## 1 FUNCTIONAL UNIT OF A COMPUTER.

A computer consists of five functionally independent main parts: input, memory, arithmetic and logic, output, and control units, as shown in Figure 1.1. 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. Figure 1.1 does not show the connections among the functional units. The arithmetic and logic circuits, in conjunction with the main control circuits, as the processor, input and output equipment is often collectively referred to as the input-output (I/O) unit.

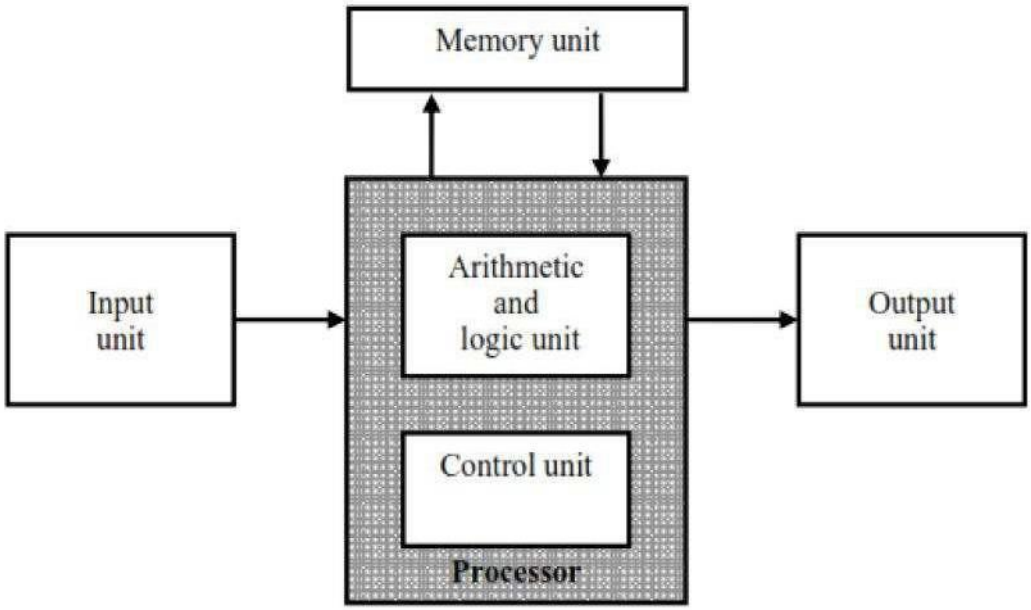


**Figure 1.1 Basic functional unit of a computer.**

### Input Unit

Computers accept coded information through input units, which read data. The 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, which can be used as pointing device. Touch screens 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. Cameras and scanners are used as to get digital images.

**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 storage* is a fast memory that operates at electronic speeds. Programs must be stored in the memory while they are being executed. The memory contains a large number of semiconductor storage cells, each capable of storing one bit of information. These cells are rarely read or written as individual cells but instead are processed in groups of fixed size called *words*. The memory is organized so that the contents of one word, containing n bits, can be stored or retrieved in one basic operation.

To provide easy access to any word in the memory, a distinct address is associated with each word location. Addresses are numbers that identify successive locations. A given word is accessed by specifying its address and issuing a control command that starts the storage or retrieval process. The number of bits in each word is often referred to as the word length of the computer. Typical word lengths range from 16 to 64 bits. The capacity of the memory is one factor that characterizes the size of a computer.

Programs must reside in the memory during execution. Instructions and data can be written into the memory or read out under the controller of the processor. It is essential to be able to access any word location in the memory as quickly as possible. 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). The time required to access one word is called the Memory access time. This time is fixed, independent of the location of the word being accessed. It typically ranges from a few nanoseconds (ns) to about 100 ns for modem RAM units. The memory of a computer is normally implemented as a Memory hierarchy of three or four levels of semiconductor RAM units with different speeds and sizes. The small, fast, RAM units are called caches. They are tightly coupled with the processor and are often contained on the same integrated circuit chip to achieve high performance. The largest and slowest unit is referred to as the main Memory.

Although primary storage is essential, it tends to be expensive. Thus additional, cheaper, *secondary storage* is used when large amounts of data and many programs have to be stored, particularly for information that is access infrequently. A wide selection of secondary storage devices is available, including magnetic disks and tapes and optical disks

**Arithmetic and Logic Unit.**

Most computer operations are executed in the arithmetic and logic unit (ALU) of the processor. Consider a typical example: Suppose two numbers located in the memory are to be added. They are brought into the processor, and the actual addition is carried out by the ALU. The sum may then be stored in the memory or retained in the processor for immediate use.

Any other arithmetic or logic operation, for example, multiplication, division, or comparison of numbers, is initiated by bringing the required operands into the processor, where the operation is performed by the ALU. 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. Access times to registers are somewhat faster than access times to the fastest cache unit in the memory hierarchy.

The control and the arithmetic and logic units are many 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 mechanical controllers.

**Output Unit**

The output unit is the counterpart of the input unit. Its function is to send processed results to the outside world. The most familiar example of such a device is a printer. Printers employ mechanical impact heads, inkjet streams, or photocopying techniques, as in laser printers, to perform the printing. It is possible to produce printers capable of printing as many as 10,000 lines per minute. This is a tremendous speed for a mechanical device but is still very slow compared to the electronic speed of a processor unit.

Monitors, Speakers, Headphones and projectors are also some of the output devices. Some units, such as graphic displays, provide both an output function and an input function. The dual role of input and output of such units are referred with single name as I/O unit in many cases. Speakers, Headphones and projectors are some of the output devices. Storage devices such as hard disk, floppy disk, flash drives are also used for input as well as output.

**Control Unit**

The memory, arithmetic and logic, and input and output '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 task of the control unit. The control unit is effectively the nerve center that sends control signals to other units and senses their states.

I/O transfers, consisting of input and output operations, controlled by the instructions of I/O programs that identify the devices involved and the information to be transferred. However, the actual timing signals that govern the transfers are generated by the control circuits. Timing signals are signals that determine when a given action is to take place. Data transfers between the processor and the memory are also controlled by the control unit through timing signals. It is reasonable to think of a control unit as a well-defined, physically separate unit that interacts with other parts of the machine. In practice, however, this is seldom the case. Much of the control circuitry is physically distributed throughout the machine. A large set of control lines (wires) carries the signals used for timing and synchronization of events in all units.

The operation of a computer can be summarized as follows:

1. The computer accepts information in the form of programs and data through an input unit and stores it in the memory.
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 inside the machine are directed by the control unit.

x----------x---------- x

## 2.BASIC OPERATIONS OF A COMPUTER.

The processor contains arithmetic logic unit (ALU), control circuitry unit (CU) and a number of registers used for several different purposes. The instruction register (IR) holds the instruction that is currently being executed. Its output is available to the control circuits, which generate the timing signals that control the various processing elements involved in executing the instruction.

The program counter (PC) is another specialized register. It keeps track of the execution of a program. "It contains the memory address of the next instruction to be fetched and executed. During the execution of an instruction, the contents of the PC are updated to correspond to the address of the next instruction to be executed. It is customary to say that the PC points to the next instruction that is to be fetched from the memory. Besides the IR and PC, Figure 1.2 shows n General-purpose registers, R0 through R n-1.

Finally, two registers facilitate communication with the memory. These are the memory address register (MAR) and the memory data register (MDR). The MAR holds the address of the location to be accessed. The MDR contains the data to be written into or read out of the addressed location.

Let us now consider some typical operating steps. Programs reside in the memory and usuaJ1y get there through the input unit Execution of the program starts when the PC is set to point to the first instruction of the program. The contents of the PC are transferred to the MAR and a Read control signal is sent to the memory. After the time required to access the memory elapses, the addressed word (in this case, the first instruction ofthe program) is read out of the memory and loaded into the MDR. Next, the contents of the MDR are transferred to the JR. At this point, the instruction is ready to be decoded and executed.

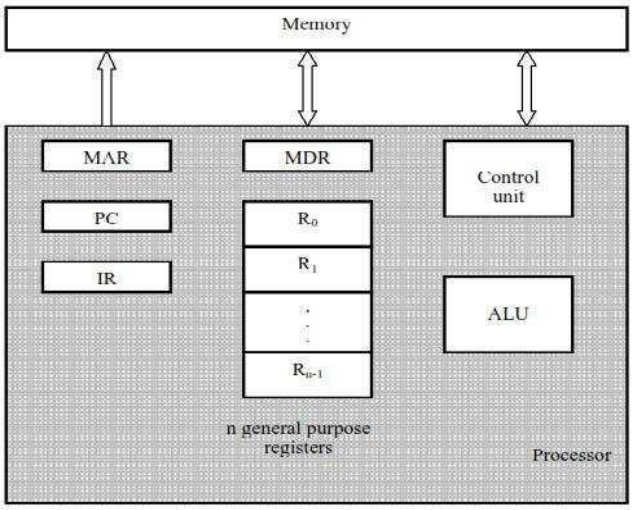


**Figure 1.2 Connections between processor and memory**

If the instruction involves an operation to be performed by the ALU, it is necessary to obtain the required operands. If an operand resides in the memory (it could also be in a general purpose register in the processor), it has to be fetched by sending its address to the MAR and

initiating a Read cycle. When the operand has been read from the memory into the MDR, it is transferred from the MDR to the ALU. After one or more operands are fetched in this way, the ALU can perform the desired operation. If the result of this operation is to be stored in the memory, then the result is sent to the MDR. The address of the location where the result is to be stored is sent to the MAR, and a Write cycle is initiated. At some point during the execution of the current instruction, the contents of the PC are incremented so that the PC points to the next instruction to be executed. Thus, as soon as the execution of the current instruction is completed, a new instruction fetch may be started.

To perform a given task, an appropriate program consisting of a list of instructions is stored in the memory.

## 3.Single bus Organization

In cases where an instruction occupies more than one word, steps 1 and 2 must be repeated as many times as necessary to fetch the complete instruction. These two steps are usually referred to as the fetch phase; step 3 constitutes the execution phase. Figure 2.1 shows an organization in which the arithmetic and logic unit (ALU) and all the registers are interconnected via a single common bus. This bus is internal to the processor and should not be confused with the external bus that connects the processor to the memory and I/O devices. The data and address lines of the external memory bus are shown in Figure 2.1 connected to the internal processor bus via the memory data register, MDR, and the memory address register, MAR, respectively. Register MDR has two inputs and two outputs. Data may be loaded into MDR either from the memory bus or from the internal processor bus. The data stored in MDR may be placed on either bus. The input of MAR is connected to the internal bus, and its output is connected to the external bus. The control lines of the memory bus are connected to the instruction decoder and control logic block. This unit is responsible for issuing the signals that control the operation of all the units inside the processor and for interacting with the memory bus.

The number and use of the processor registers R0 through R(n - 1) vary considerably from one processor to another. Registers may be provided for general-purpose use by the programmer. Some may be dedicated as special-purpose registers, such as index registers or stack pointers. Three registers, Y, Z, and TEMP in Figure 2.1, have not been mentioned before. These registers are transparent to the programmer, that is, the programmer need not be concerned with them because they are never referenced explicitly by any instruction. They are used by the processor for temporary storage during execution of some instructions. These registers are never used for storing data generated by one instruction for later use by another instruction.

The multiplexer MUX selects either the output of register Y or a constant value 4 to be provided as input A of the ALU. The constant 4 is used to increment the contents of the program counter. We will refer to the two possible values of the MUX control input Select as Select4 and SelectY for selecting the constant 4 or register Y, respectively. As instruction execution progresses, data are transferred from one register to another, often passing through the AL U to perform some arithmetic or logic operation. The instruction decoder and control logic unit is responsible for implementing the actions specified by the instruction loaded in the IR

register. The decoder generates the control signals needed to select the registers involved and direct the transfer of data. The registers, the ALU, and the interconnecting bus are collectively referred to as the datapath.

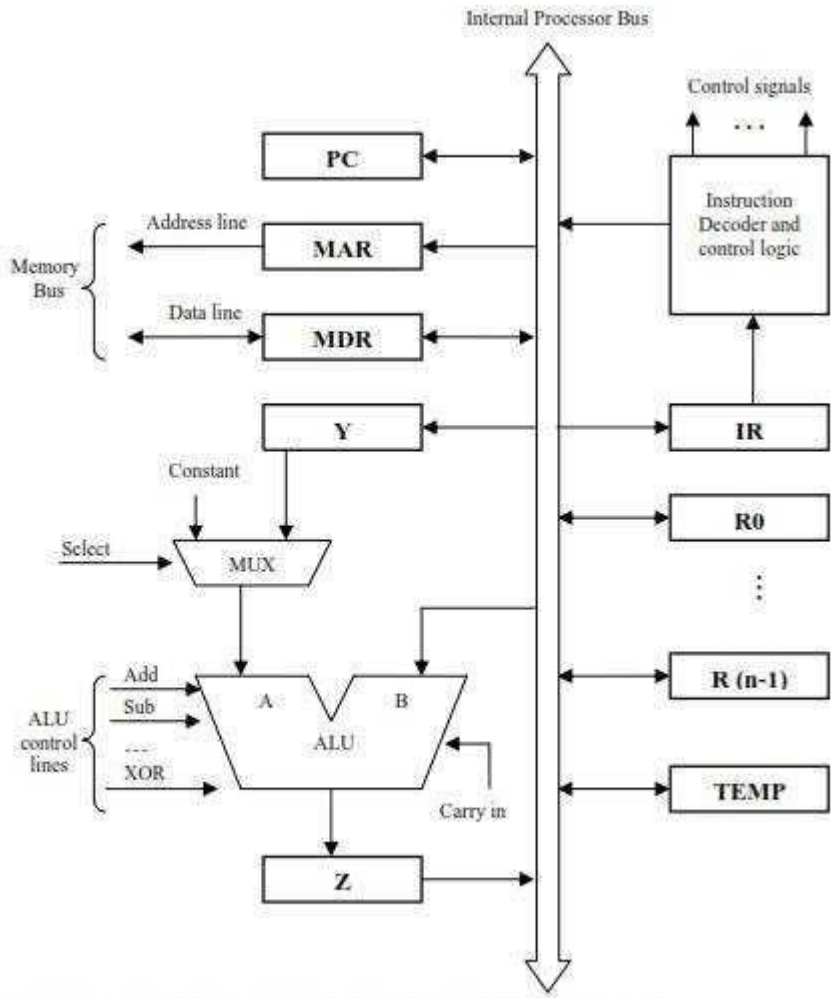


**Figure 2.1** Single bus organization of the data path inside a processor.

With few exceptions, an instruction can be executed by performing one or more of the following operations in some specified sequence:

· Transfer a word of data from one processor register to another or to the ALU
· Perform arithmetic or a logic operation and store the result in a processor register
· Fetch the contents of a given memory location and load them into a processor register
· Store a word of data from a processor register into a given memory location

## 4.ADDRESSING MODES

The different ways in which the location of an operand is specified in an instruction are referred to as addressing modes.

**1. Register addressing mode** - The operand is the contents of a processor register; the name (address) of the register is given in the instruction.

Example: **MOVE R1,R2**
This instruction copies the contents of register R2 to R1.

**2. Absolute addressing mode** - The operand is in a memory location; the address of this location is given explicitly in the instruction. (In some assembly languages, this mode is called Direct.

Example: **MOVE LOC,R2**
This instruction copies the contents of memory location of LOC to register R2.

**3. Immediate addressing mode** - The operand is given explicitly in the instruction. Example: **MOVE #200 , R0**

The above statement places the value 200 in the register R0. A common convention is to use the sharp sign (#) in front of the value to indicate that this value is to be used as an immediate operand.

**4. Indirect addressing mode** - The effective address of the operand is the contents of a register or memory location whose address appears in the instruction.

Example **Add (R2),R0**

Register R2 is used as a pointer to the numbers in the list, and the operands are accessed indirectly through R2. The initialization section of the program loads the counter value n from memory location N into Rl and uses the Immediate addressing mode to place the address value NUM 1, which is the address of the first number in the list, into R2.

**5. Autoincrement mode** - The effective address of the operand is the contents of a register specified in the instruction. After accessing the operand, the contents of this register are automatically incremented to point to the next item in a list. We denote the Autoincrement mode by putting the specified register in parentheses, to show that the contents of the register are used as the effective address, followed by a plus sign to indicate that these contents are to be incremented after the operand is accessed. Thus, the Autoincrement mode is written as **(Ri) +**
As a companion for the Autoincrement mode, another useful mode accesses the items of a list in the reverse order:

**6. Autodecrement mode** - The contents of a register specified in the instruction is first automatically decremented and is then used as the effective address of the operand. We denote the Autodecrement mode by putting the specified register in parentheses, preceded by a minus sign to indicate that the contents of the register are to be decremented before being used as the effective address. Thus, we write **- (Ri)**

**5. Basic Instruction Types**

Instruction types are classified based on the number of operands used in instructions.
They are: (i). Three - Address Instruction,
(ii). Two - Address
Instruction, (iii). One -
Address Instruction,
.
Consider a high level language program command, which adds two variables A and B, and assign the sum in third variable C.

**C = A + B ;**

To carry out this action, the contents of memory locations A and B are fetched from the memory and transferred into the processor where their Sum is computed. This result is then sent back to the memory and stored in location C.

**C [A] + [B]**

Example: Add two variables A and B, and assign the sum in third variable C

**(i). Three - Address Instruction**

The general instruction format is: **Operation Source1,Source2,Destination**
Symbolic add instruction: **ADD A,B,C**

Operands A and B are called the source operands, C is called Destination operand, and Add is the operation to be performed on the operands. If k bits are needed to specify the memory address of each operand, the encoded form of the above instruction must contain 3k bits for addressing purposes in addition to the bits needed to denote the Add operation. For a modem processor with a 32-bit address space, a 3-address instruction is too large to fit in one word for a reasonable word length. Thus, a format that allows multiple words to be used for a single instruction would be needed to represent an instruction of this type. An alternative approach is to use a sequence of simpler instructions to perform the same task, with each instruction having only one or two operands.

**(ii). Two - Address Instruction**

The general instruction format is: **Operation Source, Destination**
Symbolic add instruction: **MOVE B,C ADD A,C**

An Add instruction of this type is **ADD A,B** which performs the operation **B [A] + [B]** When the sum is calculated, the result is sent to the memory and stored in location B, replacing the original contents of this location. This means that operand B is both a source and a destination .A single two-address instruction cannot be used to solve our original problem, which is to add the contents of locations A and B, without destroying either of them, and to place the sum in location C. The problem can be solved by using another two- address instruction that copies the contents of one memory location into another. Such an instruction is **Move B,C** which performs the operation **C [B]**, leaving the contents of location B unchanged. The word "Move" is a misnomer here; it should be "Copy." However, this instruction name is deeply entrenched in computer nomenclature. The operation **C [A] + [B]** can now be performed by the two instruction sequence

Even two-address instructions will not normally fit into one word for usual word lengths and address sizes. Another possibility is to have machine instructions that specify only one memory operand. when a second operand is needed, as in the case of an Add instruction, it is understood implicitly to be in a unique location. A processor register, usually called the accumulator, may be used for this purpose. Thus, the one-address instruction

**(iii). One - Address Instruction**

The general instruction format is: **Operation operand**
Symbolic add instruction: **LOAD A ADD B STORE C**

**ADD A** means the following: Add the contents of memory location A to the contents of the accumulator register and place the sum back into the accumulator. Let us also introduce the oneaddress instructions

**Load A** and **Store A**

The Load instruction copies the contents of memory location A into the Accumulator, and the Store instruction copies the contents of the accumulator into memory location A. Using only one-address instructions, the operation **C [A] + [B]** can be performed by executing the sequence of instructions

**Load A, Add B, Store C**

The operand specified in the instruction may be a source or a destination, depending on the instruction. In the Load instruction, address A specifies the source operand, and the destination location, the accumulator, is

implied. On the other hand, C denotes the destination location in the Store instruction, whereas the source, the accumulator, is implied.