Chapter 1

Structure of Computer System

Syllabus

- Brief History of computers
- Von Neumann Architecture
- Functional Units
- Bus structures and Interconnection networks
- Performance

1.1 The Von Neumann Architecture :

Dec. 2005, May 2007

The basic function performed by a computer is the execution of a program. A program is a set of machine instructions. An instruction is a form of control code, which supplies the information about an operation and the data on which the operation is to be performed. A typical Von Neumann machine consists of five functionally independent units :

i) Input unit
ii) Output unit

iii) Arithmetic and logic units
iv) Control unit
v) Memory unit

Fig. 1.1 shows the basic structure of a conventional Von Neumann machine.

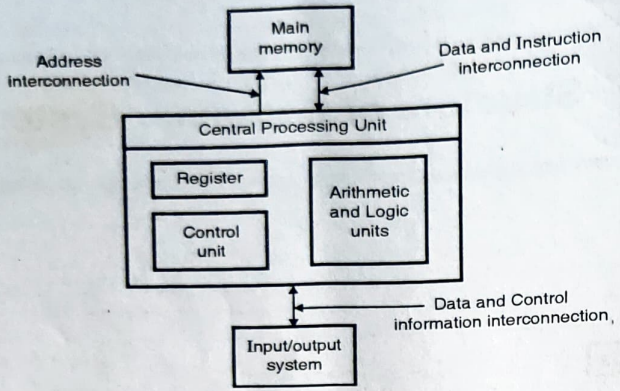


Fig. 1.1 : Structure of a computer

1.1.1 Input Unit :

Computers accept coded information through input units, which read the data. The most common input device is the keyboard. Whenever a key is pressed, the corresponding letter or digit is translated into its corresponding code and sent to the processor. There are many other kinds of input devices like :

1) Mouse
2) Joysticks
3) Floppy drive
4) CD-drive
5) Modem

1.1.2 Output Unit :

Output unit sends processed result to outside world. The familiar example of such device is a printer. There are many other kinds of output devices like :

1) Video terminal
2) Floppy drive
3) CD-drive
4) Modem

1.1.3 Arithmetic and Logic Unit (ALU) :

Arithmetic or logic operations like multiplication, addition, division are performed by ALU. Operands are brought into the ALU, where the necessary operation is performed.

1.1.4 Control Unit :

Operations of ALU, memory and input and output units are coordinated and controlled by the control unit. Control unit sends control signals to other units. Data transfer between processor and memory are controlled by the control unit through timing signals.

1.1.5 Memory Unit :

Main memory is needed in a computer to store instructions and the data at the time of program execution. It was pointed out by Von Neumann that the same memory can be used for storing data and instruction. The memory unit stores all information in a group of memory cells as binary digits. Each memory location has a unique address and can be addressed independently. The contents of the desired memory locations are provided to the central processing unit by referring to the address of the memory location. The amount of information, which can be transferred between CPU and memory, depends on the size of the BUS connecting the two.

1.1.6 Key Features of a Von Neumann Machine :

- The Von Neumann machine uses stored program concept. The program and data are stored in the same memory unit. The computers prior to this idea used to store programs and data in separate memories. Entering and modifying these programs were very difficult as they were entered manually.
- Each location of the memory can be addressed independently.
- Execution of instruction in Von Neumann machine is carried out in a sequential fashion (unless explicitly altered by the program itself) from one instruction to the next.

1.1.7 Basic Structure of the CPU :

The design of CPU in modern form was proposed by John Von Neumann and his colleagues for the IAS computer. The IAS computer had a minimal number of registers along with the essential circuits. This computer had a small set of instruction and an instruction was allowed to contain only one operand address. Fig. 1.2 gives the structure of IAS computer. The structure shown in Fig. 1.2 consists of the following registers.

Accumulator (AC) :

It interacts with ALU and stores the input or output result.

Data Register (DR) :

It acts as a buffer storage between the main memory and the CPU.

Program Counter (PC) :

It contains the address of the next instruction to be executed.

Instruction Register (IR) :

Holds the current instruction.

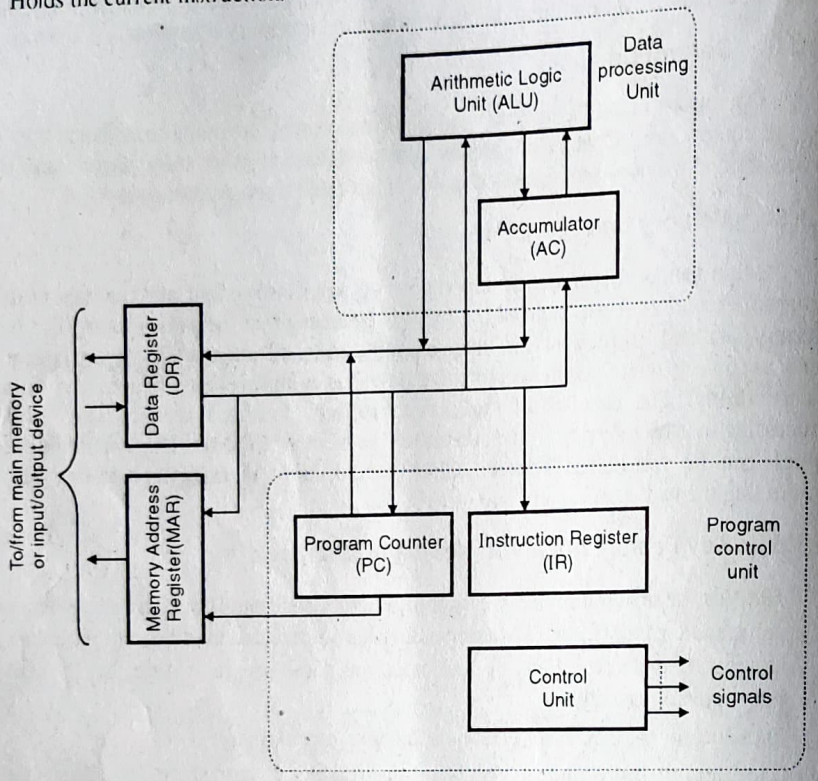


Fig. 1.2 : Basic structure of the CPU

Memory Address Register (MAR) :

It provides address of memory location from where data is to be retrieved or to which data is to be stored.

1.1.8 Execution of a Program by Von Neumann Machine :

- Program to be executed is loaded in main (primary) memory.
- A special register, PC (Program counter) is made to point to the first instruction of the program. CPU fetches the instruction pointed by PC. PC contents are subsequently changed to point to the next instruction.

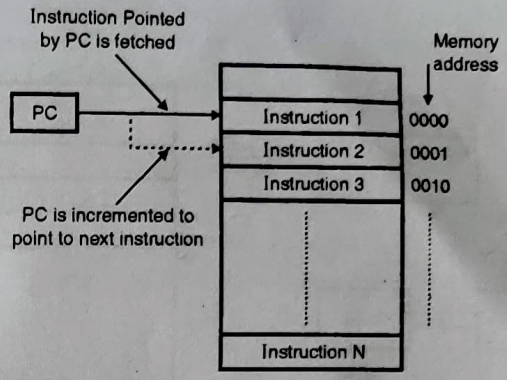


Fig. 1.3 : Instruction pointed by PC is fetched by the CPU for execution. Subsequently PC is made to point to the next instruction

- If the initial value of PC = 0000 (in binary), instruction number 1 will be fetched for execution. After fetching "Instruction number 1", PC will be incremented by one. It is assumed that the size of each instruction is 1 byte.

$$PC = PC + 1 = 0 + 1 = 1$$

Fetching an instruction :

- CPU interacts with memory through two special registers :
  (1) MAR (Memory Address Register) : It provides address of memory location from where data is to be retrieved or to which data is to be stored.
  (2) DR (Data Register) : It acts as a buffer storage between the main memory and the CPU.
- Instruction to be executed next is brought from the memory to the CPU, through the following steps :
  (1) The next instruction address is transferred from PC to MAR.

$$MAR \leftarrow PC$$

  (2) MAR puts this signal on the address bus for selection of the required location of the memory.
  (3) Controller generates the $\overline{RD}$ (read control signal) signal to perform read operation on memory. Required instruction flows on data bus. Instruction on data bus is accepted in DR (Data Register).

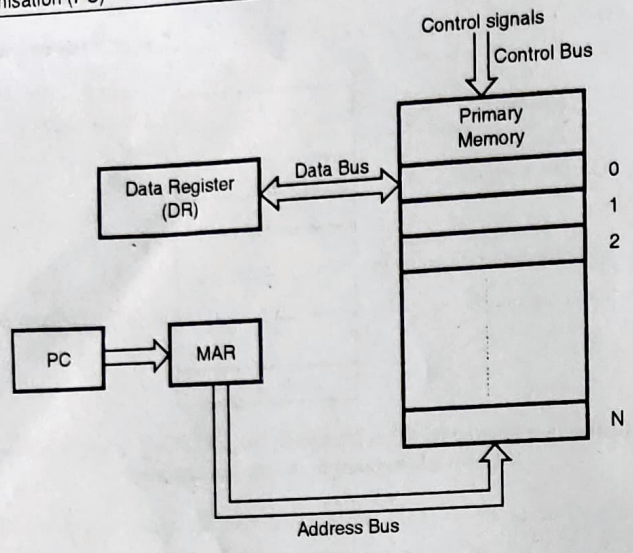


Fig. 1.4 : Fetching an instruction from memory

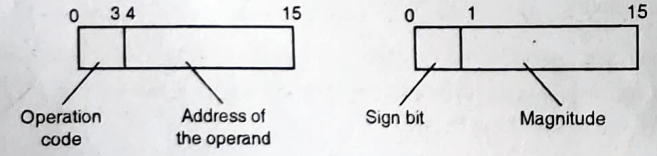
Execution of instruction :

- The fetched instruction is in the form of binary code and is loaded into instruction register (IR) from DR (Data Register). The instruction specifies what action the CPU has to take.
- The CPU interprets the instruction and performs the required action. The action could be :
  (1) CPU to memory data transfer or memory to CPU data transfer.
  (2) CPU to I/O or I/O to CPU data transfer. Data may be transferred to or from the outside world.
  (3) The CPU may perform some arithmetic or logic operation on data.

Example :

Let us assume a hypothetical machine which has a 16 bit instruction and data. Each instruction of the machine consists of two components :

(1) Operation code (2) Address of operand in memory



(a) Instruction format **(b) Data format**

Fig. 1.5 : Instruction and data format of the hypothetical machine

The memory is divided into words of 16 bits. Fig. 1.5 shows the instruction and data formats for the machine.

The machine can have $2^4 = 16$ possible operation codes. Let us assume operation codes as :

0001 as "Load accumulator (AC) with the content of memory"

0010 as "Store the content value of accumulator in the memory"

0011 as "Add the value from memory to the accumulator"

Let us try to understand execution of machine instructions with the help of the following operation.

$z = x + y$

↓ Addition operation

Assignment operation

Equivalent basic operations :

(i) Load accumulator with x

(ii) Add y to accumulator

(iii) Store contents of accumulator in z

Let us assume that the addresses of x, y and z are as given below :

	Binary	Hexadecimal
Address of x =	0000 0000 0100	004H
Address of y =	0000 0000 0101	005H
Address of z =	0000 0000 0110	006H

Three instructions given above will be encoded as given below :

Operation	OP code	Address	Instruction in hexadecimal
Load accumulator with x	0001	0000 0000 0100	1004H
Add y to accumulator	0010	0000 0000 0101	2005H
Store accumulator in z	0011	0000 0000 0110	3006H

Let us assume that these instructions are loaded in three memory locations (all addresses are in hexadecimal notation) 001H, 002H and 003H. Initial values of x and y are 5 and 10 respectively.

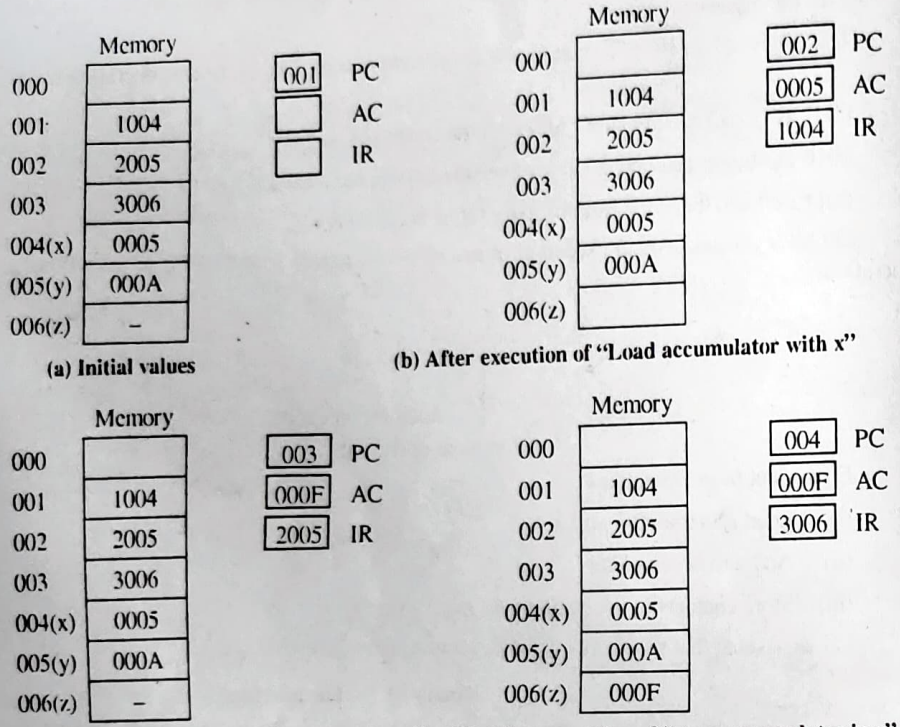


Fig. 1.6 : Memory and registers contents on execution of the three consecutive instructions

- Since, the PC contains value 001H, the address 001H stored in PC is passed to MAR.
- Memory location 001H is accessed and its contents 1004 are stored in DR (Data Register).
- Contents of DR are passed to IR and PC incremented by 1.
- IR has the value 1004H, which is decoded as "Load the content of address 004H" in the accumulator.
- Thus, the accumulator is loaded with the content of location "004H", which is "0005H".
- Similarly, next two instructions are executed. Finally, the sum of x and y are stored in memory location 006H(Z).

Instruction cycle :

Please note that the execution of the instruction in the above example requires only one data transfer operation. Does an instruction require more than one memory reference or operand addresses ? Well, one such instruction is "ADD Ax, X". The execute cycle of this instruction may consist of steps such as :

- Decode the instruction, which is ADD.

- Read contents of Ax.
- Read memory location X into the CPU.
- Add the two values.
- Write back the result.

Thus, in general, the execution cycle for a particular instruction may involve more than one stages and memory reference. In addition, an instruction may ask for an I/O operation. Considering above situations, the detailed view of instruction cycle is shown in Fig. 1.7.

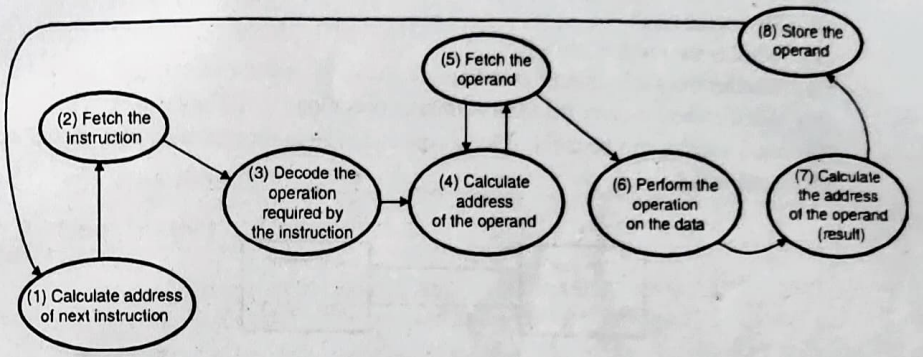


Fig. 1.7 : Instruction cycle

The instruction cycle shown in Fig. 1.7 consists of following stages :

- First the address of the next instruction is calculated based on the width of instruction and memory organization. For example, if in a computer an instruction is of 16 bits and if memory is organized as 16-bit words, then the address of the next instruction is evaluated by adding 1 in the address of the previous instruction. In case, the memory is organized in bytes, which can be addressed individually, then we need to add 2 in the previous address to get the address of the next instruction.
- Then, an instruction is fetched from memory location pointed by PC to the CPU.
- The instruction is decoded to determine the type of operation desired and what are the operands to be used.
- In case, the operands need to be fetched from memory or via I/O, then address of the memory location or I/O device is calculated.
- Next, the operand is fetched from the memory or read from I/O device.
- Now, the operation asked by the instruction is performed.
- Finally, the result is written back.

1.1.9 An Advanced Structure of Von Neumann Machine :

An advanced machine based on Von Neumann architecture is shown in Fig. 1.8. It is more powerful machine. Few additional features have been added. Some of them are :

- Provide additional registers for storing operands and addresses. A single register AC (Accumulator) is replaced by a set of registers. These are general purpose registers and they can store either operands or addresses.
- Increased capability of ALU circuit. For example, most microprocessors have capabilities for performing addition and subtraction. This capability with only little extra circuit can be used for multiplication and division.
- Includes a special register to facilitate conditional jumps within a program. A status register which gives information about various conditions like :
  - (i) Sign of the result.
  - (ii) Whether the result is zero.
  - (iii) Whether there is arithmetic overflow.
  - (iv) Whether there is carry out from arithmetic operation.
- This status register can be checked for a typical condition for execution of a conditional branch instruction.

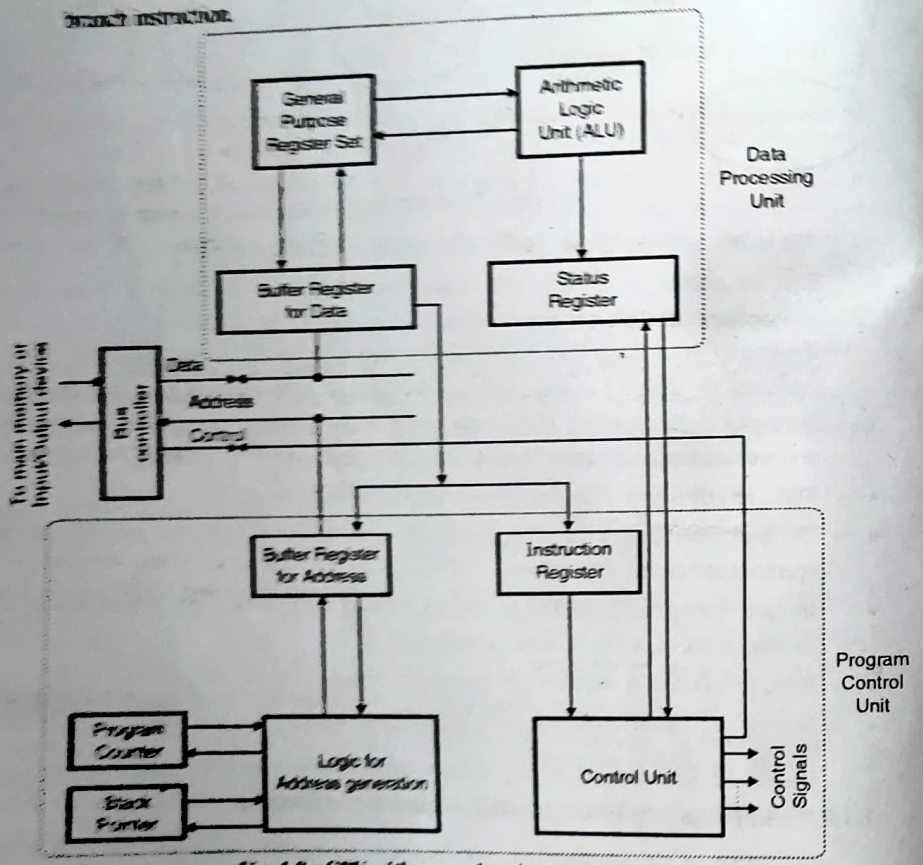


Fig. 1.8 : CPU with general register organisation

- We know that a **call** instruction is implemented through a stack. Return address is pushed on top of the stack. It includes a special register for transfer of control between different subroutines or subprograms or interrupts. It uses a special area in main memory for holding data of stack and a special register called stack pointer (SP). Stack pointer points to the top of the stack.
- Includes a special circuit for address generation to facilitate a number of addressing modes, like :
  - (1) Direct addressing
  - (2) Immediate addressing
  - (3) Register addressing
  - (4) Indirect addressing
  - (5) Indexed addressing
  - (6) Stack addressing

1.2 Brief History of Computers :

History of computers gives us the basic information about the technological development trends in computer in the past and its projections in the future.

The ancestors of modern age computers were the mechanical and electromechanical devices. This ancestry can be traced as back as 17th century, when the first machine capable of performing four mathematical operations, viz., addition, subtraction, division and multiplication.

1.2.1 Mechanical Computers :

Blaise Pascal made the very first attempt towards the automatic computing. He invented a device, which consisted of lots of gears and chains and used to perform repeated additions and subtractions. This device was called Pascaline. Later many attempts were made in this direction; we will discuss some details about the innovation by Charles Babbage, the grandfather of modern computer. He designed two computers.

The difference engine :

It was based on the mathematical principle of finite differences and was used to solve calculations on large number using a formula. It was also used for solving the polynomial and trigonometric functions.

The analytical engine by Babbage :

It was a general purpose computing device, which could be used for performing any mathematic operation automatically. It consisted of the following components :

- **The store :** A mechanical memory unit consisting of sets of counter wheels.
- **The mill :** An arithmetic unit, which is capable of performing the four basic arithmetic operations.

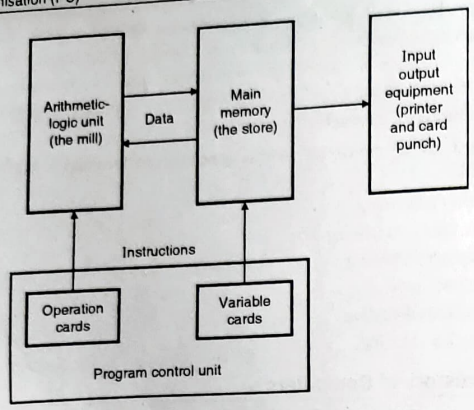


Fig. 1.9 : Structure of Babbage's analytical engine

- **Cards :** There are basically two types of cards :
  - **(i) Operation cards :** Selects one of the four arithmetic operations by activating the mill to perform the selected operation.
  - **(ii) Variable cards :** Selects the memory location to be used by the mill for a particular operation (i.e. the source of the operands and the destination of the results).
  - **(iii) Output :** Could be directed to a printer or a cardpunch device.

  The basic feature of this analytical engine were :
- It was a general purpose programmable machine.
- It had the provision of automatic sequence control, thus enabling programs to alter its sequence of operations.
- The provision of sign checking of result existed.
- Mechanism for advancing or reversing of control card was permitted thus enabling execution of any desired instruction. In other words, Babbage has devised a conditional and branching instructions. The Babbage machine is fundamentally the same as a modern computer.

1.2.2 The First Generation : Vacuum Tubes :

The first electronic computer was constructed using vacuum tubes. The first computer constructed using vacuum tube technology was ENIAC (Electronic Numerical Integrator and Calculator).

- It was enormous machine weighing about 30 tons.
- It contained more than 18,000 vacuum tubes.

- It consumed 140 kilowatts of power.
- It was a decimal rather than a binary machine.
- It was capable of 5000 additions per second.
- It had memory to hold twenty 10 digit decimal numbers.
- It stored programs and data in separate memories.

EDVAC (Electronic Device Variable Computer) :

ENIAC stored program and data in separate memories entering or altering program was a difficult task , EDVAC stores both program and data in the same memory.

- Concept of common memory for both program and data.
- EDVAC has two kinds of memory : a fast main memory with a capacity of 1024 and slower secondary memory of 20k words.
- It stored and processed number in binary form to minimize hardware cost. It processed data bit by bit.
- Prior to execution, a set of instructions forming a program is placed in the EDVAC main memory. The instructions were then transferred one a time from the main memory to the CPU for execution.
- Each instruction had a well defined structures of the form :
  A_1 A_2 A_3 A_4 op
- The meaning : perform the operation op (addition, subtraction, multiplication, etc.) on the contents of main memory locations A_1 and A_2 and then place the result in main memory location A_3. A_4 specifies the address of the next instruction to be executed.

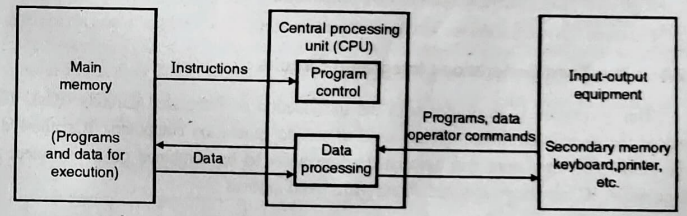


Fig. 1.10 : Organization of a fist generation computer

The trends, which were encountered during the era of first generation computers.

- Centralised control in a single CPU; all the operations required a direct intervention of the CPU.
- Use of ferrite-core main memory had started.
- Concept of virtual memory had started.
- Punched cards were used as input.

- Magnetic tapes and magnetic drums were used as secondary memory.
- Binary code or machine language was used for programming.
- Towards the end due to difficulties encountered in use of machine language as programming language, the use of symbolic language that is now called assembly language started.
- Assembler, a program that translates assembly language programs to machine language was made.
- Computer was accessible to only one programmer (single user mode).
- Advent of Von Neumann architecture.

1.2.3 The Second Generation : Transistors :

The use of the transistors defines the second generation of computers. Vacuum tubes were replaced by transistors. The transistor is smaller, cheaper and dissipates less heat than a vacuum tube. The second generation characterizes greater speed, larger memory capacity and the smaller size over first generation. Computer hardware and software evolved rapidly after introduction of second generation computer.

- Complex instructions were added to the set of instructions.
- More registers were added to the CPU to facilitate data and address manipulation.
- Floating point number was introduced to support scientific application.
- Input/output operations were added for easy transfer of information to and from peripheral devices like printer and secondary memory.
- High level programming languages were introduced.
- Provision of system software with the computer.

1.2.4 The Third Generation : Integrated Circuits :

This generation is associated with the introduction of Integrated Circuits (ICs). This replaced the discrete electronic circuits used in second generation computers. ICs allowed a large number of transistors and associated components to be combined on a tiny piece of silicon wafer. IC technology initiated a long terms trend towards :

(1) Higher speed
(2) Smaller size
(3) Lower hardware cost
(4) Lower power consumption
(5) More reliable circuit

IBM developed the most influtial third generation computer, the system/360. The machine became a standard for all main frame computers.

- It was based on Von Neumann architecture.
- It had about 200 distinct instruction types.
- It had many addressing modes.
- It supported various data types.
- It supported both fixed point and floating point numbers.
- It had 16 general purpose registers.
- The CPU had two major control states :
  (1) Supervisory state for use by the operating system.
  (2) User state for executing application programs.

1.2.5 Later Generations : VLSI era :

VLSI allowed manufactures to fabricate a CPU, main memory or even all the electronic circuit of a computer on a single IC that can be mass produced at a very low cost. This resulted in a new class of machines ranging from portable personal computers to supercomputers that contain thousands of CPUs. Two most important impact of VLSI are :

(1) Semiconductor memory
(2) Microprocessors

1.3 Interconnection Structures :

May 2007

To form a working computer, individual components must be connected in an organized way. There are many ways of doing it. The collection of paths connecting components is called the interconnection structure. The design of this structure depends on the exchanges that must be made between modules. When a word of data is transferred between modules, all its bits are transferred in parallel. These bits are transferred simultaneously over different wires. A group of wires that connects several devices is called a Bus. In addition to the wires that carry data, the computer must have some lines for addressing and control purposes.

1.3.1 Single-bus Structure :

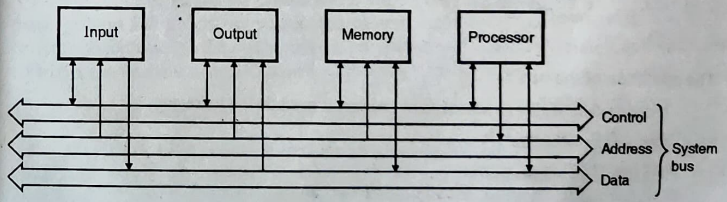


Fig. 1.11 : A single bus structure

The simplest way to interconnect functional units of a computer is to use a single system bus. System bus consists of :

(i) Data bus

(ii) Address bus

(iii) Control bus

This bus is time shared. Because the bus can be used for only one transfer at a time, only two units can communicate at any given instant.

Data bus :

- The data lines provide a path for moving data between system modules. These lines, collectively are called the 'Data bus'.
- The data lines are bi-directional, so that the data can be sent or received by the processor.

Address bus :

- Every device connected to bus has an address. A memory unit is given a block of addresses, depending on number of words in it. For example, if the CPU wishes to read a word of data from memory, it puts the address of the desired word on the address lines.
- The address lines are always unidirectional i.e. the address is transmitted by the processor to different modules.

Control bus :

- The control lines are used to control the various units like memory and I/O, Processor uses control signals to control various modules.
- Control signals transmit both command and timing information. Control signals specify operation to be performed. Typical control signals include :
  - Memory read
  - Memory write
  - I/O read
  - I/O write
  - Bus request
  - Bus grant

The operation of the bus :

If one module wishes to send data to another, it must do the followings :

(1) Obtain control of bus

(2) Transfer data via the bus

Similarly, if one module wishes to request data from another module, it must do the followings :

(1) Obtain control of bus

(2) Makes a request to the other module over the appropriate control lines and address lines.

(3) It must then wait for the requested module to send data.

1.3.2 Multiple-Bus Hierarchies :

If a greater number of devices are connected to the bus, performance will suffer due to following reasons :

- In general, the more devices attached to the bus, the greater will be the propagation delay.
- The bus may become a bottleneck as the aggregate data transfer demand approaches the capacity of the bus. This problem can be countered to some extent by increasing the data rate that the bus can carry and by using wider buses.

Most computer systems enjoy the use of multiple buses. These buses are arranged in a hierarchy.

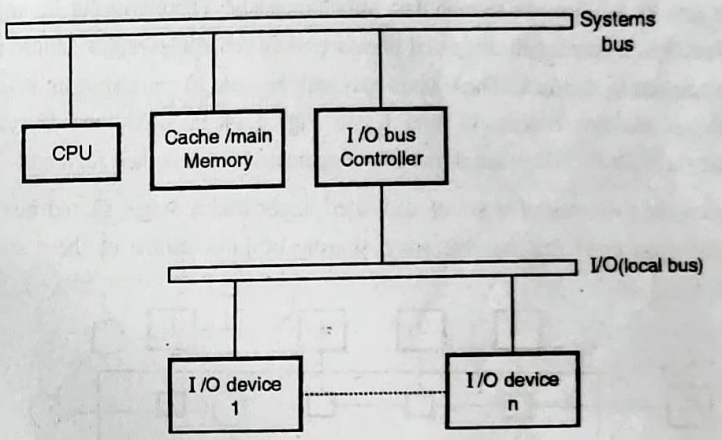


Fig. 1.12 : A multiple bus structure

The principle use of the system bus is high-speed data transfer between the CPU and memory. Most I/O devices are slower than memory and they are put on the local bus. These devices are connected to the system bus via interface circuit called I/O controller. A single I/O controller can interface many I/O devices to the system bus.

1.3.3 Other Interconnection Structures :

A system's interconnection structure can be defined by a graph whose nodes denote components such as computers, memories, I/O controller etc. Edges between these components is known as communication path or buses.

- A path designed to link only two devices is said to be dedicated.

- A path used to transfer information between different sets of devices at different times is said to be shared or multiplexed.

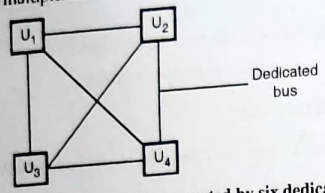


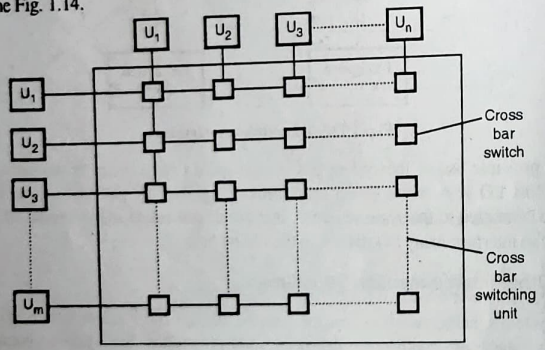
Fig. 1.13 : System of four units connected by six dedicated buses

A conceptually simple interconnection method is shown in the Fig. 1.13. There is a dedicated bus between all pairs of components that need to communicate. The general case in which n units must be connected in possible ways need.

$\frac{n \times (n-1)}{2}$ dedicated buses

All n devices can send or receive data simultaneously. There will not be any delay due to busy connection. System with dedicated lines are more reliable as a link failure effects only two units connected to that link. These units may still be able to communicate via other units. For example, if the bus linking U_1 and U_4 in Fig. 1.14 fails, U_1 and U_4 can possibly communicate via U_2 or U_3. The main drawback of dedicated buses is their high cost.

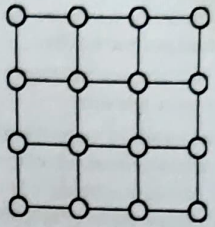
Between the extremes of a set of dedicated buses and a single shared bus lie various interconnection structures that involve some sharing of links. Some of these structures are shown in the Fig. 1.14.



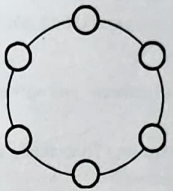
(a) Crossbar connection of two groups of units

Fig. 1.14

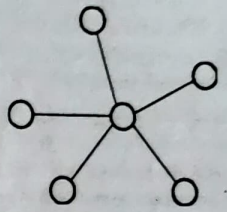
(b) Linear network



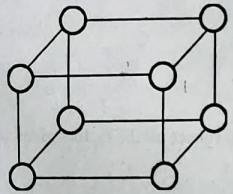
(c) Mesh network



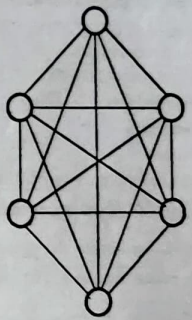
(d) Ring network



(e) Star network



(f) Hyper cube network



(g) A complete network

Fig. 1.15

1.4 Performance of a Computer :

The total time needed to execute application program is the most important measure of computer performance. Because these programs are written in a high level language, the performance of the program also depends on the efficiency of the compiler :

- Performance of a program is effected by the way in which the compiler translates programs into machine code.
- The ideal program performance demands a perfect match between machine capability and program behaviors.
- Machine capability can be enhanced with :
  (i) Better hardware technology
  (ii) Innovative architectural features
  (iii) Efficient resource management
- A program can be made efficient :
  (i) With better algorithm
  (ii) With better data structure
  (iii) Language efficiency
  (iv) Compiler technology
- It is impossible to achieve a perfect match between hardware and software by merely improving only a few factors without touching others.
- Machine performance may vary from program to program. To evaluate a machine, the machine should be evaluated by running a mix of programs.

Few fundamental factors have been suggested for projecting the performance of a computer. The simplest measure of program performance is its turn around time, which includes :

(i) Disk and memory accesses
(ii) Input and output activities
(iii) Compilation time
(iv) OS overhead
(v) CPU time

In order to reduce the turn around time, one must reduce all the factors given above.

Clock rate and CPI (Cycles per instruction) :

The CPU (or simply the processor) of a computer is driven by a clock with a constant cycle time (τ in nanoseconds). The inverse of the cycle time is the clock rate (f)

$$f = \frac{1}{\tau}$$

The size of a program is given in terms of instruction count (I_c). Different machine instructions may require different number of clock cycles to execute. Therefore, average cycles per instruction (CPI) becomes an important parameter for measuring the time needed to execute a program on a given machine.

CPI depends both on the machine and the program.

If I_c be the number of instructions in a given program, or the instruction count. The CPU time (T in seconds/program) needed to execute the program is estimated by :

$$T = I_c \times CPI \times \tau$$

MIPS rate :

The processor speed is often measured in terms of million of instructions per second (MIPS). This is called the MIPS rate of a given processor. MIPS rate varies with respect to a number of factors :

(1) Clock rate (f)
(2) Instruction count (I_c)
(3) CPI

If a program having I_c number of instruction requires T seconds of CPU time then,

$\therefore$ Time required (average) to execute 1 instruction

$$= \frac{T}{I_c}$$

$\therefore$ Time required to execute 1 million (10^6) instructions

$$= \frac{T \times 10^6}{I_c}$$

$$\therefore \text{ MIPS rate} = \frac{I_c}{T \times 10^6} \quad [\text{as } T = I_c \times CPI \times \tau]$$

$$\text{MIPS rate} = \frac{I_c}{I_c \times CPI \times \tau \times 10^6} = \frac{f}{CPI \times 10^6} \quad [f = 1/\tau]$$

Throughput rate :

Another important concept is related to how many programs a system can execute per unit time, called the system throughput ω_p. ω_p is defined by :

$$\omega_p = \frac{f}{I_c \times CPI}$$

where I_c is the average program length.

Example : A 40 MHz processor was used to execute a benchmark program with the following instruction mix and clock cycle counts :

Instruction type	Instruction count	Clock cycle count
Integer arithmetic	45,000	1
Data transfer	32,000	2
Floating point	15,000	2
Control transfer	8,000	2

Determine the effective CPI, MIPS rate and execution time for this program.

Solution :

$$\text{Average CPI} = \frac{45000 \times 1 + 32000 \times 2 + 15000 \times 2 + 8000 \times 2}{45000 + 32000 + 15000 + 8000}$$

$$= \frac{45 + 64 + 30 + 16}{45 + 32 + 15 + 8}$$

$$= \frac{155}{100} = 1.55$$

$$\text{Frequency f} = 40 \times 10^6$$

$$\text{Number of program instruction executed per second} = \frac{40 \times 10^6}{1.55}$$

$$= 25.8 \times 10^6 \text{ instructions/second}$$

$$\therefore \text{ MIPS rating} = 25.8$$

$$\text{Execution time T} = (45 + 32 + 15 + 8) \times 10^3 \times 1.55 \times \frac{1}{40 \times 10^6}$$

$$= 0.003875 \text{ seconds}$$

1.5 University Questions and Answers :

Dec. 2005 : Total Marks 06

Q. 1 Draw and explain the Von Neumann architecture. **(Section 1.1)** **(6 Marks)**

May 2007 : Total Marks 08

Q. 2 Explain Von Neumann architecture with diagram. **(Section 1.1)** **(5 Marks)**

Q. 3 Write a note on interconnection network. **(Section 1.3)** **(3 Marks)**

❑❑❑