4.2 INTRODUCTION

The computer is composed of four functional units - CPU, Memory, Input and Output and out of these; the CPU behaves like the heart of the computer. Without CPU, the computer is a pen without ink. Hence, since the beginning of the development of computing devices, several researches over the CPU development have been going on, for enhancing the performance of the CPU. Here, in this unit, it has been discussed about the building blocks of the CPU in details.

Obviously, Buses are also playing an important role as a communication medium between the different modules of the computer and so, in this chapter it has been discussed about the Bus and its design elements as well.

A computer system works on basis of the instructions stored in the memory and hence there is a need to know about the layout of the instruction and working modes of the instructions and those are discussed in this unit.

In this unit, the intermediate steps that are needed to execute a computer instruction are discussed thoroughly and as well as about the interrupts and about the situation of the instruction cycle, if interrupt occurs.

Finally, a special focus has given to the Control Unit, a sub division of CPU by looking the hardware to software design issues of the Control Unit. RISC and CISC architecture also have come to focus in this unit.

4.3 CPU BUILDING BLOCKS

CPU stands for Central Processing Unit, which is the principal part of a computer. Once some tasks will be submitted to the computer through the input devices, then the CPU is the responsible for performing the operations over the given tasks and for giving the result to the outside world through the output devices. So, the part of the computer that executes program instructions is known as the Central Processing Unit (CPU) or simply Processor. Now at this point, a question arises that there should have something within the CPU for executing the program instructions as well as something to provide the way to carry out the instructions from the rest of system or simply something for controlling the sequence of the operations. In the next paragraph we will be able to get the answer or to know about the components that makes a CPU.

Basically, the major building blocks of the CPU are-

- Arithmetic and Logic Unit (ALU)
- Control Unit

In addition to the above mentioned building blocks, the Registers within the CPU are also treated as the major components of the CPU. In the **figure-4.1**, basic organization of a computer system is presented by focusing a high level abstract view of the building blocks of the CPU, along with indicating its connection to the rest of the system via the system bus.

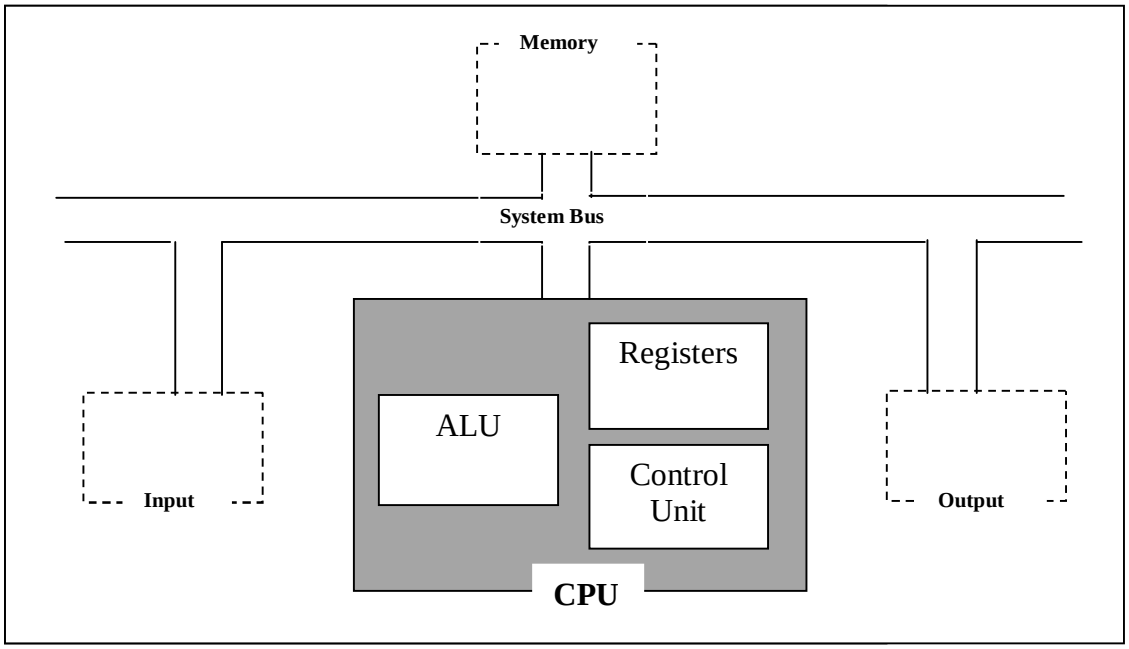


Figure- 4.1: A basic organization of Computer System

4.3.1 ARITHMATIC AND LOGIC UNIT

John Von Neumann proposed the Arithmetic Logic Unit in 1945 when he was working on EDVAC. Arithmetic and Logic unit, simply ALU is a digital circuit for performing the arithmetic and logical operations, such as-

- Addition
- Subtraction
- Logical AND
- Logical OR
- Logical Exclusive OR
- Complement
- Increment
- Decrement
- Left Shift, Left Rotate, etc

To have a clear understanding about ALU, we can consider a situation that one number is located in the memory location and the CPU needs to decrement the number by one. Then the number

needs to bring to the CPU and needs to store in a high speed tiny storage element, called register and then the actual operation will be done by the ALU. Then the modified number may be stored back into the memory or retained in the register for the immediate use. That means, data are presented to the ALU in registers and the results of an operation are stored in registers. Executing one operation by the ALU, ALU may also set the flags as the result of an operation. For example, if in above mentioned decrement operation done by the ALU, the result becomes zero, then ALU will set the ZERO flag.

In the **figure-4.2**, it has been shown that how ALU is interconnected with the rest of the CPU.

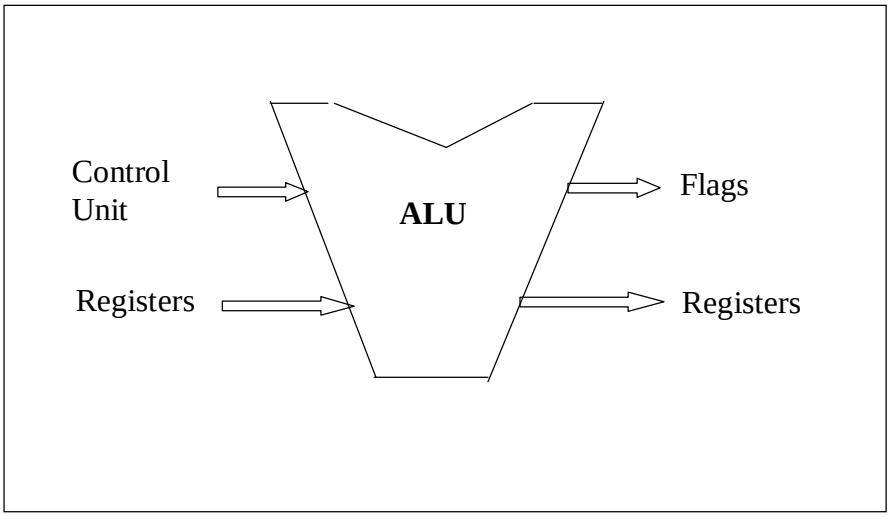


Figure- 4.2: ALU inputs and Outputs

4.3.2 CONTROL UNIT

The control unit, the name itself reflects its functionality that it is a control center for sending control signal to other units to carry out their job and receiving the status of the other unit. The operations that will be performed by the ALU must be coordinated in some way and this is the one of the functionality of the control unit. It generates timing and control signals which are necessary for triggering the operations to be executed in ALU. It directs the movement of electronic signal between memory and ALU. It also directs the control signals between CPU and input/output devices for mutual understanding for communication purpose. After all, control unit can be assumed as the circuitry that controls the flow of information through the processor, coordinates the activities of the other units.

As we know that the execution of program consists of operations and these operations consists of sequence of micro-operations which fall into one of the following categories:

- Data transfer from register to register
- Data transfer from a register to an external interface.
- Data transfer from external interface to register.
- Perform an arithmetic and logical operation, using registers for input and output.

Now, it is easy to summarize that why actually control unit is required. The basic needs of control unit are as follows-

- **Sequencing**: The control unit directs the processor to execute the micro-operations in an ordered way, based on the program being executed.
- **Execution**: The control unit makes execute micro-operations by triggering control signal.

At this point it is clear why control unit is needed, but it is not known to us that how the control unit works by issuing the control signal. Let us have a look at the **figure-4.3**.

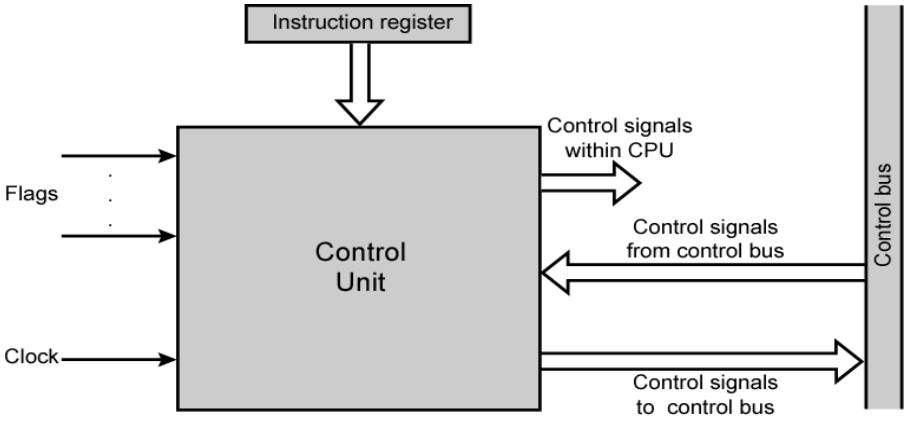


Figure- 4.3: Block diagram of Control Unit

In the above **figure 4.3,** the inputs are Clock, Flags, Instruction Register, Control signals from control bus and the outputs are Control signals within the CPU and Control signals to control bus.

- **Clock:** The Control unit performs one micro-operation to be executed in each clock pulse, which is known as processor cycle time or clock cycle time.
- **Instruction Register:** Instruction register holds the current instruction to be executed. Looking at the operation code of the instruction that is currently available in the instruction register, it is determined that which micro-operations to be executed during execute cycle.

- **Flags:** Flags are set or reset by the ALU operation based on the result of the operation, i.e. status of the flags give the status of the processor and the outcome of the previous ALU operations. Therefore, based on the status of the flags, it is determined what to do in the next.

- **Control signals from control Bus:** This are the signals for mutual understanding that comes from other units and peripherals, such as interrupt signal, acknowledgment signal etc.

- **Control signals within the processor:** These signals cause the data to move from one register to another and for triggering the appropriate ALU operation.

- **Control signals to control bus:** Signals that sent to the Memory or to the other I/O module.

Now, let us consider that how to fetch the instruction from memory. For that Memory Address Register (holds the address of the memory location to be accessed currently) must be filled by the content of the Program Counter (holds the address of the next instruction to be executed) and that is done by activating a control signal from the Control Unit that opens the gates between the bits of the PC and MAR. Next, a control signal will be issued for allowing the content of MAR onto the address bus, and then a memory read control signal will be issued in the control bus to the memory from the control unit. Now, next control signal is issued by the control unit to open the gates to move the content of the data bus in MBR (Memory Buffer Register). Next one control signal will be issued to add 1 to the content of PC and store the result back to the PC again. Finally, the control unit issues a control signal to open the gates between MBR and IR such that the content of MBR can be moved to the IR and hence fetching is over. So, from the above mentioned fetching example, it is noticed that why actually control unit is needed.

4.3.3 REGISTERS

Registers are assumed to be the building blocks of the CPU, as they are providing themselves as a temporary storage at the time of the execution of the instructions.

The registers available in the CPU, can be categorized in two broad sections-

User Visible Registers- These registers make the main memory references to be minimized, as these registers are used as temporary storage for storing data, address and flags status.

Therefore a machine level language programmer can use these registers. Again, the User Visible Register can be sub-categorized into the following-

- **General Purpose Register**- The use or functionality of General Purpose registers, totally depend on the programmer. It is up to the programmer, for what purpose he wants to use this type of register. That means, any General Purpose Register can contain the operand for any opcode.

- **Data Register**- Data Registers are a bit strict that it can hold only data, but cannot be employed in the calculation of an operand address.

- **Address Register**- Address registers are like General Purpose Registers, but they may be devoted to a particular addressing mode. For example,

  Segment pointers (a segment pointer register hold the address of the base of the segment, when a machine supports segmented addressing), **Index register** (Used for indexed addressing), **Stack pointer** (dedicated to hold the address of the top element of the stack in user visible stack addressing)

- **Conditional Codes Register**- This holds the status of the flags affected by the execution of the operation.

Control and Status Registers- These are the special purpose registers used by the control unit to control the operations of the CPU and by the operating system privileged program.

- **Program Counter-** Program Counter is providing the way to control of the flow of execution of the instruction, as it holds the address of the next instruction to be executed or address of the instruction to be fetch.

- **Instruction Register-** This is the register to hold the instruction that is currently being fetched. When the CPU fetch the instruction from the memory, then the fetched instruction will be first stored in MBR and from MBR, the instruction will be copied to the Instruction register. Once the instruction will be available in the Instruction register, the ALU can execute the instruction from the IR.

- **Memory Address Register-** It holds the address of the memory location that is to be accessed. Once CPU try to access one memory location, the content of Program Counter will be copied to the MAR.

- **Memory Buffer Register-** It holds the data that is to be written into the memory or CPU read recently from the memory.

- **Program Status Word-** It holds the status information of the CPU. Status information means the conditional codes along with some other information about interrupt, processor execution mode etc. Basically common fields of PSW are as follows-

  - **Sign Flag:** It is set to 1, if the result of the last arithmetic operation is negative; else it is 0.

  - **Carry Flag:** It is set to 1, if the execution of the last arithmetic operation produces a carry, otherwise it is 0. The carry flag is set or reset in case of addition as well as subtraction. In case of addition, if the operation results carry and in case of subtraction, if the borrow occurs in the operation, then carry flag will be set.

  - **Zero Flag:** It is set to 1, if the result of the last arithmetic or logical operation performed is zero; otherwise set to 0.

  - **Equal Flag:** It is set to 1, if the last logical compare result is equality; else set to 0.

  - **Parity Flag:** It is set to 1, if the arithmetic or logical operation produces a result that contains even number of 1's; else it is set to 0.

  - **Interrupt enable/ disable:** Once there will be interrupt request in the processor and the processor gives the service to the interrupt, then the interrupt enable/ disable bit will be set to 1. After completion of the interrupt service routine, the bit will be set to 0.

  - **Supervisor:** Indicates where the CPU is executing in supervisor mode or user mode. If it is in supervisor mode, the bit is set to 1; else it is set to 0.

CHECK YOUR PROGRESS

1. Fill in the blanks.

(i) CPU stands for_____.

(ii) Major building blocks of CPU are _____, _____and _____ also treated as major building block.

(iii) ALU stands for _____.

(iv) The basic needs of control unit are _____ and _____.

(v) The Control unit performs one micro-operation to be executed in each clock pulse, which is known as _____.

(vi) _____ holds the address of the next instruction to be executed.

(vii)_____register hold the address of the base of the segment, when a machine supports segmented addressing.

4.4 SYSTEM BUS CHARACTERISTICS

As we know that computer is a collection of different modules. The most basic modules of a computer are CPU, Memory and Input/ Output module. Therefore, there should have a way within the computer to communicate with the each other modules or we can say that there must be some paths to connect the modules. The whole collection of the connecting paths between the modules is known as **Interconnection Structure**.

The following types of transfers are necessary to support by an **Interconnection Structure**-

- From Memory to CPU: The CPU should be able to read the instructions or data from the memory.
- From CPU to Memory: The CPU must be able to write into the memory.

- From I/O to CPU: The CPU must be able to read from an I/O device through I/O module.

- From CPU to I/O: The CPU has to be able to send data to the I/O devices, by writing into the I/O module.

- I/O to or from Memory: Without involving the CPU, the Memory and I/O should be able to exchange the information directly in between them.

Now, one question may arise that what is a Bus? Well, Bus is a communication pathway connecting two or more modules. The information transmission in Bus is broadcast in nature and a Bus consists of multiple communication lines.

4.4.1 SYSTEM BUS

The Bus which is connecting the major three components of a computer (CPU, Memory and I/O), is called System Bus. The System Bus generally consists of 50 to 100 separate lines. The System Bus is divided into three categories based on their functionalities as follows-

- **Address Lines:** These lines are used to specify the source/ destination of the data transfer. Address lines width determine the maximum possible memory capacity of the system.

- **Data Lines:** These are the lines to carry the data from one module to another module of the system. Data lines width determine the overall system performance.

- **Control Lines:** The Control lines are used to control the access to and use of data and address lines. Some of the control lines are- Memory Write, Memory Read, I/O Write, I/O Read, Bus request, Bus grant, Interrupt request, Interrupt ACK, Clock, etc.

4.4.2 BUS DESIGN ELEMENTS

Some of the basic design elements that can differentiate buses are as follows-

- **Type:** Bus can be divided into two types- Dedicated and Multiplexed. A Dedicated Bus line is permanently assigned either to one function or to a physical subset of computer components.

But, the address and data information can be transmitted over the same lines by introducing an Address Valid control signal for time multiplexing purpose. This method is called Multiplexed Bus.

- **Method of Arbitration:** The device that is allowed to transfer the data at a given time is called the Bus Master and the process of selecting the next Bus Master is called Arbitration. There are two types of Arbitration- Centralized and Distributed Arbitration.

  In Centralized Arbitration, one single hardware device is responsible for selecting the Bus Masters at a given time. In Distributed Arbitration, there is no central device responsible for arbitration, but each module contains access control logic and modules act together to share the Bus.

- **Timing:** Timing refers to the way in which events are coordinated on the Bus. Two types of Timing exist- Synchronous Timing and Asynchronous Timing.

  In Synchronous Timing, the occurrence of event on the Bus is determined by a clock. Again, in Asynchronous Timing, the occurrence of one event on a Bus, follows and depends on the occurrence of a previous event.

- **Bus Width:** Bus width means the number of lines available in the Bus. So, if the number of data line is more, then more number of bits can be transferred from one module to another one and which leads to the better system performance.
  Again, if the number of address lines is more, then more number of memory locations can be referenced and which leads to better system capacity.

4.5 INSTRUCTION FORMAT

An Instruction is composed of Operation Code (OPCODE) and Operand. The first part of the instruction specifies the task to be performed called OPCODE and the second part of the instruction is data to be operated on, and it is called Operand. Now, operand of the instruction can be in various forms such that 8-bit or 16-bit data, 8-bit or 16-bit address, register or memory address etc. So, instruction format may be different in different instructions (An instruction format is the layout of the bits of an instruction, in terms of its constituent fields). Therefore, generally an instruction is of the following form-

Generally, based on the address field, four common instruction formats are available and they are-

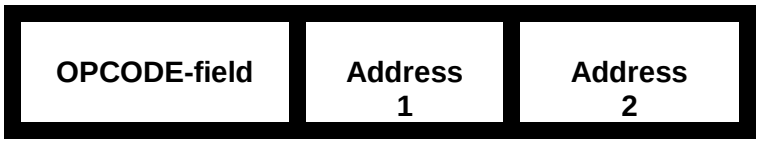
- **Zero-Address Instructions**

For example,
PUSH A i.e., insert the content of A into the stack

- **One-Address Instructions**

For example,
ADD B i.e. Accumulator ← Accumulator + B

- **Two-Address Instructions**



For example,
ADD A, B i.e. A ← A+B

- **Three-Address Instructions**

For example,
ADD A, B, C i.e. A ← B+C

Some of the design points for the Instruction Format-

Instruction Length: Question is that what should be the length of the instruction format? Obviously, it depends on the memory size, memory organization, bus structure, processor complexity and processor speed. For making more richness and flexible, assembly language programmer wants more opcodes, more operands, and more addressing modes, because more opcodes and more operands provide a easy way to write program and more addressing modes give better flexibility for implementing certain functions. But, problem is with the space, because more space is required for more opcodes, operands and for more addressing modes, and hence we say there is a trade off. Again, one more important point is that whether the instruction length should be equal to the memory transfer length or should be multiple of the transfer length.

Allocation of Bits: How to allocate the bits of the instruction for opcode and address field. If more number of bits allocated for the opcode part, then more number of opcodes can have, but reduces the number of bits available for addressing. Hence, here is also trade off in allocation of bits.

Some of the factors for determining number of the bits for the addressing part of the instructions are-

- Number of addressing mode.
- Number of operands.
- Number of register in the register set.
- Address range for referencing memory.

4.6 ADDRESSING MODES

Each instruction of a computer specifies an operation on certain data. There are various ways of specifying address of the data to be operated on. These different ways of specifying data are called the addressing modes. The most common addressing modes are:

- **Immediate addressing mode:** This is the simplest form of addressing. Here, the operand is given in the instruction itself. This mode is used to define a constant or set initial value of variables. The advantage of this mode is that no memory reference other than instruction fetch is required to obtain operand. The disadvantage is that the size of the number is limited to the size of the address field, which is small compared to word length in most instruction sets.

- **Direct addressing mode:** In direct addressing mode, effective address of the operand is given in the

address field of the instruction. It requires one memory reference to read the operand from the given location and provides only a limited address space. Length of the address field is usually less than the word length.

- **Indirect addressing mode:** In Indirect addressing mode, the address field of the instruction refers to the address of a word in memory, which in turn contains the full length address of the operand. The advantage of this mode is that for the word length of N, an address space of 2^N can be addressed. The disadvantage is that instruction execution requires three or more memory references is required to fetch the operand.

- **Register addressing mode:** Register addressing mode is similar to direct addressing. The only difference is that the address field of the instruction refers to a register rather than a memory location. So, only 3 or 4 bits are used as address field to reference 8 to 16 general purpose registers. The advantages of register addressing are small address field is needed in the instruction and no time-consuming memory references are need. The disadvantage of register addressing is that the address space is very limited.

- **Register indirect addressing mode:** This mode is similar to indirect addressing. The address field of the instruction refers to a register. The register contains the effective address of the operand. This mode uses one memory reference to obtain the operand. The address space is limited to the width of the registers available to store the effective address. The advantage of this mode is large address space and disadvantage is one extra memory reference is needed.

- **Displacement addressing mode:** The displacement addressing modes is combination of the direct addressing and register indirect addressing. In displacement addressing, the instruction have two address fields, one will have the memory location address and another will have displacement value to be added to get the effective address. There are 3 types of addressing mode in displacement addressing mode. They are :

  Relative addressing- Here next instruction address is added to the address field to produce the effective address of the operand. Thus the effective address is a displacement relative to the address of the instruction. Here, address field is assumed as 2's complement number.

Base register addressing- Here the referenced register contains a main memory address and address field contains a displacement from that address.

Indexing addressing- Here, the address field references a main memory address, and the reference register contains a positive displacement from that address.

The advantage of this mode is flexibility in addressing and disadvantage is complexity.

- **Stack addressing mode:** Stack is a linear array of locations referred to as last-in first out queue. The stack is a reserved block of location, appended or deleted only at the top of the stack. Stack pointer is a register which stores the address of top of stack location. This mode of addressing is also known as implicit addressing. Here the effective address is the top of the stack, so no memory reference which is the advantage. It's disadvantage is limited applicability.