CHAPTER 5 Computer Architecture

Overview

In 1951, Van Neumann and his team proposed a design of a **stored program computer.** According to his design a sequence of instructions (called a Program) and the data are stored in the memory of the machine. The machine reads the instructions one by one and executes these instructions accordingly. This seemingly simple design is proved to be very powerful and general purpose. It is the basis of most modern day computers.

If we consider the architecture of the modern stored program machine the following are most important components

Control Unit (CU): The control unit reads the instructions from the memory and decodes these instructions. This unit uses other components of the computer to execute the instructions given to the computer

Arithmetic and Logical Unit (ALU): As the general-purpose computer can perform different arithmetic operations on the data so it has a special unit that has electronic circuits to perform the basic arithmetic and logical operations on the data. This is called the **A**rithmetic and **L**ogical **U**nit or **ALU**.

Main Memory: The stored program computer has another very important component that is use to store program and data while these are being executed. This unit is commonly known as the Main Memory of the computer. Sometimes we also call it the working area of the computer

I/O Unit: This handles the processor's communication with its peripherals. For example, Disc drive, monitor, printer etc. There are registers to hold the data coming in or going out and a peripheral device selection unit which determines which interface to send the data to. As the data and instructions should be in the memory before the computer can start executing it so to place data and instructions in the memory this stored program computer also has some I/O devices.

Bus Interconnection: This is another important component of the basic architecture and this component is used to connect different parts of the computer together.

The figure given below shows clearly the main components in the architecture of the computer.

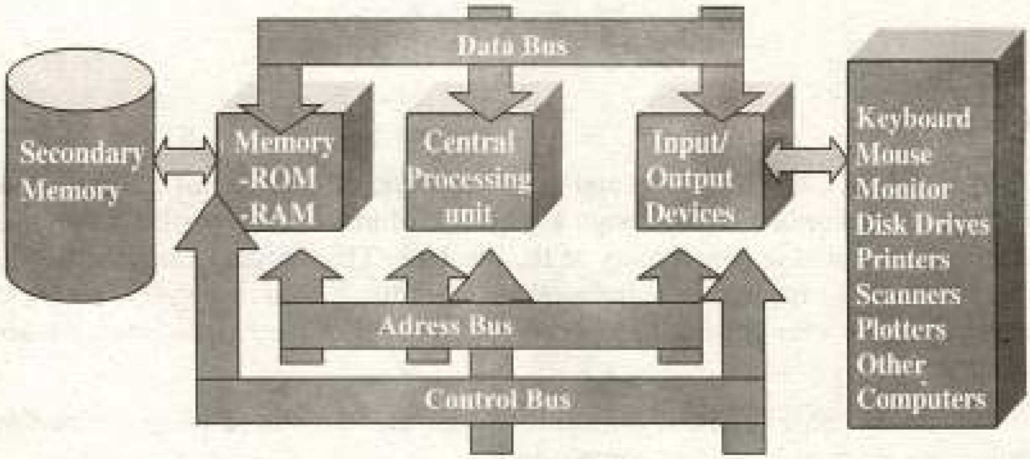
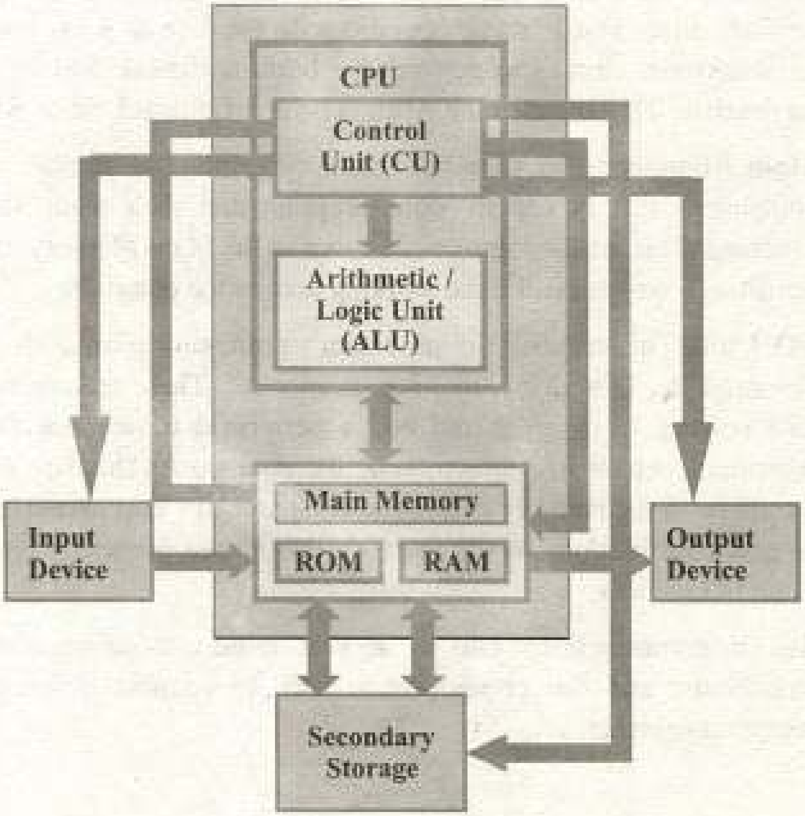


Figure 5.1: Architecture of main Components of Computer

The CPU

The CPU is the brains of the computer. In terms of computing power, the CPU is the most important element of a computer system.



The CPU is centrally located on the motherboard. Since the CPU carries out a large share of the work in the computer, data pass continually through it. The data come from the RAM and the units (keyboard, drives etc.). After processing, the data is send back to RAM and the units.

The CPU continually receives *instructions* to be executed. Each instruction is a data processing order. The work itself consists mostly of *calculations* and *data transport:*

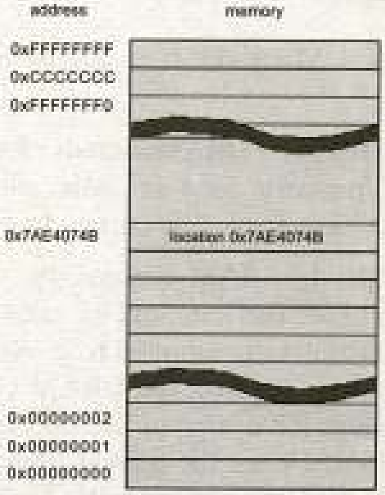
On large machines, CPUs require one or more printed circuit boards. On personal computers and small workstations, the CPU is housed in a single chip called a microprocessor. Two typical components of a CPU are:

The arithmetic logic unit *(ALU)*: The ALU part of a computer that performs all arithmetic computations, such as addition and multiplication, and all comparison operations. The ALU is one component of the CPU.

The ***control unit(CU)***, which extracts instruction from memory and decodes and executes them, calling on the ALU when necessary.

Main Memory: As mentioned earlier, a computer executes a program in its main memory, which is another very important component of the **stored program computer.** A computer cannot work without having some kind of main memory in it. In these section, we will learn more about different types of memories used in a computer and their working.

Mostly the modern computer memory is built in the form of a chip of a semi conductor material. It is built in the form of thousand or even millions of cells each capable of storing a bit i-e a 0 or 1. This is shown in figure 5.3. below.



These cells are logically organized into group of 8 bits called a byte. Each byte in the memory has a unique number assigned it is called the address of that byte. This scheme of arranging cells into a byte and bytes into memory chip is shown in the figure 5.4 below.

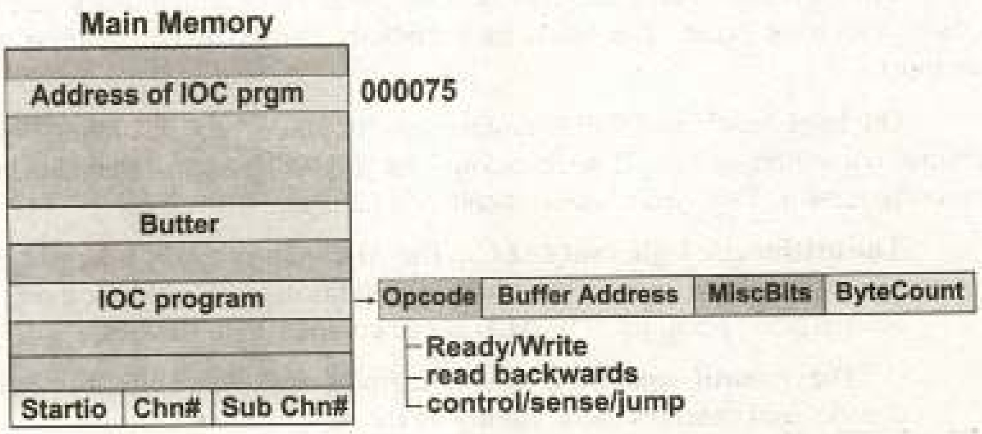


Figure 5.4: Main Memory

From this figure, it is obvious that a memory is sequence of bytes. Also the CPU or any other component of the computer can access any byte from the main memory by specifying its address. Different bytes of the main memory can be accessed directly at random the memory is build from electronic components so accessing any part of the memory takes equal amount of time. So the main memory is direct access storage device. As no mechanical movement is involved in accessing any byte of the memory so the main memory of the computer is very fast as compared to other storage devices like the magnetic and optical disks. There are two types of main memory.

RAM (**R**andom **A**ccess **M**emory): It is usually build by using two different technologies.

DRAM (Dynamic RAM): DRAM stands for dynamic random access memory, a type of memory used in most computers. Dynamic Random Access Memory must have an electric current to maintain electrical state.

SRAM (Static RAM): In SRAM technology, the memory cells are made form digital gates and each cell can hold its value without any need to refresh the data as long as the power is supplied to it. As no refreshing is required to SRAM, these chips are faster than the DRAM chips also utilize less power. Because of these reasons the design of SRAM chip is more complex than the design of DRAM chips. Hence the SRAM chip is more expensive than the DRAM chip. In most modern computers this technology is used to build very fast memory inside a CPU. This memory is known as the cache memory.

Cache memory usually has a very small size as compared to the main memory in the computer but plays a very important role in increasing the performance of a computer system. This memory arrangement is shown in the figure 5.5 below:

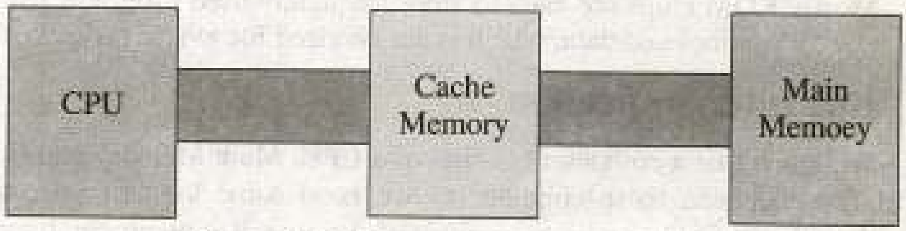


Figure 5.5: Memory arrangements

It is important to note that the main memory is **volatile (unstable)** and the contents of the memory are lost as soon as the electricity supply is cut-off. The CPU can not only read the data stored in RAM but also can write data in the RAM, so RAM is read/write memory. It is used to store all data and instruction of a program while it is being executed.

ROM (**R**ead **O**nly **M**emory): As is obvious from this name the contents stored in this memory can be read but new data can not be written onto it so it is read only. The manufacturer of the ROM writes the data and programs permanently onto it and this data and programs cannot be changed afterwards. ROM contains frequently used instructions and data.

Another commonly found form of **ROM** is **PROM** (**P**rogrammable **R**ead **O**nly **M**emory). This form of ROM is initially blank and the user or manufacturer can write data onto it by using special devices. Once the program/data is written onto PROM it can be changed or altered. It is obvious that this kind of ROM will be used for storing user made programs and data and the data should have a very long life time as the data written onto this kind of ROM can not be changed.

Another important form of Read Only Memory is **EPROM** (**E**rasable **P**rogrammable **R**ead **O**nly **M**emory). Like PROM it is initially blank. Programs and data can be written on it by the manufacturer or by the used by using special devices. Unlike PROM, the data written on it can be erased by using special devices using ultraviolet rays. So data/program written on it can be changed and new data can also be added on this form of ROM. As the data written on this kind of ROM can be changed so data that is to be updated can be written onto it but frequently changing data should not be written on this ROM.

Yet another form of ROM is **EEPROM** (**E**lectrically **E**rasable **P**rogrammable **R**ead **O**nly **M**emory). This kind of ROM can be re-written by using electrical devices and so data stored on this ROM can be easily modified.

It is important to note that all the forms of ROM described above are non-volatile so the data stored on these chips is not lost when electricity is cut-off. Mostly ROM chips are used to store frequently used programs like operating system routines and data, which is not changed for longer periods of time.

5.1 Bus Interconnection

We know that a computer consists of a CPU, Main Memory and I/O unit. For data to flow between these components we need some kind of interconnections, which is another very important component of the overall computer architecture.

These components are interconnected by using a set of parallel lines (**Conducting Wires**). Each of these lines can be used to transfer a sequence of bits from one component of the computer to the other component. This set of parallel lines is called **BUS**. This kind of a bus is shown in the figure 5.6 below:

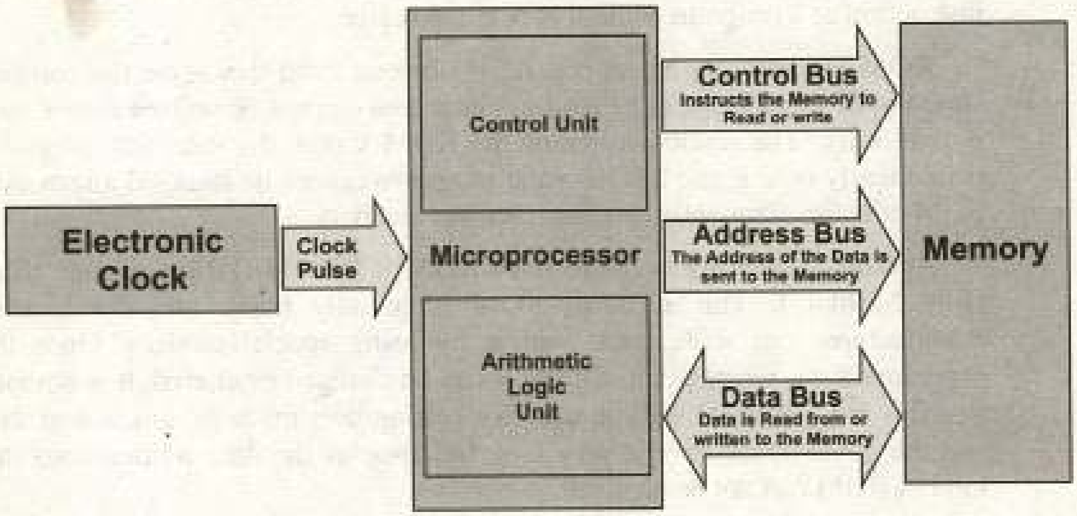

Control Unit
Control Bus
Instructs the Memory to Read or write
Electronic Clock
Clock Pulse
Microprocessor
Address Bus
The Address of the Data is sent to the Memory
Memory
Arithmetic Logic Unit
Data Bus
Data is Read from or written to the Memory

Figure 5.6: Bus interconnection

Generally a computer has more than one bus interconnection. The bus used to connect the main components of a computer is called the **system Bus**. General-purpose computers have a 70-100 line system bus. The system bus is divided into three main categories.

Control Bus: These lines are used to transmit different commands from one component to the other. For example, if the CPU wants to read data from the main memory; it will use the control bus to send the **memory read** command to the main memory of the computer. The control bus is also used to transmit other control signals like ACKS (Acknowledgement signals). For example when CPU give a command to the main memory for writing data, the memory sends a acknowledgement signal to the CPU after writing the data successfully so that the CPU can move forward and perform some more actions. A few commonly used commands and their purpose are given in the table below.

MEMORY WRITE	This command is used to write some data to a given location in the main memory.
MEMORY. READ	This command is used to read some data from a given location in the main memory.
I/O WRITE	This command is used to write some data to a given output device.
I/O READ	This command is used to read some data from a given input device.
BUS REQUEST	This command is used to request for a control on the bus so that the requesting device can use it to transmit data.
BUS GRANT	This command is used by the bus controller to indicate the grant of the bus to a device.
TRANSFER ASK	This command is used deliver information that the data was read by the devices.

Data Bus: On the system bus 32 or 64 lines are reserved to transfer data from one component to the other. These lines are commonly known as the data bus. A 64-line data bus can transfer 64 bits of data simultaneously so it is not difficult to see that the width of the data bus has a direct impact on the performance of the computer.

Address Bus: As we know that many components are connected to one another through the system bus so it is important to assign a unique ID to each component. This ID is called the address of that component. When a computer component wants to communicate with another, it uses a few of the system bus lines to specify the destination component by using its address. These lines are commonly known as the address bus. Not only the address is used to identify different components of a system but it is also used to specify different memory locations within the main memory.

For example; if the CPU wants to write some data at a location 9872 in the main memory it places the address of main memory and location (i.e. 9872) on the address bus. When the main memory sees its address on the address bus it reads the data from the data bus and writes it to the specified location with in the main memory.

As the number of components connected to the system bus increases more components will be trying to use the system bus simultaneously. This will slow-down the computer as components will have to wait longer to get access to the bus. To solve this problem only the major components of the computer are connected to the system bus and remaining components are connected to another bus usually known as the expansion bus. The expansion bus is connected to the system bus. This is shown in the figure 5.7 below.

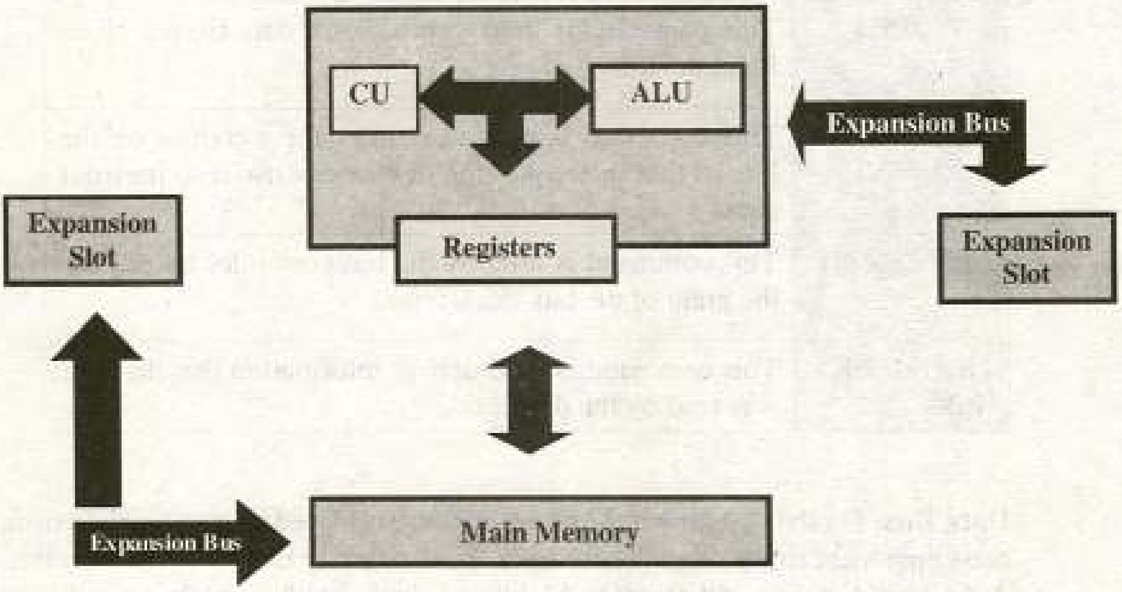


Figure 5.7: Expansion Bus

5.2 The I/O Unit

The I/O Unit is another very important component of a computer. Now a days we have many input/output devices like keyboard, mouse, disks etc. All these devices are very different from one another in their organization. Also these devices can handle different data-transfer rates and support different data formats. Because of all these differences it is impractical to connect all these devices directly to the system bus. It is not sensible to require the CPU to control these devices directly as they will take a lot of CPU time and will fill the system bus capacity.

To avoid these difficulties, a special hardware component I/O unit is used. Only the I/O unit is connected to the bus and the processor and all other devices are connected to it as shown 5.8 below.

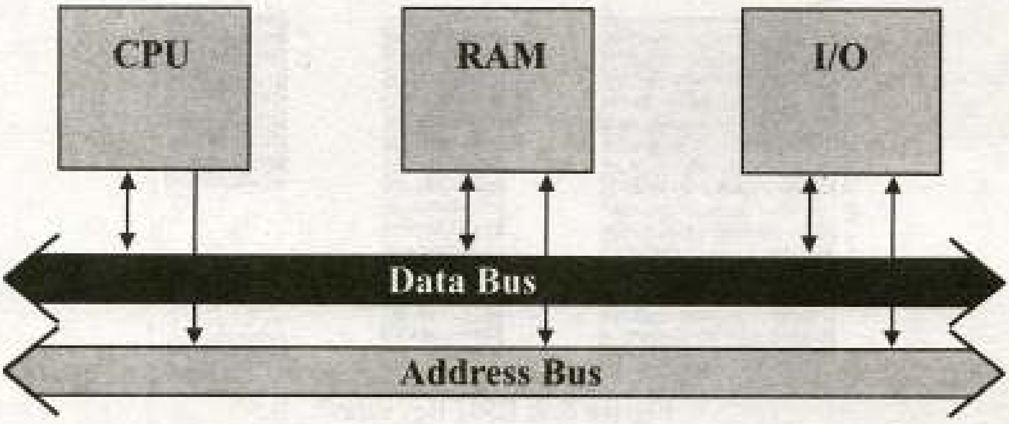


Figure 5.8: Hardware Components

The I/O unit is responsible for keeping the track of states of different devices attached with it. It is also responsible for compensating the speed difference between the processor and the I/O devices. There are two main ways of transferring data from the peripherals into the computer.

Interrupts: In this scheme the processor issues the command to the I/O devices. When the devices get ready, they generate an interrupt signal for the processor. On sensing this signal, the processor suspends all other processing and performs the I/O operation. The disadvantage of this scheme is that it reduces the over all performance of the processor.

DMA: The second scheme is DMA. In this scheme the processor issues the I/O command and then gets busy in some other useful task. The special hardware gets the data from the I/O device and uses the system bus to place if in the main memory. It is useful to note that the data is transfused when the processor does not need the system bus. So the processor does not have to wait for the I/O operation to complete. The disadvantage of this scheme is that it is more complex and extensive, as more hardware is needed.

5.2.1 CPU Registers

The program is stored in the main memory of the computer on contiguous memory locations. The data is also loaded into the computer's memory before the processing starts and then the control is given to the CPU.

The CPU needs storage areas where the data can be stored temporarily. As these storage areas are used frequently, so for efficiency these special-purpose temporary storage areas are provided within the CPU for enhancing the performance of the CPU. These special purpose storage areas are called **registers**. The figure 5.9.given below shows the most commonly used CPU registers.

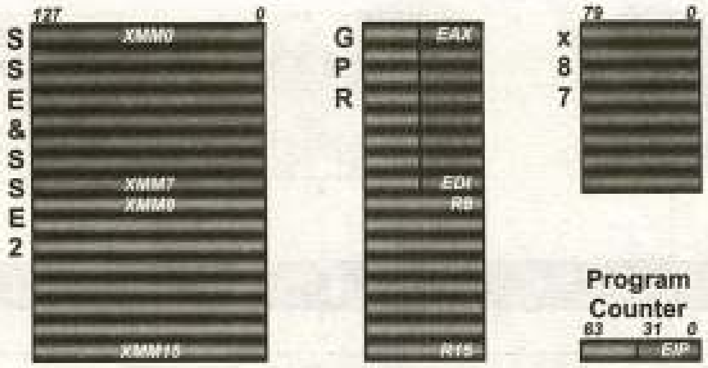


Figure 5.9: CPU Register

PC (Program Counter): This register holds the address of the next instruction to be fetched for execution. As soon this instruction is fetched, its value is incremented so that it still has the address of next instruction.

IR (Instruction **R**egister): Once the instruction is fetched it is stored in the **IR** where this instruction is decoded.

MAR (Memory **A**ddress **R**egister**):** When the CPU wants to store some data in the memory or reads the data from the memory, it places the address of the required memory location in the MAR.

MBR (Memory **B**uffer **R**egister**):** The CPU uses this register to store data coming from the memory or going to the memory.

Stack Pointer: To understand the purpose of this register it is important to understand a very important data structure (Arrangement of data) called a Stack.

GPR(General **P**urpose **R**egisters**):** These registers are called EAX, EBX, ECX, EDX and can be used for any mathematical or logical operations. These are used for arithmetic and data movement purposes. Each can be divided into an upper case and lower byte called AH, AL, BH, BL, CH, CL, DH, DL respectively. A stand for Accumulator, B for base, C for count and D for data Each of these registers can alternately be used as one byte, two byte, or four byte registers, AL (1 byte), AH (1 byte),AX (2 byte), EAX (4 byte).We can access 16 bit or 8 bit.

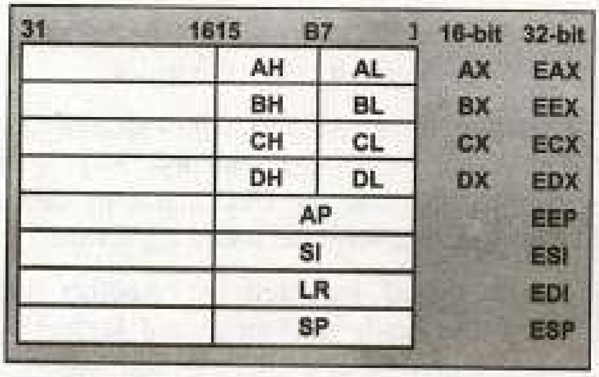


Figure 5.10: General Purpose Registers

AX (Accumulator Register): Used for arithmetic and data operations.

BX (Base Register): Used for arithmetic and data movement and special addressing abilities.

CX (Counter Register): Used for counting purpose. Acts as a counter for repeating or looping.

DX (Data): Has special role in division and multiplication.

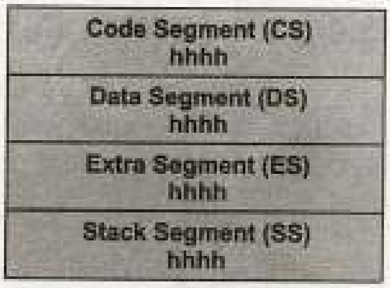
Address or Segment Registers: The address or segment register is a group of 4, some times registers named CS, DS, ES, SS. The segment register used as base location for program instruction, data, and the stack.

CS(Code Segment): The CS register holds the base location of all executable instructions (code) in the program.

DS(Data Segment): The DS register is the default base location for memory variables. The CPU calculates the offsets of variables using the current value of DS.

ES(Extra Segment): The ES register is an additional base location for the memory variables.

SS(Stack Segment): The SS register contains the base location of the current program stack.



Each has 2-byte. These registers are called segment register and are used in conjunction with either the IP register or two index registers DI and SI to address various areas of computer memory .CS is the primary register or two index register used to fetch instruction in conjunction with the IP register. DS is the primary register, used to point out data in the computer memory along with the DI or SI registers.

5.3 Computer Operations and the instruction format

We know the basic architecture of a computer, now we will learn about different types of operations performed by the computer.

Data transfer instructions: All CPUs provide different instructions for the transfer of data from and to the memory. A programmer can use these instructions to bring data into the CPU and copy data from the CPU to the main memory. Instructions have the following format.

Arithmetic and Logical instructions: Another important category of operations a CPU can do is Arithmetic and logical operations. Most CPU provides the basic arithmetic operations of add, subtract, multiply and divide for signed numbers and floating point numbers. Logical operations of comparing two numbers, performing XOR of number, shifting and rotating a number are some common form of logical operations provided by the CPU.

I/O Instructions: Every CPU provides if users with the operations of reading data from a peripheral device and writing data to a peripheral device. To use these operations a programmer may use input and print commands provided by the CPU.

Control Transfer: In all real world programs, given to the CPU must be repeated a number of times. To support such operations, all CPUs provide its programmers with control flow operations some examples of these operations are:

Jump, Jumpz (Jump if zero)

Instruction set: Each CPU provides its users with a number of instructions so that the users can perform different operation supported by the CPU. The set of all instructions provided by a CPU is commonly known as the ***instruction set*** of that CPU. In this section we will take an overview of the instruction set provided by the most modern CPU manufacturers and see how these instructions are used to solve different problems. Most commonly the CPU provides the following instructions to the programmers.

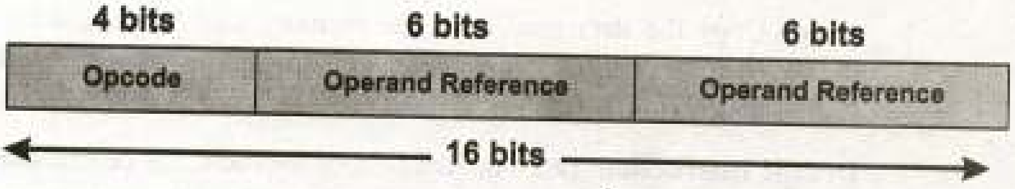
5.4 Instruction Format

A computer will usually have a variety of instruction code formats. It is the function of the control unit within the CPU to interpret each instruction code and provide the necessary control functions needed to process the instruction.

Each instruction for the CPU is specified by giving.

- A code for the instruction (opcode).
- Addresses of the operands.

Although other things (like addressing mode) are also specified but in most general form the instruction is specified in the format given in the figure below



Zero-Address Instruction Format: The name **"zero-address"** is given to this type of computer because of the absence of an address field in the computational instructions. A stack-organized computer does not use an address field for the instructions ADD and MUL. The PUSH and POP instruction, however, need an address field to specify the operand that communication with the stack.

One-Address Instruction Format: One-address instructions use an implied accumulator (AC) register for all data manipulation. For multiplication and division there is a need for a second register.

Two-Address Instruction Format: For two addresses instruction, each address field can again specify either a possible register or a memory address. Two-address instructions are the most common in commercial computers. Examples of such instructions are MOV, ADD, CMP and BIS.

Three Address Instruction: Computer with three-address instruction formats can use each address field to specify either a processor register or memory operand. The advantage of the three-address format is that it results in short programs when evaluating arithmetic expression. The disadvantage is that the binary-coded instructions required too many bits to specify three addresses. The instruction formats in the computer are restricted to either three register address fields or two register address fields and one memory address field.

Fetch-decode-Execute cycle: Now that you know the basic architecture let us see how the CPU enacts the instructions, specified in a program.

When we want to execute a sequence of instructions those instructions/data are first of all loaded into the main memory of the computer by using some I/O device. Once these instructions have been loaded into main memory, the address of the first instruction is copied into the program counter and the control is given to the CPU. The CPU performs the following steps:

Fetch Instruction: The CPU reads the value of PC and the instruction pointed to by PC into the instruction register.

This fetching of instruction involves the following steps:

- Copy the contents of PC into the MAR and request a memory read.
- Copy the data read from the memory into MBR and instruction then in the IR. Increment PC so that it points to the next instruction.

Decode Instruction: Once the fetching of instruction is completed the CU decodes the instruction by analyzing the opcode of the instruction. It also reads the values of operands specified in the instruction. Decoding means activating the appropriate circuit to execute the instruction.

Execute Instruction: After decoding the instruction the processor executes the instruction by using the activated circuit. Then the results of the execution are written back to registers and memory.

The CPU repeatedly does these steps. These steps are known as **Fetch-Decode-Execute** Cycle.

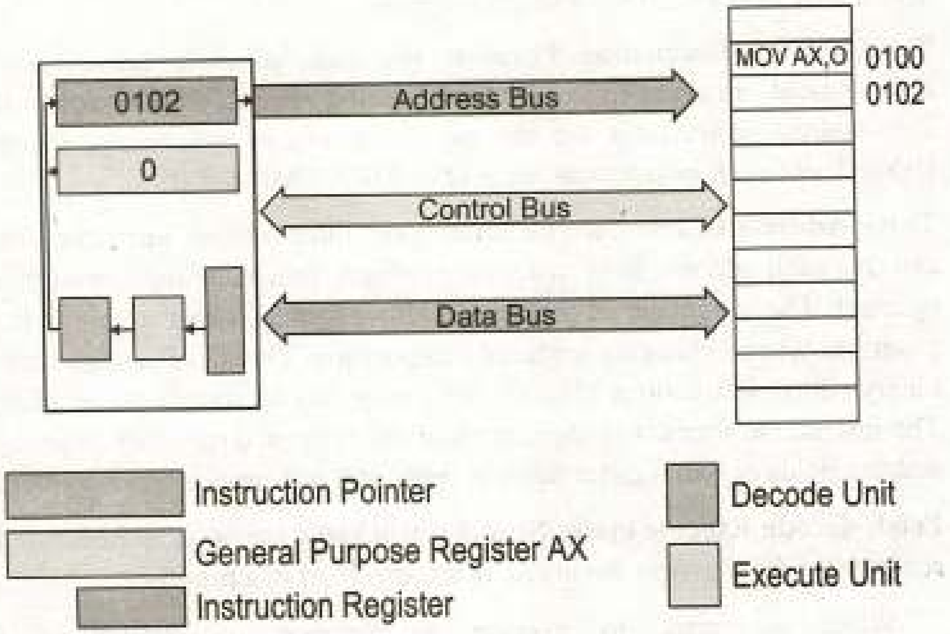


Figure 5.11: Fetch-Decode-Execute

5.5 Software

The term software is used for a sequence of instruction given to the computer to perform a specific task. Software consists of the step-by-step instructions that tell the computer what to do. In general, software is divided into applications software and software. Applications software, which may be customized or packaged, performs useful work on general-purpose tasks. System software, which includes operating system, enables the application software to interact the computer.

Application Software: Application software is defined as software that can perform useful work on general-purpose tasks. It may be either customized or packaged.

- **Customized software** is software designed for a particular customer. The program can be developed by a single computer professional programmer or by a team of programmers depending upon the requirements.
- **Package software** is the kind of "off" the "self" program developed for sale to the general public. Package software includes word processing, spreadsheet, database manager, graphics, and communications programs. These are the software development by experts in high/low level languages for non-experts. It is to facilitate all the fields of life.

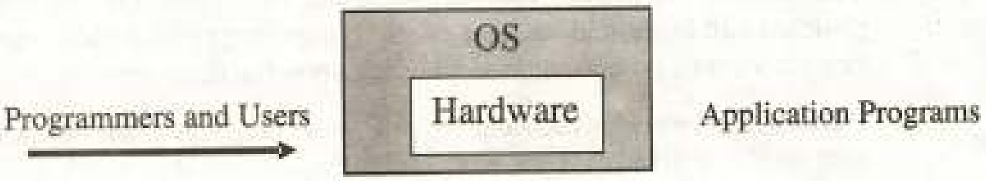
5.5.1 Operating Systems

It is obvious that to solve some problem on a computer a programmer will write instructions. But other than writing instructions for solving the problem every programmer will also have to write instructions for the following tasks.

- Read data from the input devices
- Show results on the output devices
- Perform memory management tasks (more details latter)
- Organize data on the storage devices.

These tasks are very complex and only expert programmers can write these instructions. From this discussion it is obvious that only expert programmers can write programs and use computers. To overcome this difficult situation some programs (instructions) can be written only once and can be stored in the computer so that every programmer does not have to write these instructions but rather use the stored instructions. This set of programs evolved and became what is known as an operating system (OS). We can define an OS as **a set of programs running in the background on a computer system and providing an environment in which other programs can be executed and the computer system can be used efficiently.**

In this section we will discuss the main functions of an operating system and see how the OS provides these facilities. Some of the programs are stored in the processor memory and provide the basic utilities to the users. The remaining programs are stored on the hard disk or backing store of the computer and are loaded in the memory of the computer when these are needed. Any programmer or user trying to use the computer system now does not need to write instructions for performing the common tasks but can issue command to the OS and OS can do the rest. So we can visualize as if the OS is sitting between the hardware and the application program or user. This is shown in the figure below. From this figure it is obvious that the OS will not only provide programs for doing different tasks but will also provide an interface to its users (i.e. programs, programmers etc)



Functions of Operating Systems: In this section we will discuss the main functionality provided by the OS to its users.

Manage Hardware Resources: The operating system must provide programs for managing the hardware resources of the computer like disks, memory, and CPU.

Memory Management: In the stored program computer every program has to be loaded into the computer's main memory during the execution. If there are several programs simultaneously loaded in the main memory, as is the case when time-sharing is used, the program and its data must be protected from the actions of other programs.

Load and Execute programs: As we know that a program has to be loaded into the main memory before the processor can execute it. The OS provides the facility of easily loading a program into memory and starts its execution.

Data Security: The OS must also protect the user data against illegal access and modification.

Providing interface to the users: The OS must provide an interface between the user and the computer and also between software and the computer. Most Operating systems provide the following two types of interfaces to their users.

Command prompt: In such interfaces the users communicate with the operating system by typing commands using a keyboard. Each command given to the OS activates one of the many programs in the OS. Example of such an interface is the command prompt provided by MS-DOS to its users.

Graphical User Interface (GUI): The GUI interface consists of Window, Menus, Icons and pointers. The user of the system communicates with the OS by selecting commands from the menus or by selecting different Icons with the pointing device. MS-Windows is a well-known example of the OS with a GUI interface. In MS-Windows the user selects commands by using a mouse and a keyboard.

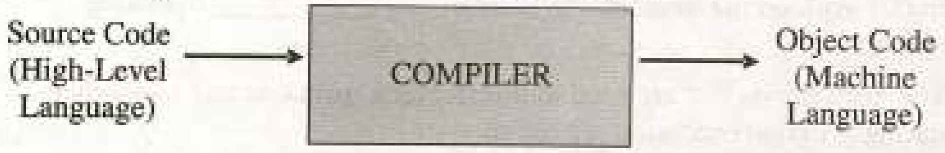
5.6 The Translators and their functions

Interpreters and Compilers

In the early days of computers the programs had to be written in machine code. The machine code instructions are the instructions provided as the instruction set of the machine and these codes are represented by a binary pattern in the computer. To write a program the programmer had to write the instructions in binary. This was a very complex task and even writing very simple programs took a log time. It was very difficult to produce correct programs as detecting errors in the programs and correcting these errors (Debugging the program) was very difficult. A program that produces the binary instruction for a given assembly language program is called an **Assembler.**

Each assembly instruction maps to a single machine instruction so it is very easy to translate a program written in assembly language to machine code. Writing programs in assembly language is very easy but is still a tedious work and took a long time.

The program that translates a high-level language program into machine language is called a **compiler**. Once a program has been translated into machine code it can be loaded into the main memory and executed by the CPU. The high-level language version of the program is usually called the **source code** and the resulting machine code program is called the **object code**. The relationship between the source code and the object code is shown in the figure below:


Source Code (High-Level Language)
COMPILER
Object Code (Machine Language)

A more appropriate definition of a compiler is that it is a program that takes as input a high-level language program and generates an object program. This object program may or may not be the absolute machine code.

Another useful translator is an **interpreter**. An interpreter takes as input a high-level language program and performs the following actions.

- It repeatedly reads instructions (one at a time) and translates it to machine code.
- It then executes the instruction

One difference between a compiler and an interpreter is that a compiler converts each instruction only once but an interpreter may translate an instruction several times. Clearly if an instruction has some error an interpreter can easily identify it. Also an interpreted program runs slower than a compiled program as once a program is compiled it does not need any further translation but the original program has to be translated every time it is executed by an interpreter. And some instructions will be translated several times.

Exercise 5C

1. Fill in the blank:

(i) DMA stands for ________ .

(ii) In ________ mode, data can be transmitted in both directions simultaneously.

(iii) The lexical analyzer also commonly known as a________.

(iv) The ________reads the instructions from the memory and decodes these instructions.

(v) The ________interface consists of Window, Menus, Icons and pointers.

(vi) EEPROM stands for _______.

(vii) Stack pointer register is used for _________.

(viii) DRAMs require _________ refreshing to maintain data storage.

(ix) __________language was developed for business applications.

(x) __________ register is used in mathematical or logical operations.

(xi) DVD stands for ___________.

(xii) BIOS stands for ___________.

(xiii) Initial work on the internet was done in ____________ operating system.

(xiv) The instructions that are used to transfer data from one unit to another during program execution are called ________.

(xv) _______ registers are used in mathematical and logical operation.

2. Choose the correct answer.

(i) Data and program not being used by computer are stored in:
(a) Secondary storage (b) cache (c) Primary storage (d) printer

(ii) A set of instructions that run the computer are:
(a) hardware (b) document. (c) CPUs (d) software

(iii) The program that contains instructions to operate a device is called.
(a) Device driver (b) Device operator (c) Device linking (d) Device system

(iv) CPU is an example of:
(a) software (b) A program (c) hardware (d) an output unit

(v) The address of instruction under the processor execution is contained within:

(a) Program Counter (b) Current Instruction register.

(c) Memory Address register (d) Memory Buffer register

(vi) A computer drives its basic strength from

(a) Speed (b) Memory

(c) Accuracy (d) All of above

(vii) The arithmetic/logic unit performs the following actions

(a) Control computer operations

(b) Perform arithmetic functions such as addition and subtraction etc.

(c) Perform logical comparisons such as equal, greater than, less than

(d) Both b and c

(viii) Which is a storage device

(a) CPU (b) Clock (c) Floppy disk (d) Bus

(ix) Which component is responsible for comparing the contents of two pieces of data

(a) ALU (b) Control unit (c) Memory (d) None

(x) Which one is faster

(a) RAM (b) Cache (c) Register (d) Hard disk

3. Write T for true and F for false statement.

(i) Bps stands for byte per second.

(ii) In simplex transmission mode, communication can take place in both directions.

(iii) Random access memory is volatile memory.

(iv) Operating system is an application program.

(v) External buses and internal buses are similar.

(vi) Accumulator register is used to control the stacks in the computer.

(vii) LIFO stands for Last-in-first-off.

(viii) Expansion slot is a place where an expression card is fitted.

(ix) Static Ram holds the data as long as power is supplied to it.

(x) The clock of the computer ticks once in one second just like an ordinary clock.

(xi) VDU is an input device.

4. What is CPU? Describe briefly.

5. Explain the architecture of computer system.

6. Differentiate between the following.

(i) PROM and EPROM

(ii) Address Bus and Control Bus

(iii) Serial and Parallel Ports

(iv) Linker and Loader

(v) CU and ALU

7. Differentiate b/w Compiler and Interpreter.
8. How to transfer data from CPU to memory explains in steps?
9. Define the different types of RAM.
10. Define the general-purpose registers.
11. Describe the Bus and its types.
12. Define the machine Instructions.
13. Differentiate between Fetch instruction and Decode instruction.
14. Differentiate between ROM and RAM.

Answers

1.	(i) Direct Memory Access	(ii) Full Duplex	(iii) Lexer	(iv) Control Unit
	(v) Graphical User	(vi) Electrically Erasable Programmable Read Only Memory		
	(vii) Maintaning Stack	(viii) Periodic	(ix) COBOL	(x) General Purpose
	(xi) Digital versatile disk	(xii) Basic Input/Output System		(xiii) Unix
	(xiv) Read/Write Instruction	(xv) General Purpose		

2.	(i) a	(ii) d	(iii) a	(iv) c	(v) a
	(vi) d	(vii) d	(viii) c	(ix) a	(x) c
3.	(i) T	(ii) F	(iii) T	(iv) F	(v) F
	(vi) F	(vii) F	(viii) F	(ix) F	(x) F
	(xi) F				