Chapter – 3

Central Processing Unit

Computer Organization/Structure:

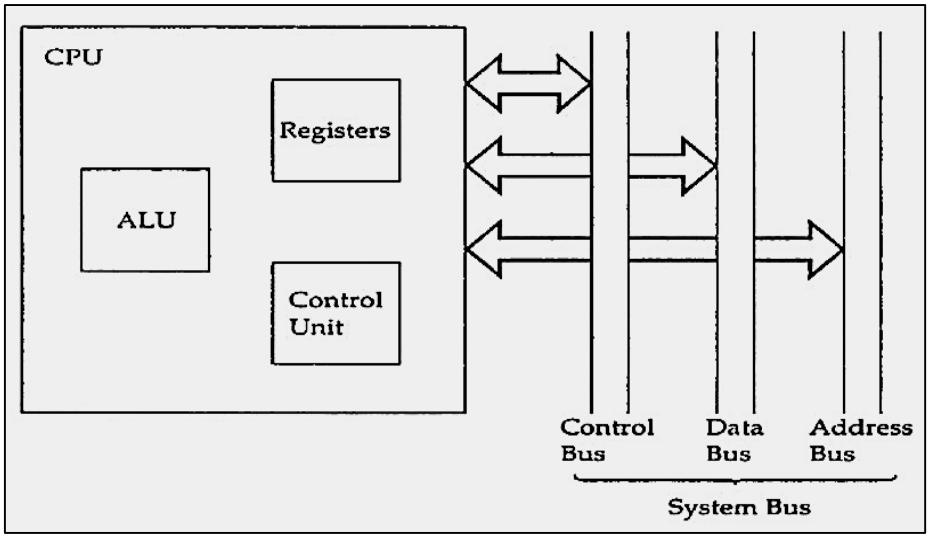

Fig: General View of CPU

CPU consists of three components that is ALU, CU and Register, where ALU performs Arithmetic and Logical operation, CU provides control signals and Registers store binary information which provides fast information to ALU.

These all components are connected with each other and with external environment through system bus as shown in figure above. The operations that must be performed by CPU are as follows:

1. **Fetch Instruction:** The CPU reads an information form memory.
2. **Interpret Instruction:** The instruction is decoded to determine what action is required.
3. **Fetch Data:** The execution of an instruction may require reading data from memory or an input/output module.
4. **Process Data:** The execution of an instruction may require performing some arithmetic or logical operations on data.
5. **Write Data:** The result of an execution may require writing data to memory or an input/output module.

Detailed View of CPU:

The figure shown below is detailed view of CPU. The data transfer and logic control path are indicated including an element labelled as internal CPU BUS. The different parts of ALU are:

1. **Status Flag:** It stores status of any operation like zero, sign, overflow, etc.
2. **Shifter:** Performs left and right shift.
3. **Complementer:** Performs bitwise operation.

4. **Arithmetic and Boolean Logic:** Performs arithmetic and logical operations.
5. **Registers:** Collection of all registers like IR, MBR, MAR, etc.

Similarly, control unit is another important part of CPU organization, which provides control signals to all components like READ, WRITE, HALT, etc.

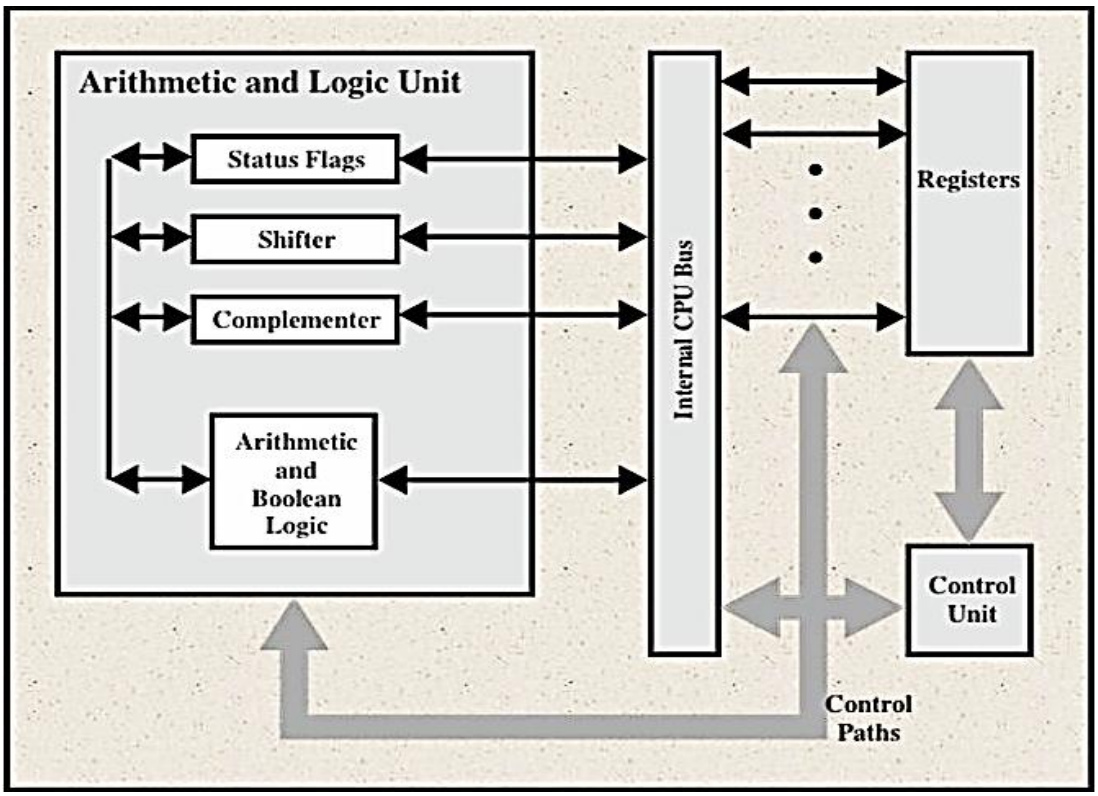

Fig: Detailed View of CPU

REGISTER ORGANIZATION:

Within CPU, there is a set of register that works as an internal memory of CPU. The number of register set in different computer may differ. All these registers can be categorized into two groups which are as follows:

1. USER/PROGRAMMABLE:

These registers can be used by machine or assembly language programmer to minimize the references to main memory. In general there are four types of User visible registers:

a. General Purpose Registers (GpRs):

They are used for various functions desired by processor. It may contain an operand or can be used for calculation of address of operand and can even store some data.

b. Data Register:

They are used only for storing immediate result or data. These data registers are not used for address calculation of an operand.

c. **Address Registers:**

It may be any general purpose register but in some cases few dedicated address registers are used such as:

- **Segment Pointer:** Used to point out segment of memory.
- **Index Register:** Used for index addressing scheme.
- **Stack Pointer:** Used for storing address of top of stack.

d. **Flag Register:**

They are also known as conditional code registers. They are used for set or reset the condition of operation.

2. Status and Control Registers:

For control of various operation several registers are used. These registers cannot be used in data manipulation. Some of the important status and control registers are:

a. **Program Counter:** Contain address of next instruction
b. **Instruction Register:** Contain instruction most recently fetched.
c. **Memory Address Register:** Contain address of instructions and data.
d. **Memory Buffer Register:** Contains a word of data to be written to memory or data word that are most recently used.

DATA PATHS:

A datapath is a collection of functional units (such as arithmetic logic units or multipliers, which perform data processing operations), registers, and buses. Along with the control unit it composes the central processing unit (CPU).

During the late 1990s, there was growing research in the area of reconfigurable datapaths, which may be re-purposed at run-time using programmable fabric, as such designs may allow for more efficient processing as well as substantial power savings.

Functional Blocks of a Datapath:

In computer processors, the datapath often consists of the following functional blocks, or some variation thereof:

- The instruction register stores the current instruction to be executed.
- The program counter (PC) stores the address of the next instruction to be fetched.
- The memory address register (MAR) is a register that either stores the memory address from which data will be fetched to the CPU or the address to which data will be sent and stored.
- The memory data register (MDR) is a register of a computer's control unit that contains the data to be stored in the computer storage (e. g. RAM), or the data after a fetch from the computer storage.

There are also two registers inherent in the processor that facilitate the communication of the processor with the memory, or basically help in the memory operations of the register.

INSTRUCTION CYCLE:

The basic function performed by computer is execution of program which consist of a set of instruction stored in memory. The processing required for a single instruction execution is called instruction cycle. Normally, instruction cycle include following sub-cycles:

a. **Fetch Cycle:** Read next instruction from memory into CPU.
b. **Execution Cycle:** Decode the Opcode and perform indicated operation.
c. **Interrupt Cycle:** If interrupts are enabled or an interrupt is executed, set the current process state and service for interrupt.

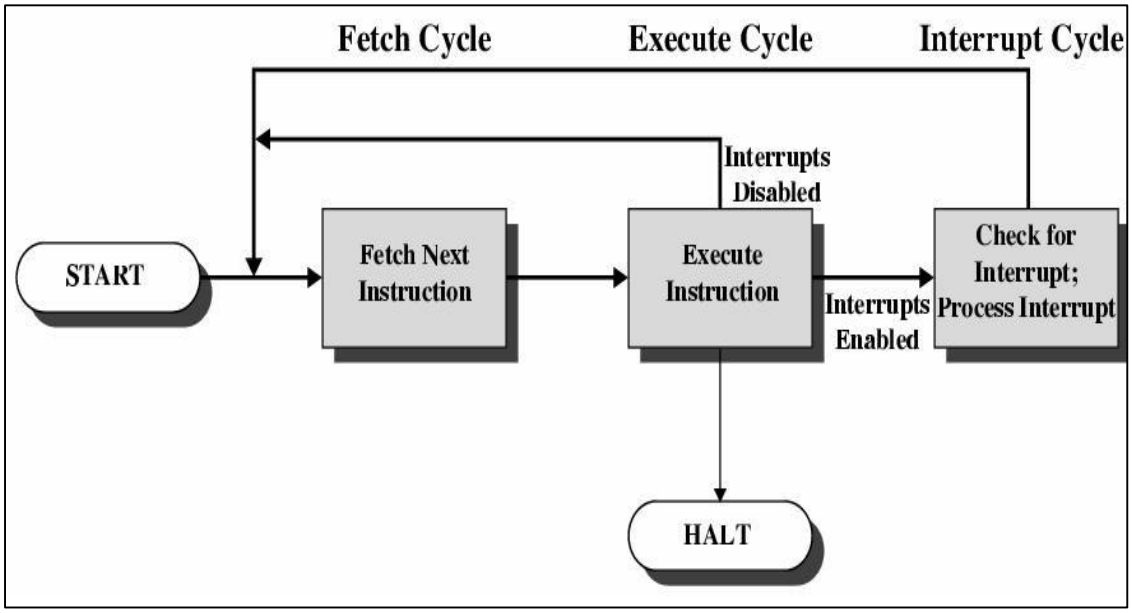

Fig: Instruction Cycle

The execution of an instruction may involve one or more operand in memory, each of which requires a memory access if indirect addressing is used then additional memory access is also required so, fetching of indirect address must be considered as extra cycle indirect execution.

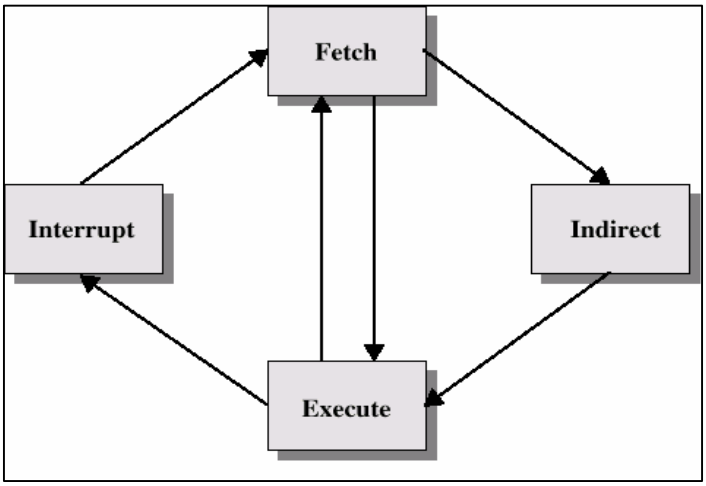

Fig: Instruction Cycle

INSTRUCTION CYCLE STATE DIAGRAM (DETAIL):

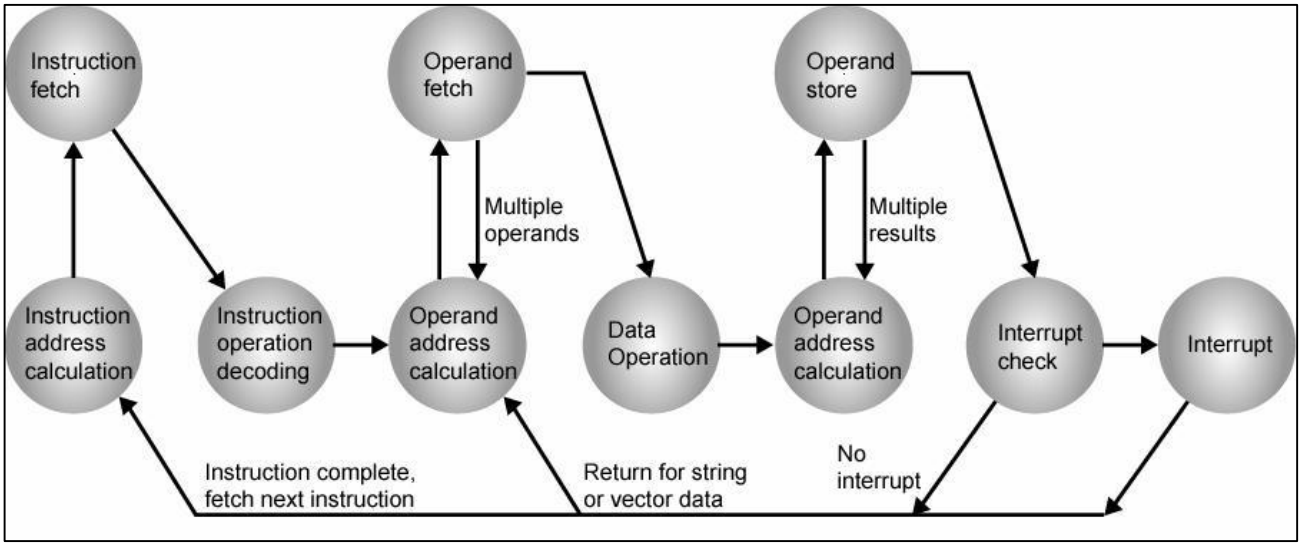

Fig: Instruction Cycle State Diagram

The figure above shown is in the form of state diagram. For any given instruction cycle some states may be NULL and others may be visited more than once. The states can be described as follows:

1. **Instruction Address Calculation:** Determine the address of the next instruction to be executed. Usually this involves adding a fixed number to the address of previous instruction.
2. **Instruction Fetch:** Read instruction from its memory location into the processor.
3. **Instruction Operand Decoding:** Analyze instruction to determine the type of operation to be performed and operations to be used.
4. **Operand Address Calculation:** If the operation involved reference to an operand in memory then determine the address of operand.
5. **Operand Fetch:** Fetch the operand from memory or read it in the form of input/output.
6. **Data Operation:** Perform the operation indicated in the instruction.
7. **Operand Store:** Write the result into memory or out to input/output.
8. **Interrupt Check:** If any type of error occurs in the program then service those interrupts.

ARITHMETIC AND LOGICAL UNIT:

An arithmetic logic unit (ALU) is a digital circuit used to perform arithmetic and logic operations. It represents the fundamental building block of the central processing unit (CPU) of a computer. Modern CPUs contain very powerful and complex ALUs. In addition to ALUs, modern CPUs contain a control unit (CU).

Most of the operations of a CPU are performed by one or more ALUs, which load data from input registers. A register is a small amount of storage available as part of a CPU. The control unit tells the ALU what operation to perform on that data, and the ALU stores the result in an output register. The control unit moves the data between these registers, the ALU, and memory.

An ALU performs basic arithmetic and logic operations. Examples of arithmetic operations are addition, subtraction, multiplication, and division. Examples of logic operations are comparisons of values such as NOT, AND, and OR.

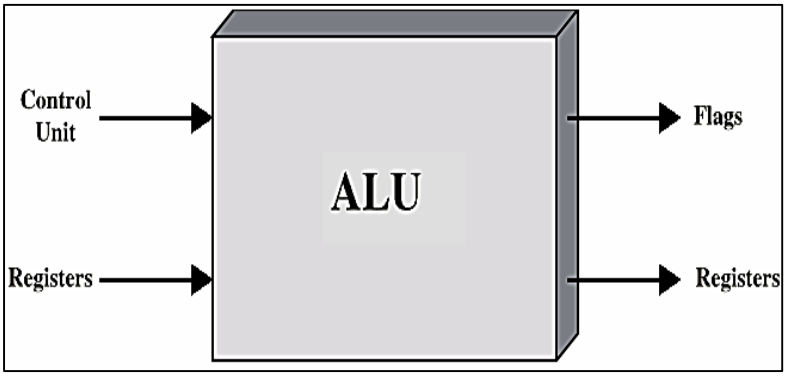

Fig: Block Diagram of ALU

DESIGN PRINCIPLES FOR MODERN SYSTEM:

There is a set of design principles, sometimes called the RISC design principles that architects of general-purpose CPUs do their best to follow:

1. **All Instructions Are Directly Executed by Hardware:**
   - Eliminates a level of interpretation

2. **Maximize the Rate at Which Instructions are Issued:**
   - MIPS = millions of instructions per second
   - MIPS speed related to the number of instructions issued per second
   - Parallelism can play a role

3. **Instructions Should be Easy to Decode:**
   - A critical limit on the rate of issue of instructions
   - Make instructions regular, fixed length, with a small number of fields.
   - The fewer different formats for instructions the better.

4. **Only Loads and Stores Should Reference Memory:**
   - Operands for most instructions should come from- and return to- registers.
   - Access to memory can take a long time
   - Thus, only LOAD and STORE instructions should reference memory.

5. **Provide Plenty of Registers:**
   - Accessing memory is relatively slow, many registers (at least 32) need to be provided, so that once a word is fetched, it can be kept in a register until it is no longer needed.