Introduction to CPU

The operations or tasks that must perform by CPU are:

- **Fetch Instruction:** The CPU reads an instruction from memory.
- **Interpret Instruction:** The instruction is decoded to determine what action is required.
- **Fetch Data:** The execution of an instruction may require reading data from memory or I/O module.
- **Process data:** The execution of an instruction may require performing some arithmetic or logical operation on data.
- **Write data:** The result of an execution may require writing data to memory or an I/O module.

To do these tasks, it should be clear that the CPU needs to store some data temporarily. It must remember the location of the last instruction so that it can know where to get the next instruction. It needs to store instructions and data temporarily while an instruction is being executed. In other words, the CPU needs a small internal memory. These storage locations are generally referred as registers.

The major components of the CPU are an arithmetic and logic unit (ALU) and a control unit (CU). The ALU does the actual computation or processing of data. The CU controls the movement of data and instruction into and out of the CPU and controls the operation of the ALU.

The CPU is connected to the rest of the system through system bus. Through system bus, data or information gets transferred between the CPU and the other component of the system. The system bus may have three components:

- **Data Bus:** Data bus is used to transfer the data between main memory and CPU.
- **Address Bus:** Address bus is used to access a particular memory location by putting the address of the memory location.
- **Control Bus:** Control bus is used to provide the different control signal generated by CPU to different part of the system. As for example, memory read is a signal generated by CPU to indicate that a memory read operation has to be performed. Through control bus this signal is transferred to memory module to indicate the required operation.

There are three basic components of CPU: register bank, ALU and Control Unit. There are several data movements between these units and for that an internal CPU bus is used. Internal CPU bus is needed to transfer data between the various registers and the ALU. The internal organization of CPU in more abstract level is shown in the Figure 5.1 and Figure 5.2.

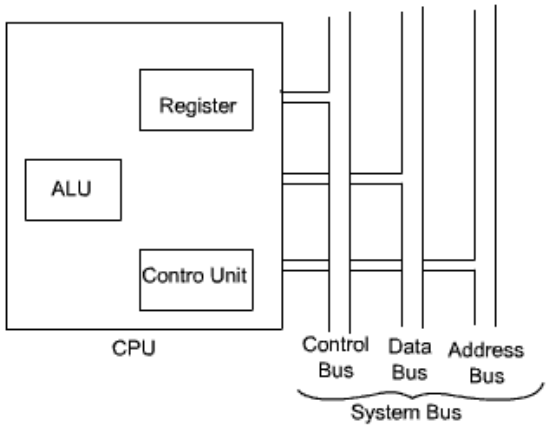

Figure 5.1: CPU with the system Bus

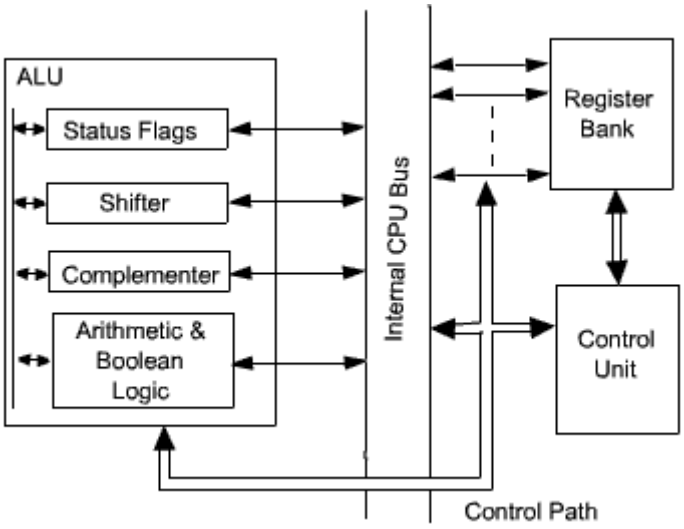

Figure 5.2: Internal Structure of the CPU

Register Organization:

A computer system employs a memory hierarchy. At the highest level of hierarchy, memory is faster, smaller and more expensive. Within the CPU, there is a set of registers which can be treated as a memory in the highest level of hierarchy. The registers in the CPU can be categorized into two groups:

- **User-visible registers:** These enable the machine - or assembly-language programmer to minimize main memory reference by optimizing use of registers.
- **Control and status registers:** These are used by the control unit to control the operation of the CPU. Operating system programs may also use these in privileged mode to control the execution of program.

User-visible Registers: The user-visible registers can be categorized as follows:

- General Purpose Registers
- Data Registers
- Address Registers
- Condition Codes

General-purpose registers can be assigned to a variety of functions by the programmer. In some cases, general- purpose registers can be used for addressing functions (e.g., register indirect, displacement). In other cases, there is a partial or clean separation between data registers and address registers.

Data registers may be used to hold only data and cannot be employed in the calculation of an operand address.

Address registers may be somewhat general purpose, or they may be devoted to a particular addressing mode. Examples include the following:

- *Segment pointer:* In a machine with segment addressing, a segment register holds the address of the base of the segment. There may be multiple registers, one for the code segment and one for the data segment.
- *Index registers:* These are used for indexed addressing and may be auto indexed.

- *Stack pointer:* If there is user visible stack addressing, then typically the stack is in memory and there is a dedicated register that points to the top of the stack.

Condition Codes (also referred to as flags) are bits set by the CPU hardware as the result of the operations. For example, an arithmetic operation may produce a positive, negative, zero or overflow result. In addition to the result itself being stored in a register or memory, a condition code is also set. The code may be subsequently be tested as part of a condition branch operation. Condition code bits are collected into one or more registers.

There are a variety of CPU registers that are employed to control the operation of the CPU. Most of these, on most machines, are not visible to the user. Different machines will have different register organizations and use different terminology. We will discuss here the most commonly used registers which are part of most of the machines.

Four registers are essential to instruction execution:

- *Program Counter (PC):* Contains the address of an instruction to be fetched. Typically, the PC is updated by the CPU after each instruction fetched so that it always points to the next instruction to be executed. A branch or skip instruction will also modify the contents of the PC.
- *Instruction Register (IR):* Contains the instruction most recently fetched. The fetched instruction is loaded into an IR, where the opcode and operand specifiers are analyzed.
- *Memory Address Register (MAR):* Contains the address of a location of main memory from where information has to be fetched or information has to be stored. Contents of MAR are directly connected to the address bus.
- *Memory Buffer Register (MBR):* Contains a word of data to be written to memory or the word most recently read. Contents of MBR are directly connected to the data bus. It is also known as Memory Data Register (MDR).

Apart from these specific register, we may have some temporary registers which are not visible to the user. As such, there may be temporary buffering registers at the boundary to the ALU; these registers serve as input and output registers for the ALU and exchange data with the MBR and user visible registers.

Processor Status Word

All CPU designs include a register or set of registers, often known as the processor status word (PSW) that contains status information. The PSW typically contains condition codes plus other status information. Common fields or flags include the following:

- *Sign:* Contains the sign bit of the result of the last arithmetic operation.
- *Zero:* Set when the result is zero.
- *Carry:* Set if an operation resulted in a carry (addition) into or borrow (subtraction) out of a high order bit.
- *Equal:* Set if a logical compare result is equal.
- *Overflow:* Used to indicate arithmetic overflow.
- *Interrupt enable/disable:* Used to enable or disable interrupts.
- *Supervisor:* Indicate whether the CPU is executing in supervisor or user mode. Certain privileged instructions can be executed only in supervisor mode, and certain areas of memory can be accessed only in supervisor mode.

Apart from these, a number of other registers related to status and control might be found in a particular CPU design. In addition to the PSW, there may be a pointer to a block of memory containing additional status information (e.g. process control blocks).

Concept of Program Execution

The instructions constituting a program to be executed by a computer are loaded in sequential locations in its main memory. To execute this program, the CPU fetches one instruction at a time and performs the functions specified. Instructions are fetched from successive memory locations until the execution of a branch or a jump instruction.

The CPU keeps track of the address of the memory location where the next instruction is located through the use of a dedicated CPU register, referred to as the program counter (PC). After fetching an instruction, the contents of the PC are updated to point at the next instruction in sequence.

For simplicity, let us assume that each instruction occupies one memory word. Therefore, execution of one instruction requires the following three steps to be performed by the CPU:

1. Fetch the contents of the memory location pointed at by the PC. The contents of this location are interpreted as an instruction to be executed. Hence, they are stored in the instruction register (IR). Symbolically this can be written as:
   IR = [[PC]]
2. Increment the contents of the PC by 1.
   PC = [PC] + 1
3. Carry out the actions specified by the instruction stored in the IR.

The first two steps are usually referred to as the fetch phase and the step 3 is known as the execution phase. Fetch cycle basically involves read the next instruction from the memory into the CPU and along with that update the contents of the program counter. In the execution phase, it interprets the opcode and performs the indicated operation. The instruction fetch and execution phase together known as instruction cycle. The basic instruction cycle is shown in the Figure 5.3.

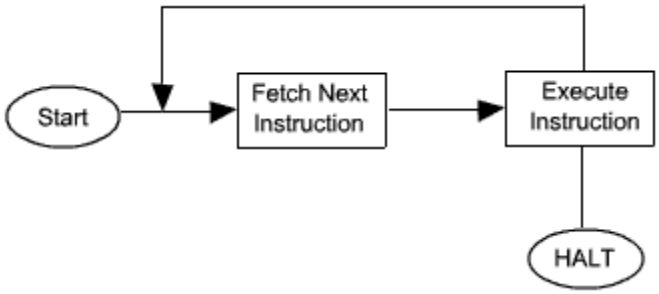

Figure 5.3: Basic Instruction cycle

In cases, where an instruction occupies more than one word, step 1 and step 2 can be repeated as many times as necessary to fetch the complete instruction. In these cases, the execution of a instruction may involve one or more operands in memory, each of which requires a memory access. Further, if indirect addressing is used, then additional memory access is required.

The fetched instruction is loaded into the instruction register. The instruction contains bits that specify the action to be performed by the processor. The processor interprets the instruction and performs the required action. In general, the actions fall into four categories:

- *Processor-memory:* Data may be transferred from processor to memory or from memory to processor.
- *Processor-I/O:* Data may be transferred to or from a peripheral device by transferring between the processor and an I/O module.
- *Data processing:* The processor may perform some arithmetic or logic operation on data.
- *Control:* An instruction may specify that the sequence of execution be altered.

The main line of activity consists of alternating instruction fetch and instruction execution activities. After an instruction is fetched, it is examined to determine if any indirect addressing is involved. If so, the required operands are fetched using indirect addressing.

The execution cycle of a particular instruction may involve more than one reference to memory. Also, instead of memory references, an instruction may specify an I/O operation. With these additional considerations the basic instruction cycle can be expanded with more details view in the Figure 5.4. The figure is in the form of a state diagram.

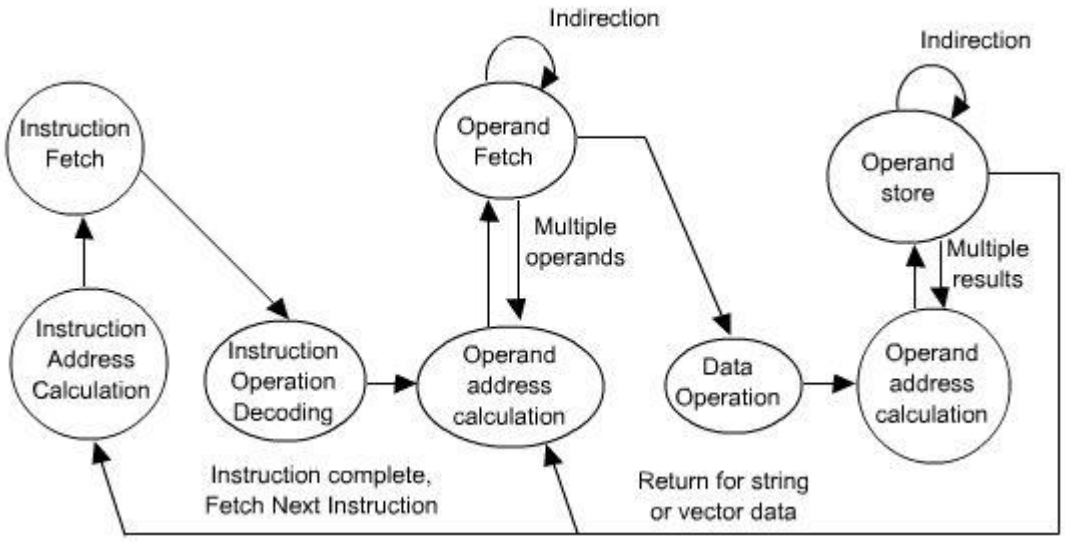

Figure 5.4: Instruction cycle state diagram.

Interrupts

Virtually all computers provide a mechanism by which other module (I/O, memory etc.) may interrupt the normal processing of the processor. The most common classes of interrupts are:

Program: Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, and reference outside the user's allowed memory space.

Timer: Generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis.

I/O: Generated by an I/O controller, to signal normal completion of an operation or to signal a variety of error conditions.

Hardware failure: Generated by a failure such as power failure or memory parity error.

The issue of interrupts is discussed in later module, but we need to introduce the concept of interrupt now to understand more clearly the nature of instruction cycle.

Interrupts are provided primarily as a way to improve processing efficiency. For example, most external devices are much slower than the processor. With interrupts, the processor can be engaged in executing other instructions while an I/O operation is in progress.

For I/O operation, say an output operation, like printing some information by a printer. Printer is much slower device than the CPU. The CPU puts some information on the output buffer. While printer is busy printing these information from output buffer, CPU is lying idle. During this time CPU can perform some other task which does not involve the memory bus.

When the external device becomes ready to be serviced, that is, when it is ready to accept more data from the processor, the I/O module for that external device sends an interrupt request signal to the processor. The processor responds by suspending operation of the current program, branching off to a program to service the particular I/O device (known as an interrupt handler), and resuming the original execution after the device is serviced.

From the point of view of the user program, an interrupt is just that : an interruption of the normal sequence of execution. When the interrupt processing is completed, execution resumes.

To accommodate interrupts, an interrupt cycle is added to the instruction cycle, which is shown in the Figure 5.5. In the interrupt cycle, the processor checks if any interrupt have occurred, indicated by the presence of an interrupt signals. If no interrupts are pending, the processor proceeds to the fetch cycle and fetches the next instruction of the current program. If an interrupt is pending, the processor does the following:

1. It suspends the execution of the current program being executed and saves its contents. This means saving the address of the next instruction to be executed (current contents of the program counter) and any other data relevant to the processor's current activity.
2. It sets the program counter to the starting address of an interrupt handler routine.

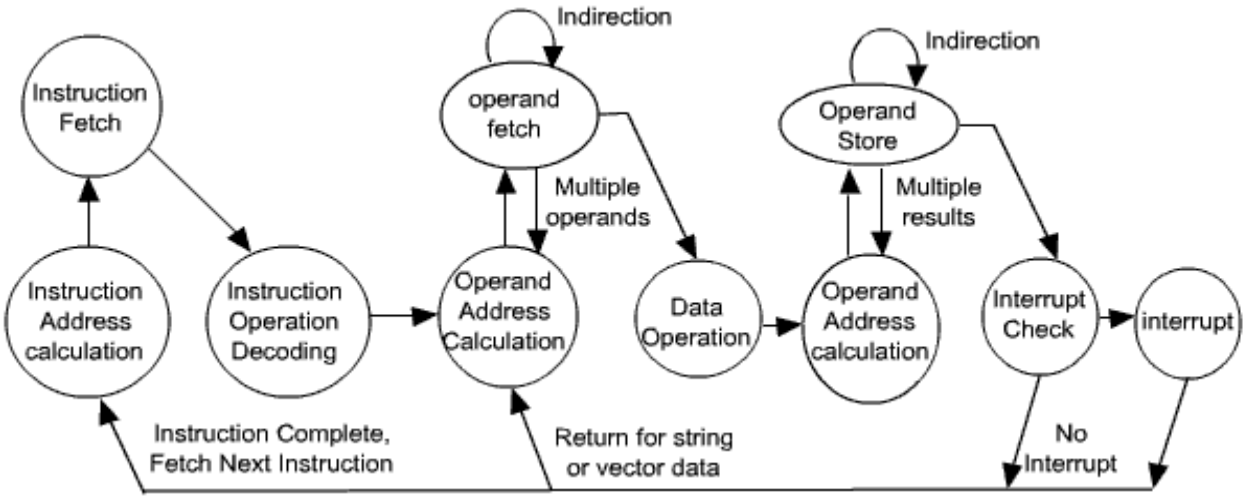

Figure 5.5: Instruction cycle state diagram with interrupt.

The Figure 5.5 shows a revised instruction cycle state diagram that includes interrupt cycle processing.