.1) Memory Units

<i>Unit</i>	<i>Exact Number of Bytes</i>	<i>Approximation</i>
kilobyte	$2^{10}$ (1024) bytes	$10^3$ bytes
megabyte	$2^{20}$ (1 048 576) bytes	$10^6$ bytes
gigabyte	$2^{30}$ (1 073 741 824) bytes	$10^9$ bytes
terabyte	$2^{40}$ bytes	$10^{12}$ bytes

The main memory organization

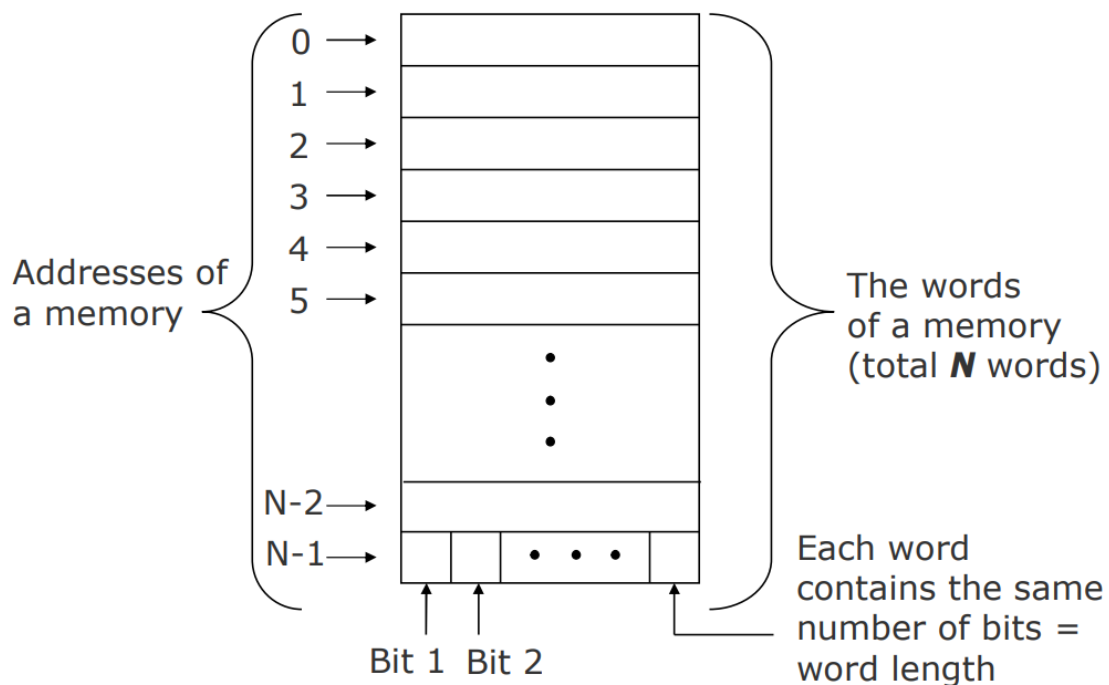


Figure (4.2) The main memory organization

### Addresses as bit patterns

Because computers operate by storing numbers as bit patterns, a memory address is also represented as a bit pattern. So if a computer has 64 kilobytes ( $2^{16}$ ) of memory with a word size of 1 byte, we need a bit pattern of 16 bits to define an address. In other words, the first location is referred to as address 0000000000000000 (address 0), and the last location is referred to as address 1111111111111111 (address 65535). In general, if a computer has  $N$  words of

memory, we need an unsigned integer of size  $\log_2 N$  bits to refer to each memory location.

### **Example**

A computer has 32 MB (megabytes) of memory. How many bits are needed to address any single byte in memory?

### **Solution**

The memory address space is 32 MB, or  $2^{25}$  ( $2^5 \times 2^{20}$ ). This means that we need  $\log_2 2^{25}$ , or 25 bits, to address each byte.

## **Memory types**

Two main types of memory exist: RAM and ROM

### 1- RAM

Random access memory (RAM) makes up most of the main memory in a computer. In a random access device, a data item can be accessed randomly—using the address of the memory location—without the need to access all data items located before it. However, the term is confusing, because ROM can also be accessed randomly. What distinguishes RAM from ROM is that RAM can be read from and written to. Another characteristic of RAM is that it is volatile: the information (program or data) is lost if the computer is powered down. RAM technology is divided into two broad categories:

- SRAM( Static )
- DRAM (Dynamic)

### 2- ROM

The contents of read-only memory (ROM) are written by the manufacturer, and the CPU can read from, but not write to, ROM. Its advantage is that it is nonvolatile—its contents are not lost if you turn off the computer. Normally, it is used for programs or data that must not be erased or changed even if you turn off the computer. For example, some computers come with ROM that holds the boot program that runs when we switch on the computer.

- PROM (programmable)

- EPROM (erasable programmable)
- EEPROM (electrically erasable programmable)

## Memory hierarchy

Computer users need a lot of memory, especially memory that is very fast and inexpensive. This demand is not always possible to satisfy—very fast memory is usually not cheap. A compromise needs to be made. The solution is hierarchical levels of memory (Figure 4.3). The hierarchy is based on the following:

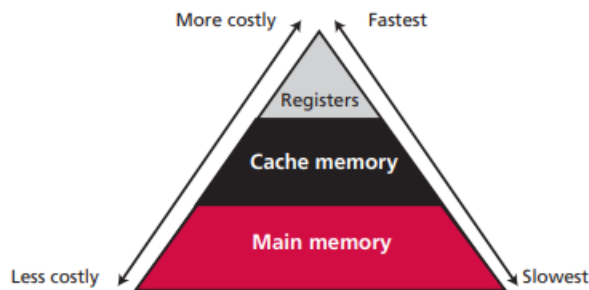


Figure (4.3) Memory hierarchy

- ✓ Use a very small amount of costly high-speed memory where speed is crucial. The registers inside the CPU are of this type.
- ✓ Use a moderate amount of medium-speed memory to store data that is accessed often. Cache memory, discussed next, is of this type.
- ✓ Use a large amount of low-speed memory for data that is accessed less often. Main memory is of this type

## Cache memory

Cache memory is faster than main memory but slower than the CPU and its registers. Cache memory, which is normally small in size, is placed between the CPU and main memory.

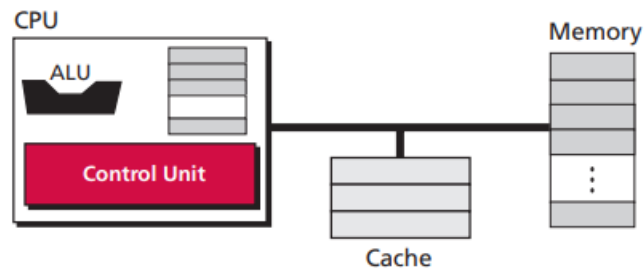


Figure (4.4) Cache Memory

Cache memory at any time contains a copy of a portion of main memory. When the CPU needs to access a word in main memory, it follows this procedure:

1. The CPU checks the cache.
2. If the word is there, it copies the word: if not, the CPU accesses main memory and copies a block of memory starting with the desired word. This block replaces the previous contents of cache memory.
3. The CPU accesses the cache and copies the word.

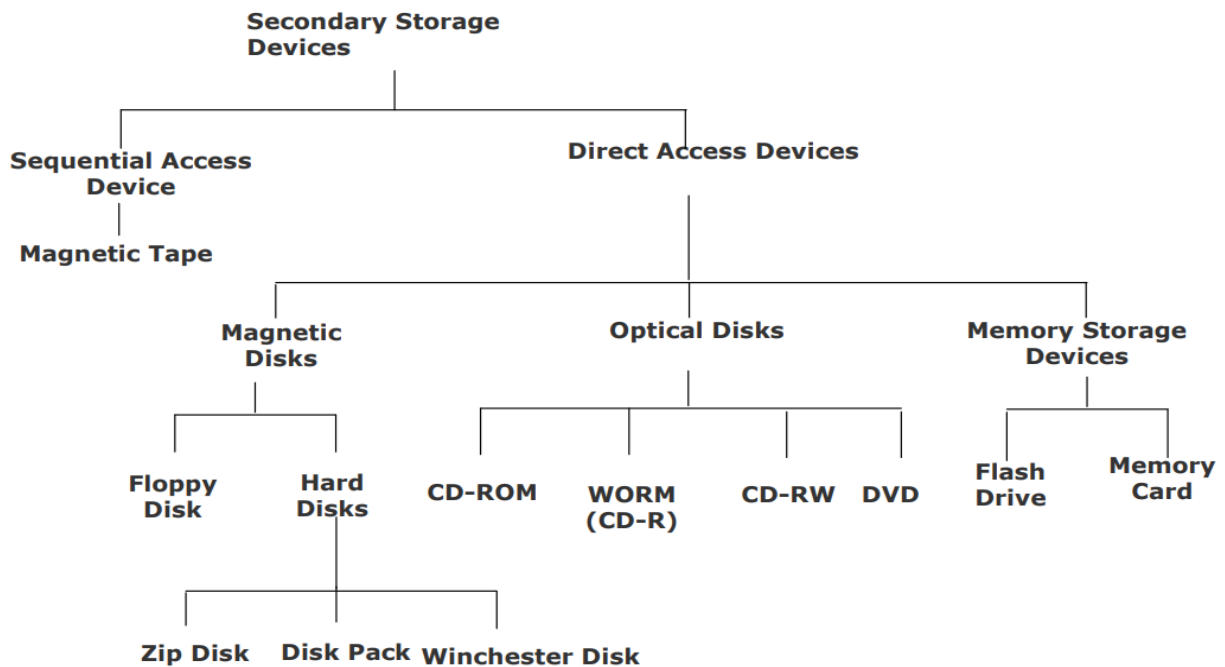
This procedure can expedite operations; if the word is in the cache, it is accessed immediately. If the word is not in the cache, the word and a whole block are copied to the cache. Since it is probable that the CPU, in its next cycle, will need to access the words following the first word, the existence of the cache speeds processing. We might wonder why cache memory is so efficient despite its small size. The answer lies in the '80–20 rule'. It has been observed that most computers typically spend 80 per cent of their time accessing only 20 per cent of the data. In other words, the same data is accessed over and over again. Cache memory, with its high speed, can hold this 20 per cent to make access faster at least 80 per cent of the time.

## Secondary storage devices

### Secondary storage

- ❖ Used in a computer system to overcome the limitations of primary storage
- ❖ Has virtually unlimited capacity because the cost per bit of storage is very low
- ❖ Has an operating speed far slower than that of the primary storage
- ❖ Used to store large volumes of data on a permanent basis
- ❖ Also known as auxiliary memory

### Classification of Commonly Used Secondary Storage Devices



### Sequential-access Storage Devices

- Arrival at the desired storage location may be preceded by sequencing through other locations
- Data can only be retrieved in the same sequence in which it is stored
- Access time varies according to the storage location of the information being accessed
- Suitable for sequential processing applications where most, if not all, of the data records need to be processed one after another

- Magnetic tape is a typical example of such a storage device

## Direct-access Storage Devices

- ❖ Devices where any storage location may be selected and accessed at random
- ❖ Permits access to individual information in a more direct or immediate manner
- ❖ Approximately equal access time is required for accessing information from any storage location
- ❖ Suitable for direct processing applications such as online ticket booking systems, on-line banking systems
- ❖ Magnetic, optical, and magneto-optical disks are typical examples of such a storage device

## Storage devices

Storage devices, although classified as I/O devices, can store large amounts of information to be retrieved at a later time. They are cheaper than main memory, and their contents are nonvolatile—that is, not erased when the power is turned off. They are sometimes referred to as auxiliary storage devices. We can categorize them as either magnetic or optical.

### Magnetic storage devices

Magnetic storage devices use magnetization to store bits of data. If a location is magnetized, it represents 1, if not magnetized, it represents 0.

#### 1- Magnetic disks

A magnetic disk consists of one or more disks stacked on top of each other. The disks are coated with a thin magnetic film. Information is stored on and retrieved from the surface of the disk using a read/write head for each magnetized surface of the disk. Figure 5.1 shows the physical layout of a magnetic disk drive and the organization of a disk.

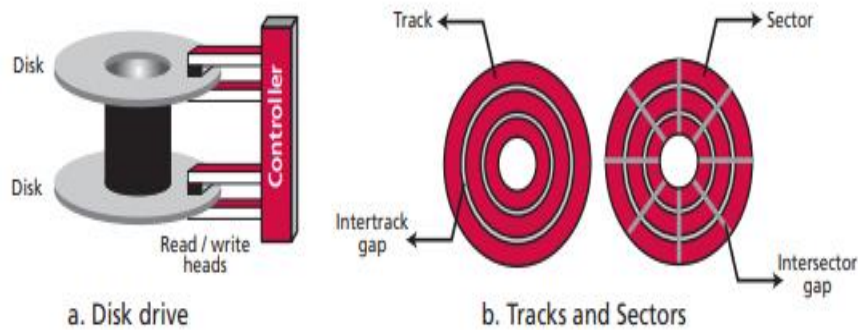


Figure 5.1 Magnetic disks

❑ **Surface organization.** To organize data stored on the disk, each surface is divided into tracks, and each track is divided into sectors (Figure 5.1). The tracks are separated by an **intertrack gap**, and the sectors are separated by an **intersector gap**.

❑ **Data access.** A magnetic disk is considered a random access device. In a random access device, a data item can be accessed randomly without the need to access all other data items located before it.

❑ **Performance.** The performance of a disk depends on several factors, the most important being the rotational speed, the seek time, and the transfer time. The rotational speed defines how fast the disk is spinning. The seek time defines the time to move the read/write head to the desired track where the data is stored. The transfer time defines the time to move data from the disk to the CPU/memory.

Commonly used direct-access secondary storage device. Magnetic Disk – Storage Capacity: -

**Storage capacity of a disk system = Number of recording surfaces**

**\*Number of tracks per surface**

**\* Number of sectors per track**

**\* Number of bytes per sector**

## 2- Magnetic tape

Magnetic tape comes in various sizes. One common type is half-inch plastic tape coated with a thick magnetic film. The tape is mounted on two reels and uses a

read/write head that reads or writes information when the tape is passed through it. Figure 5.2 shows the mechanical configuration of a magnetic tape drive.

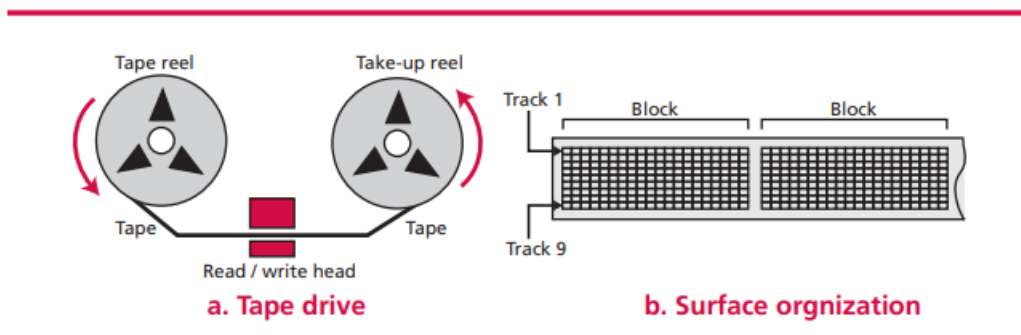


Figure 5.2 Magnetic tape

- ❑ **Surface organization.** The width of the tape is divided into nine tracks, each location on a track storing 1 bit of information. Nine vertical locations can store 8 bits of information related to a byte plus a bit for error detection (Figure 5.2).
- ❑ **Data access.** A magnetic tape is considered a sequential access device. Although the surface may be divided into blocks, there is no addressing mechanism to access each block. To retrieve a specific block on the tape, we need to pass through all the previous blocks.
- ❑ **Performance.** Although magnetic tape is slower than a magnetic disk, it is cheaper. Today, people use magnetic tape to back up large amounts of data.

**Storage capacity is virtually unlimited because as many tapes as required can be used for storing very large data sets**

## Optical storage devices

Optical storage devices, a relatively recent technology, use laser light to store and retrieve data. The use of optical storage technology followed the invention of the compact disk (CD) used to store audio information. Today, the same technology—slightly improved—is used to store information in a computer. Devices that use this technology include CD-ROMs, CD-Rs, CD-RWs, and DVDs.



## Data Storage and Representation

### DATA TYPES

Data today come in different forms including numbers, text, audio, images, and video (Figure 6.1).

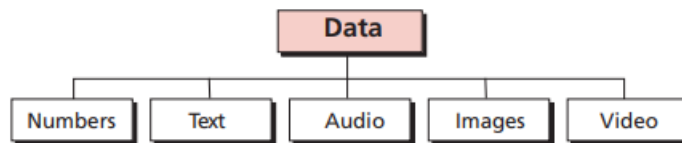


Figure (6.1) Data type

### Data inside the computer

All data types are transformed into a uniform representation when they are stored in a computer and transformed back to their original form when retrieved. This universal representation is called a bit pattern.

### Bits

A bit (binary digit) is the smallest unit of data that can be stored in a computer and has a value of 0 or 1. A bit represents the state of a device that can take one of two states. For example, a switch can be **on** or **off**.

### Bit patterns

To represent different types of data, we use a bit pattern, a sequence, or as it is sometimes called, a string of bits. Figure 6.2 shows a bit pattern made up of sixteen bits. It is a combination of sixteen 0s and 1s. This means that if we need to store a bit pattern made of sixteen bits, we need sixteen electronic switches. If we need to store 1000 bit patterns, each sixteen bits long, we need 16 000 switches, and so on. By tradition a bit pattern with eight bits is called a byte. Sometimes the term word is used to refer to a longer bit pattern.

---

1 0 0 0 1 0 1 0 1 1 1 1 1 1 0 1

---

Figure (6.2) Bit patterns: a piece of data belonging to different data types can be stored as the same pattern in the memory.

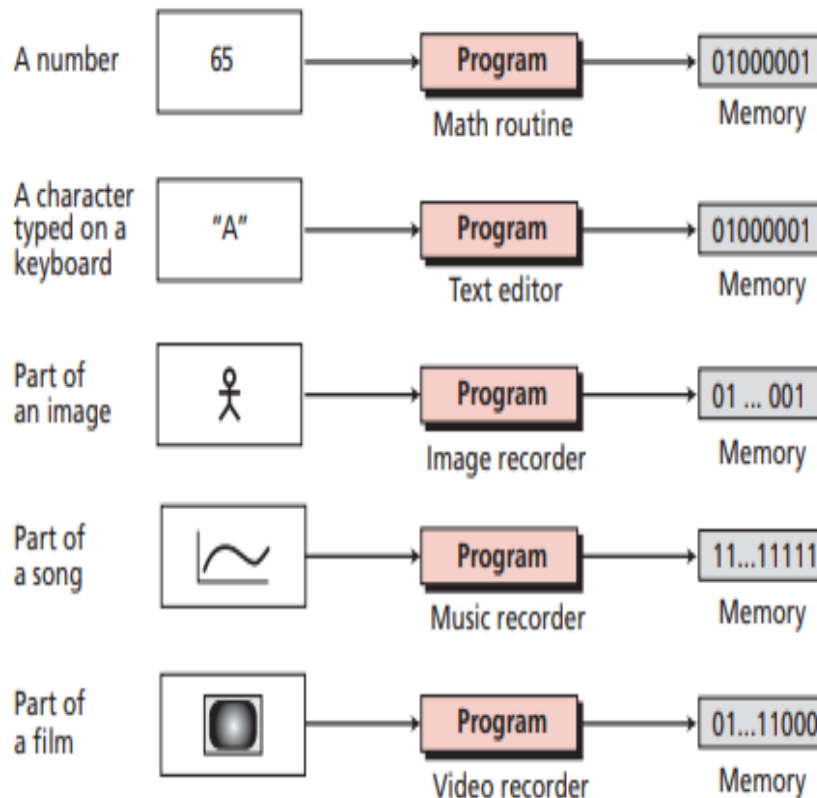


Figure (6.3) Storage of different data types

If we are using a text editor (a word processor), the character A typed on the keyboard can be stored as the 8-bit pattern 01000001. The same 8-bit pattern can represent the number 65 if we are using a mathematical routine. Moreover, the same pattern can represent part of an image, part of a song, or part of a scene in a film. The computer's memory stores all of them without recognizing what type of data they represent figure (6.3).

## 1- STORING NUMBERS

A number is changed to the binary system before being stored in the computer memory, For the decimal point, computers use two different representations: fixed-

point and floatingpoint. The first is used to store a number as an integer—without a fractional part, the second is used to store a number as a real—with a fractional part.

### Storing integers

Integers are whole numbers (numbers without a fractional part). For example, 134 and -125 are integers, whereas 134.23 and -0.235 are not. An integer can be thought of as a number in which the position of the decimal point is fixed: the decimal point is to the right of the least significant (rightmost) bit.

**Note ((An integer is normally stored in memory using fixed-point representation))**

### Unsigned representation

An unsigned integer is an integer that can never be negative and can take only 0 or positive values. Its range is between 0 and positive infinity. However, since no computer can possibly represent all the integers in this range, most computers define a constant called the maximum unsigned integer, which has the value of  $(2^n - 1)$  where  $n$  is the number of bits allocated to represent an unsigned integer.

### Storing unsigned

integers An input device stores an unsigned integer using the following steps:

1. The integer is changed to binary.
2. If the number of bits is less than  $n$ , 0s are added to the left of the binary integer so that there is a total of  $n$  bits. If the number of bits is greater than  $n$ , the integer cannot be stored. A condition referred to as overflow will occur, which we discuss later.

### Example 1

Store 7 in an 8-bit memory location using unsigned representation.

### Solution

First change the integer to binary,  $(111)_2$ . Add five 0s to make a total of eight bits,  $(00000111)_2$ . The integer is stored in the memory location. Note that the subscript 2 is used to emphasize that the integer is binary, but the subscript is not stored in the computer:

Change 7 to binary → 1 1 1  
 Add five bits at the left → 0 0 0 0 0 1 1 1

## Example 2

Store 258 in a 16-bit memory location.

### Solution

First change the integer to binary  $(100000010)_2$ . Add seven 0s to make a total of sixteen bits  $(0000000100000010)_2$ . The integer is stored in the memory location:

Change 258 to binary → 1 0 0 0 0 0 0 1 0  
 Add seven bits at the left → 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0

## Storing reals

A real is a number with an integral part and a fractional part. For example, 23.7 is a real number—the integral part is 23 and the fractional part is  $7/10$ . Although a fixed-point representation can be used to represent a real number, the result may not be accurate or it may not have the required precision. The solution for maintaining accuracy or precision is to use floating-point representation.

## 2- STORING TEXT

A section of text in any language is a sequence of symbols used to represent an idea in that language. For example, the English language uses 26 symbols (A, B, C, ..., Z) to represent uppercase letters, 26 symbols (a, b, c, ..., z) to represent lowercase letters, ten symbols (0, 1, 2, ..., 9) to represent numeric characters (not actual numbers—numbers are treated separately, and symbols (., ?, :, ; , ..., !) to represent punctuation. Other symbols such as blank, newline, and tab are used for text alignment and readability. We can represent each symbol with a bit pattern. In other words, text such as 'CATS', which is made up from four symbols, can be represented as four n-bit patterns, each pattern defining a single symbol (Figure 6.4).

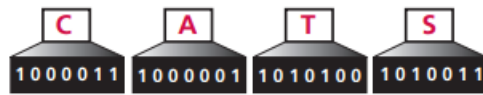


Figure 6.4 Representing symbols using bit patterns

### 3-STORING AUDIO

Audio is a representation of sound or music. Audio, by nature, is different from the numbers or text we have discussed so far. Text is composed of countable entities (characters): we can count the number of characters in text. Text is an example of digital data. In contrast, audio is not countable. Audio is an entity that changes with time—we can only measure the intensity of the sound at each moment. When we discuss storing audio in computer memory, we mean storing the intensity of an audio signal, such as the signal from a microphone, over a period of time: one second, one hour.

### 4-STORING IMAGES

Images are stored in computers using two different techniques: raster graphics and vector graphics

### 4-STORING VIDEO

Video is a representation of images (called frames) over time. A movie consists of a series of frames shown one after another to create the illusion of motion. In other words, video is the representation of information that changes in space (single image) and in time (a series of images). So, if we know how to store an image inside a computer, we also know how to store video: each image or frame is transformed into a set of bit patterns and stored. The combination of the images then represents the video. Today video is normally compressed.

## Number systems: Non-positional number systems and Positional number systems

**Introduction**

A number system (or numeral system) defines how a number can be represented using distinct symbols. A number can be represented differently in different systems. Several number systems have been used in the past and can be categorized into two groups: positional and non-positional systems. Our main goal is to discuss the positional number systems, but we also give examples of non-positional systems.

**1- POSITIONAL NUMBER SYSTEMS**

In a positional number system, the position a symbol occupies in the number determines the value it represents. In this system, a number represented as:

$$\pm (S_{K-1} \dots S_2 S_1 S_0 \cdot S_{-1} S_{-2} \dots S_{-L})_b$$

has the value of:

$$n = \pm S_{K-1} \times b^{K-1} + \dots + S_1 \times b^1 + S_0 \times b^0 \\ + S_{-1} \times b^{-1} + S_{-2} \times b^{-2} + \dots + S_{-L} \times b^{-L}$$

in which  $S$  is the set of symbols,  $b$  is the base (or radix), which is equal to the total number of the symbols in the set  $S$ , and  $S_K$  and  $S_L$  are symbols in the whole and fraction parts of the number.

**1.1 The decimal system (base 10)**

The first positional number system we discuss in this chapter is the decimal system. The word decimal is derived from the Latin root *decem* (ten). In this system the base  $b = 10$  and we use ten symbols to represent a number. The set of symbols is  $S = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ . As we know, the symbols in this system are often referred to as decimal digits or just digits. In the decimal system, a number is written as:

$$\pm (S_{K-1} \dots S_2 S_1 S_0 \cdot S_{-1} S_{-2} \dots S_{-L})_{10}$$

## Integers

An integer (an integral number with no fractional part) in the decimal system is familiar to all of us—we use integers in our daily life. The value is calculated as:

$$N = \pm S_{K-1} \times 10^{K-1} + S_{K-2} \times 10^{K-2} + \dots + S_2 \times 10^2 + S_1 \times 10^1 + S_0 \times 10^0$$

**Example 1//** The following shows the place values for the integer +224 in the decimal system:

		$10^2$		$10^1$		$10^0$	Place values
		2		2		4	Number
$N = +$	$+$	$2 \times 10^2$	$+$	$2 \times 10^1$	$+$	$4 \times 10^0$	Values

## Reals

A real (a number with a fractional part) in the decimal system is also familiar. For example, we use this system to show dollars and cents (\$23.40). The value is calculated as:

	<b>Integral part</b>		<b>Fractional part</b>
$R = \pm$	$S_{K-1} \times 10^{K-1} + \dots + S_1 \times 10^1 + S_0 \times 10^0$	$+$	$S_{-1} \times 10^{-1} + \dots + S_{-L} \times 10^{-L}$

**Example 2//** The following shows the place values for the real number +24.13:

	$10^1$		$10^0$		$10^{-1}$		$10^{-2}$	Place values
	2		4	.	1		3	Number
$R = +$	$2 \times 10$	$+$	$4 \times 1$	$+$	$1 \times 0.1$	$+$	$3 \times 0.01$	Values

### 1.2 The binary system (base 2)

The second positional number system we discuss in this chapter is the binary system. The word binary is derived from the Latin root *bini* (or two by two). In this system the base  $b = 2$  and we use only two symbols,  $S = \{0, 1\}$ . The symbols in this system are often referred to as binary digits or bits (binary digit).

## Integers

We can represent an integer as  $+\_ (S_{K-1} \dots S_1 S_0)_2$ . The value is calculated as:

**Example3//** The following shows that the number  $(11001)_2$  in binary is the same as 25 in decimal. The subscript 2 shows that the base is 2:

$2^4$	$2^3$	$2^2$	$2^1$	$2^0$	Place values
1	1	0	0	1	Number
$1 \times 2^4$	$+ 1 \times 2^3$	$+ 0 \times 2^2$	$+ 0 \times 2^1$	$+ 1 \times 2^0$	Decimal

Note that the equivalent decimal number is  $N = 16 + 8 + 0 + 0 + 1 = 25$ .

## Reals

A real—a number with an optional fractional part—in the binary system can be made of K bits on the left and L bits on the right,  $+(S_{K-1} \dots S_1 S_0 \cdot S_{-1} \dots S_{-L})_2$ . The value can be calculated as:

$$R = \pm \left[ \begin{array}{c} \text{Integral part} \\ S_{K-1} \times 2^{K-1} + \dots + S_1 \times 2^1 + S_0 \times 2^0 \end{array} \right] + \left[ \begin{array}{c} \text{Fractional part} \\ S_{-1} \times 2^{-1} + \dots + S_{-L} \times 2^{-L} \end{array} \right]$$

**Example 4//** The following shows that the number  $(101.11)_2$  in binary is equal to the number 5.75 in decimal:

$2^2$	$2^1$	$2^0$	$2^{-1}$	$2^{-2}$	Place values
1	0	1	1	1	Number
$1 \times 2^2$	$+ 0 \times 2^1$	$+ 1 \times 2^0$	$+ 1 \times 2^{-1}$	$+ 1 \times 2^{-2}$	Values

Note that the value in the decimal system is  $R = 4 + 0 + 1 + 0.5 + 0.25 = 5.75$ .

## 1.3 The hexadecimal system (base 16)

The word hexadecimal is derived from the Greek root hex (six) and the Latin root *decem* (ten). To be consistent with decimal and binary, it should really have been called sexadecimal, from the Latin roots sex and decem. In this system the base b is 16 and we use 16 symbols to represent a number. The set of symbols is  $S = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$ . Note that the symbols A, B, C, D, E, F (uppercase or lowercase) are equivalent to 10, 11, 12, 13, 14, and 15 respectively. The symbols in this system are often referred to as hexadecimal digits.

## Integer



**Example 5//** The following shows that the number  $(2AE)_{16}$  in hexadecimal is equivalent to 686 in decimal:

$N =$	$16^2$	+	$16^1$	+	$16^0$	Place values
	2		A		E	Number
	$2 \times 16^2$	+	$10 \times 16^1$	+	$14 \times 16^0$	Values

Note that the value in the decimal system is  $N = 512 + 160 + 14 = 686$ .

## Real

Although a real number can be also represented in the hexadecimal system, it is not very common.

### 1.4 The octal system (base 8)

The second system that was devised to show the equivalent of the binary system outside the computer is the octal system. The word octal is derived from the Latin root *octo* (eight). In this system the base is 8 and we use eight symbols to represent a number. The set of symbols is  $S = \{0, 1, 2, 3, 4, 5, 6, 7\}$ . The symbols in this system are often referred to as octal digits.

## Integer

**Example 6//** The following shows that the number  $(1256)_8$  in octal is the same as 686 in decimal:

$N =$	$8^3$	+	$8^2$	+	$8^1$	+	$8^0$	Place values
	1		2		5		6	Number
	$1 \times 8^3$	+	$2 \times 8^2$	+	$5 \times 8^1$	+	$6 \times 8^0$	Values

Note that the decimal number is  $N = 512 + 128 + 40 + 6 = 686$ .

## Reals

Although a real number can be also represented in the octal system, it is not very common.

## Summary of the four positional systems

System	Base	Symbols	Examples
Decimal	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	2345.56
Binary	2	0, 1	$(1001.11)_2$
Octal	8	0, 1, 2, 3, 4, 5, 6, 7	$(156.23)_8$
Hexadecimal	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F	$(A2C.A1)_{16}$

## 2- NONPOSITIONAL NUMBER SYSTEMS

Although nonpositional number systems are not used in computers, we give a short review here for comparison with positional number systems. A nonpositional number system still uses a limited number of symbols in which each symbol has a value. However, the position a symbol occupies in the number normally bears no relation to its value—the value of each symbol is fixed. To find the value of a number, we add the value of all symbols present in the representation.

**Example7** //The Roman number system is a good example of a nonpositional number system. This system was invented by the Romans and was used until the sixteenth century in Europe. Roman numerals are still used in sports events, clock dials, and other applications. This number system has a set of symbols  $S = \{I, V, X, L, C, D, M\}$ . The values of each symbol are shown

Symbol	I	V	X	L	C	D	M
Value	1	5	10	50	100	500	1000

Some example:-

III	→	$1 + 1 + 1$	=	3
IV	→	$5 - 1$	=	4
VIII	→	$5 + 1 + 1 + 1$	=	8
XVIII	→	$10 + 5 + 1 + 1 + 1$	=	18
XIX	→	$10 + (10 - 1)$	=	19
LXXII	→	$50 + 10 + 10 + 1 + 1$	=	72
CI	→	$100 + 1$	=	101
MMVII	→	$1000 + 1000 + 5 + 1 + 1$	=	2007
MDC	→	$1000 + 500 + 100$	=	1600

## Computer Codes and Data Structure

### Computer Codes

- ❖ Computer codes are used for internal representation of data in computers
- ❖ As computers use binary numbers for internal data representation, computer codes use binary coding schemes
- ❖ In binary coding, every symbol that appears in the data is represented by a group of bits
- ❖ The group of bits used to represent a symbol is called a byte
- ❖ As most modern coding schemes use 8 bits to represent a symbol, the term byte is often used to mean a group of 8 bits
- ❖ Commonly used computer codes are BCD, EBCDIC, and ASCII

### BCD

- ❖ BCD stands for Binary Coded Decimal
- ❖ It is one of the early computer codes
- ❖ It uses 6 bits to represent a symbol
- ❖ It can represent 64 ( $2^6$ ) different characters

#### **Example**

Using octal notation, show BCD coding for the word DIGIT

#### **Solution:**

D = 64 in BCD octal notation  
 I = 71 in BCD octal notation  
 G = 67 in BCD octal notation  
 I = 71 in BCD octal notation  
 T = 23 in BCD octal notation

Hence, BCD coding for the word DIGIT in octal notation will be

<u>64</u>	<u>71</u>	<u>67</u>	<u>71</u>	<u>23</u>
D	I	G	I	T

### EBCDIC

- ❖ EBCDIC stands for Extended Binary Coded Decimal Interchange Code
- ❖ It uses 8 bits to represent a symbol
- ❖ It can represent 256 ( $2^8$ ) different characters

**Example**

Using binary notation, write EBCDIC coding for the word BIT. How many bytes are required for this representation?

**Solution:**

B = 1100 0010 in EBCDIC binary notation  
 I = 1100 1001 in EBCDIC binary notation  
 T = 1110 0011 in EBCDIC binary notation

Hence, EBCDIC coding for the word BIT in binary notation will be

<u>11000010</u>	<u>11001001</u>	<u>11100011</u>
B	I	T

3 bytes will be required for this representation because each letter requires 1 byte (or 8 bits)

**ASCII**

- ❖ ASCII stands for American Standard Code for Information Interchange.
- ❖ ASCII is of two types – ASCII-7 and ASCII-8
- ❖ ASCII-7 uses 7 bits to represent a symbol and can represent 128 ( $2^7$ ) different characters
- ❖ ASCII-8 uses 8 bits to represent a symbol and can represent 256 ( $2^8$ ) different characters
- ❖ First 128 characters in ASCII-7 and ASCII-8 are same

**Example**

Write binary coding for the word BOY in ASCII-7. How many bytes are required for this representation?

**Solution:**

B = 1000010 in ASCII-7 binary notation  
 O = 1001111 in ASCII-7 binary notation  
 Y = 1011001 in ASCII-7 binary notation

Hence, binary coding for the word BOY in ASCII-7 will be

<u>1000010</u>	<u>1001111</u>	<u>1011001</u>
B	O	Y

Since each character in ASCII-7 requires one byte for its representation and there are 3 characters in the word BOY, 3 bytes will be required for this representation

## *Data Structure*

A data structure uses a collection of related variables that can be accessed individually or as a whole. In other words, a data structure represents a set of data items that share a specific relationship. We discuss three data structures in this chapter: arrays, records, and linked lists. Most programming languages have an implicit implementation of the first two. The third, however, is simulated using pointers and records.

### **1- An array**

An array is a sequenced collection of elements, of the same data type. We can refer to the elements in the array as the first element, the second element, and so forth until we get to the last element. If we were to put our 100 scores into an array, we could designate the elements as scores [1], scores [2], and so on. The index indicates the ordinal number of the element, counting from the beginning of the array. The elements of the array are individually addressed through their subscripts (Figure 9.1). The array as a whole has a name, scores, but each score can be accessed individually using its subscript.

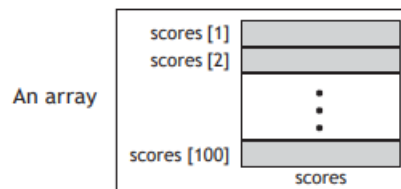


Figure (9.1) An array with index

We can use loops to read and write the elements in an array. We can also use loops to process elements. Now it does not matter if there are 100, 1000, or 10000 elements to be processed—loops make it easy to handle them all. We can use an integer variable to control the loop, and remain in the loop as long as the value of this variable is less than the total number of elements in the array (Figure 9.2).

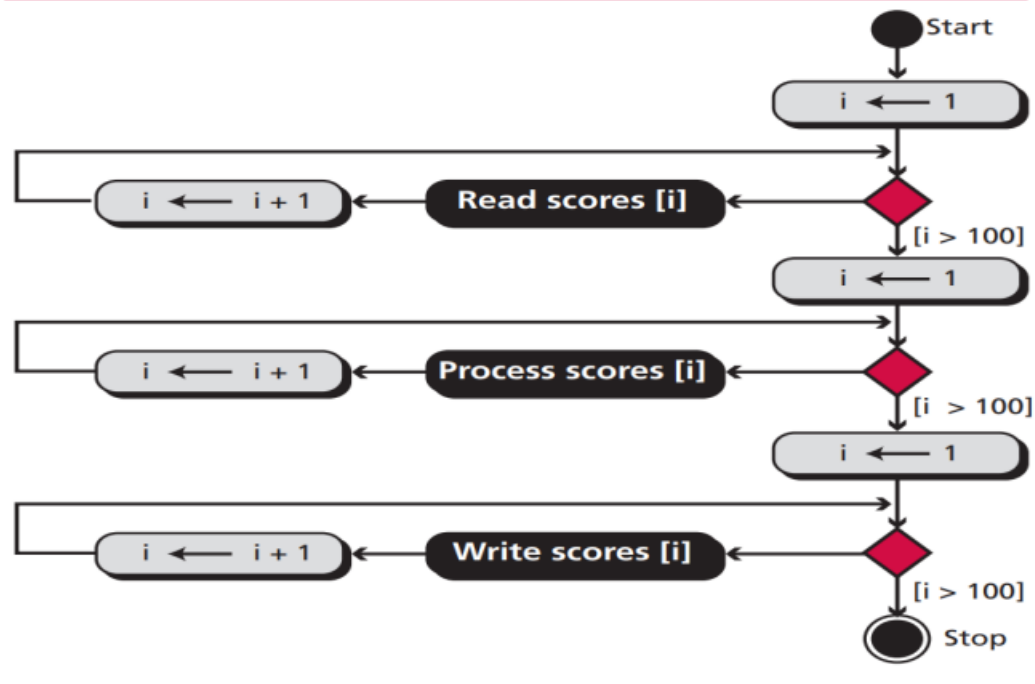


Figure 9.2 Processing an array

**Example 1**

Compare the number of instructions needed to handle 100 individual elements and the array with 100. Assume that processing each score needs only one instruction.

**Solution**

- In the first case, we need 100 instructions to read, 100 instructions to write, and 100 instructions to process. The total is 300 instructions.
- In the second case, we have three loops. In each loop we have two instructions, giving a total of six instructions. However, we also need three instructions for initializing the index and three instructions to check the value of the index. In total, we have 12 instructions.

**2- RECORDS**

A record is a collection of related elements, possibly of different types, having a single name. Each element in a record is called a field. A field is the smallest element of named data that has meaning. A field has a type, and exists in memory. Fields can be assigned values, which in turn can be accessed for selection or manipulation. A field differs from a variable primarily in that it is part of a record.

Figure 9.3 contains two examples of records. The first example, fraction, has two fields, both of which are integers. The second example, student, has three fields made up of three different types.

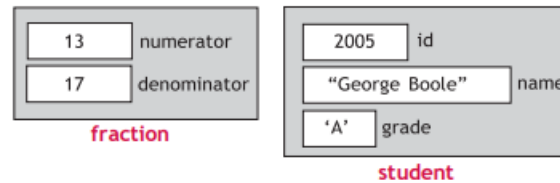


Figure (9.3) A record

## Example 2

The following shows how the value of fields in Figure 9.3 are stored:

**student.id ← 2005 student.name ← "G. Boole" student. grade ← 'A'**

## 3- LINKED LISTS

A linked list is a collection of data in which each element contains the location of the next element—that is, each element contains two parts: data and link. The data part holds the value information: the data to be processed. The link is used to chain the data together, and contains a pointer (an address) that identifies the next element in the list. In addition, a pointer variable identifies the first element in the list. The name of the list is the same as the name of this pointer variable. The link in each element, except the last, points to its successor. The link in the last element contains a null pointer, indicating the end of the list. We define an empty linked list to be only a null pointer. Figure (9.4) also shows an example of an empty linked list.

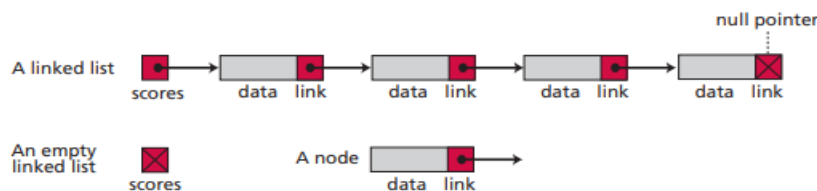


Figure (9.4) empty linked list.



## Algorithms

### Algorithm:

A step-by-step method for solving a problem or doing a task.

In this definition, an algorithm is independent of the computer system. More specifically, we should also note that the algorithm accepts input data and creates output data (figure. 10.1).

---

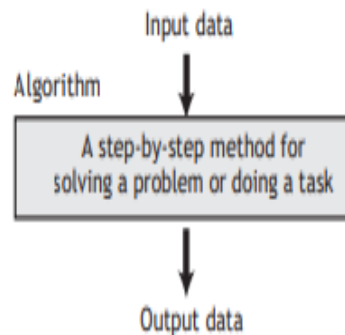


Figure 10.1 Informal definition of an algorithm used in a computer

Let us elaborate on this simple definition with an example. We want to develop an algorithm for finding the largest integer among a list of positive integers. The algorithm should find the largest integer among a list of any size (for example 5, 1000, 10 000, 1000000).

The algorithm should be general and not depend on the number of integers. It is obvious that finding the largest integer among many integers is a task that cannot be done in one step, either by a human or a computer.

The algorithm needs to test each integer one by one.

To solve this problem, we need an intuitive approach.

- ❖ First use a small number of integers (for example, five),
- ❖ then extend the solution to any number of integers.

Our solution for five integers follows the same principles and restrictions for one thousand or one million integers.

- ✓ Assume, even for a five-integer case, that the algorithm handles the integers one by one.
- ✓ It looks at the first integer without knowing the values of the remaining integers.
- ✓ After it handles the first one, it looks at the second integer, and so on.

Figure 10.2 shows one way to solve this problem. We call the algorithm Find Largest. Each algorithm has a name to distinguish it from other algorithms. The algorithm receives a list of five integers as input and gives the largest integer as output.

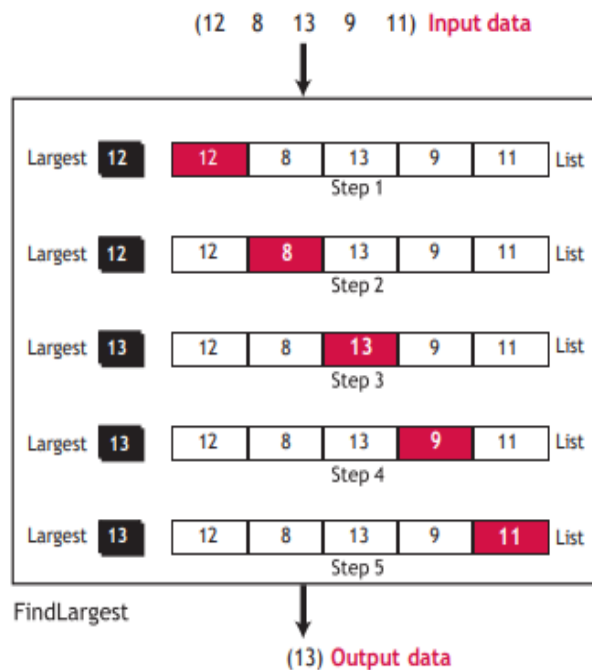


Figure 10.2 Finding the largest integer among five integers

### Input

The algorithm accepts the list of five integers as input.

### Processing

The algorithm uses the following five steps to find the largest integer:

### Step 1

In this step, the algorithm inspects the first integer (12). Since it does not know the values of other integers, it decides that the largest integer (so far) is the first integer. The algorithm defines a data item, called Largest, and sets its value to the first integer (12).

### Step 2

The largest integer so far is 12, but the new integer may change the situation. The algorithm makes a comparison between the value of Largest (12) and the value of the second integer (8). It finds that Largest is larger than the second integer, which means that Largest is still holding the largest integer. There is no need to change the value of Largest.

### Step 3

The largest integer so far is 12, but the new integer (13) is larger than Largest. This means that the value of Largest is no longer valid. The value of Largest should be replaced by the third integer (13). The algorithm changes the value of Largest to 13 and moves to the next step.

### Step 4

Nothing is changed in this step because Largest is larger than the fourth integer (9)

### Step 5

Again nothing is changed because Largest is larger than the fifth integer (11).

## Output

Because there are no more integers to be processed, the algorithm outputs the value of Largest, which is 13.

## ALGORITHM REPRESENTATION

So far, we have used figures to convey the concept of an algorithm. During the last few decades, tools have been designed for this purpose. Two of these tools, UML and pseudocode, are presented here.

**1- FlowChart**

Flowchart is a graphical or symbolic representation of an algorithm. It is the diagrammatic representation of the step-by-step solution to a given problem.

**2- Pseudocode**

Pseudocode is an English-language-like representation of an algorithm.

There is no standard for pseudocode—some people use a lot of detail, others use less. Some use a code that is close to English, while others use a syntax like the Pascal programming language.