Unit 2 Basics of Computers

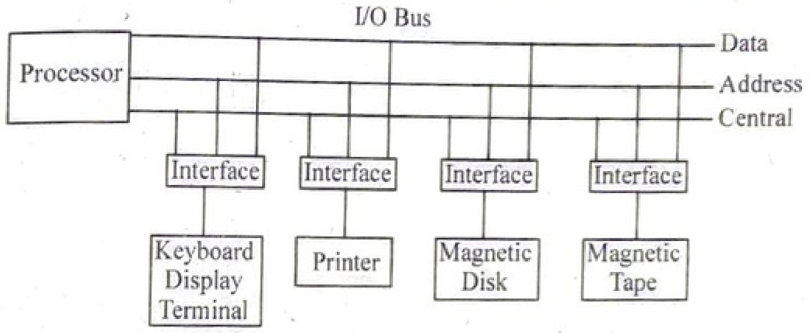
2.1. Input Unit

It accepts instructions and data from outside world.

It converts these instructions and data in computer acceptable from.

It supplies the converted instructions and data to computer system for further processing.

Input Output Interface



Input output interface provides a method for transferring into^4 b/w internal storage and external I/O devices peripherals connected to a computer need special communication links for interfacing them with the central processing unit the purpose of the communication link is to resolve the difference that exist b/w the central computer and each peripheral.

Major differences are :

1. Peripherals are electromechanical and electromagnetic devices and their manner of operation is different from the operation of the CPU and memory which are electronic devices.
2. The data transfer rate of peripherals is usually slower than the transfer rate of the CPU and consequently a synchronization mechanism may be needed.
3. Data codes and formats in peripherals differ from the word format in the CPU and memory.
4. The operating nodes of peripherals are different from each other and each must be controlled so as not to disturb the operation of other peripherals connected to the CPU.

To resolve these differences computer systems include special h/w compnents b/w the CPU and peripherals to supervise and synchronize all input and output transfers. These components are called interface units because they interface b/w the processor bus and the peripheral, devices.

In addition, each device may have its own controller that supervises the operations of the particular mechanism in the peripheral.

I/O Bus and Interface modules

A typical cain m link b/w the processor and several peripherals. The I/O bus consists of data lines, address lines and control lines. The magnetic disc, printer and terminal and employed in practically any general purpose computer.

The magnetic tape is use a in source computer for backup storage. Each peripheral device has associated with it an interface unit. Each interface decodes the address and control received from the I/O bus, interprets them for the peripherals and provide signals for the peripheral controller. It also synchronizes the data flow and supervises the transfer b/w peripheral of and processor. Each peripheral has its own controller that operates the particular electromechanical device.

The I/O bus from the processor is attached to all peripheral interfaces. To communicate with a particular device, the processor places a device address on the address lines. Each interface attached to the I/O bus contains an address decoder that monitors the address line. When the interface detects its own address it activates the path b/w the bus lines and the device that it controls.

2.2 OUTPUT UNIT :

It accepts the result produced by a computer which are in coded form and hence, we can't easily understand them.

It converts these coded result to human acceptable (readable) form.

It supplies the converted result to outside world.

I/O : OUTPUT DEVICES

Any device that provides us like output which is transformed from input is known as output device because most information from a computer is provided to the user as a required output is either visual form or audio form therefore, the most commonly used output devices are :

Monitor and speaker for output in text format an a paper a printer is used. While monitors and speakers are the most common output devices there are many other devices too.

Some examples are :

1. Head phone
2. Projector
3. Lighting control system.
4. Audio recording system etc.

Output is what we interact with.

Monitor

1. Monitor is also caused visual display unit. It is a piece of electrical equipment which display image generated from video output of devices such as computer without creating a permanent record.

2. Most of the monitors now a days considered of LCD technology where as older monitor work based on **CRT** (Cathod Ray Technology

3. VDU are used to visually interface with the computer and are similar is appearance to a television VDU image are made up of small blocks of colours light caused pixels. The resolution of the screen improves as the no. of pixels is increased.

COMPARISON BETWEEN CRT AND LCD

- CRT has high speed response.
- CRT never lag in taking the input.
- CRT is a reliable proven display technology.

DISADVANTAGES

- CRT has large size and weight.
- Older CRT's are prone to burn is.
- When used under lower refresh rates it causes Hickring.
- Greater power consumption than similarly sized LCD

LCD (ADVANTAGE)

- Very compact and light Weighed.
- Low power consumption.
- Little or on Hickering

LCD DISADVANTAGE

- It has limited viewing angle which causes colour, contrast and brightness to very.
- LCD are some what more expensive than CRT's.
- On put lag's.
- It has slow response time as compared to CRT screen.

Output Devices

Monitor

Printer.

Printer : A printer is a device that print text or images on paper is a computer system a printer is a output device which produces a hard copy which is readable by human is in the form of text or images. The hard copy is generated by the document which is stored in electronic form. There are different types of printers having various technologies used.

Type of Printer

1. Daisy wheel : it is similar to a type writer having a ball at the specific side. This type of printer has a plastic or metals wheel on which the shape of each character stands out a hammer pushes the wheel against the ribbon which inturn make an ink stain in the shape of character.

2. Dot-Matrix Printer : It creates characters by striking pins against an ink ribbon and each pin makes a dot and the combination of dots form characters.

3. Ink-Jet Printer : These printers sprays ink at a sheet of paper the output produce by this printer is high in quality.

4. Laser-Printers : Laser Printer uses the same technology as copy machine it stores the images of document is its memory and an produce a number of printouts without any interruption.

5. Line-Printers : It contains a chain of characters or pins that print an entire line at one time these printers are very fast in working but produce low quality output.

6. Thermal Printers : It is an expensive printer that work by pushing heated pins against the heat sensitive paper and produce text on them such printers are widely used in fax machine.

Plotter : A plotter is a vector graphics printing devices which is used to print graphical plots that connects to a computer it is a device that draws picture on the paper by receiving commands from computer it is different from printer because. It draws line using a pen.

Type of Plotter

- Pen plotter
- Electrostatic Plotter

Pen Plotter : Pen plotter have an ink pen attached to draw the images. Pen plotters print by moving open across the surface of a paper. Which means that plotters are restricted to line art rather than graphics. unlike printers it can draw line art including text but it perform very slowly because of the mechanical movement of the pen. It is capable of drawing solid regions but cannot fill colours in the place of colour it uses no. of closed lines to fill an are.

When computer memory was very expensive and the processor was having a limited power this was the often fastest way to produce vector based art work.

Flat Bed Plotter : This is a plotter where the paper is fixed an a flat surface and pens are moved to draw the image this plotter can use various different colour pens for drawing in the plotter the size of the plot is limited only by the size of the plotter's bed.

Drum Plotter : In this plotter the pen is moved in a single axis track and the paper it self moves on a cylindrical drum to add the other dimension. The size of the graph is limited by the width of the drum and can be of any length.

Electrostatic Plotter : An electrostatic plotter produces image by charging the paper with a high voltage. This voltage attracts toner which is then melted in to the paper with heat. This type of plotter is fast but the quality is generally considered to be poor when compared to pen plotter.

Voice output : Voice output are the standard jacks used to connect audio component to each other most mother board require a separate sound card in order to get audio output.

Storage Devices

1. Primary
2. Secondary

Storage Devices : Computer Data storage is often called storage or memory refers to computer components, devices & recording media that store digital data used for computing for some interval of time. The basic purchase for storage device is information retention. That can be utilise further. Such devices are fundamental components of mordern computer system for system efficiency. The storage devices are classified into secondary & primary category the secondary storage is use to store information that is not currently used by the system and will be useful in future.

Secondary memory is slower in speed because of large amount of data storage & serial excess

on the other hand primary storage capacity but faster in speed. It has nature of volatility unlike secondary storage.

Removable Storage Media

The removable storage media are the types of storage devices that are used for storage of data the main reason for using removable storage is portability.

In the case of portability the removable storage devices comes first such devices and be C.D, pendrives, floppy drive etc.

It is also useful in cases where some situations are like data transpher it is not possible through networking because. Security regions in that case storage media of removable nature solves the problem.

Non-Removable Storage Media

Non-removable storage media is the storage components that cannot be detached from the computer

e.g.- RAM, but RAM is a volatile memory which cannot be used to store data permanently but non-removable storage devices need to be non volatile and having higher storage capacity.

Sequential Access assumes that records can be processed only sequentially on the other hand. One more accessing concept is there that is direct or random access in random access data need not to be process requently a random access is performed to read or write data on such storage. Some devices like magnetic tape and hard disks are example of sequential access but on hard-disk can be accessed randomly also by the use of cash concept in which indexing is prepared for the entire stored data so, that the index can provide direct address of the required data and there is no need to be sequentially reached.

There are a number of devices that can be sequentially accessed like floppy disk a C.D, and a hard-disk

Floppy disk is further categarised into several forms on the bases of :

Size & Storage capacity

1. Generally a floppy-size 3.5 inches in having a capacity 1.44 MB. data stor
2. The floppy of size 5.25 inches have storage capacity of 1.2 MB. and flexible in nature.
3. The floppy of size 8 inches is having capacity of 100-500 KB data storage.

2.3 Memory

The memory unit is an essential component in any digital computer since it is needed for storing programs and data. We can divide the computer memory into two parts. The memory unit that communicate directly with the CPU is called the main memory. Devices that provide backup storage are called secondary memory.

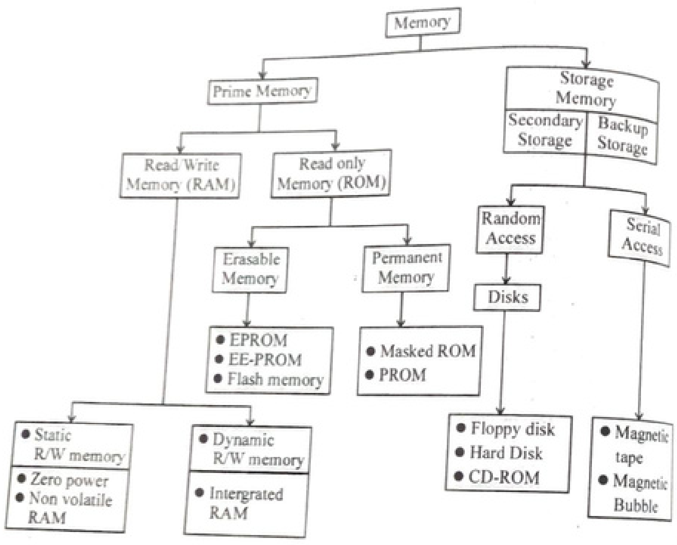


Fig. 1

(1) RAM - Random Access Memory

(2) ROM - Read Only Memory.

(3) Cache - A fast storage type of memory.

Before we confuse you more, let us give you a table which will list these types and more and the features of each and every:

Memory Type	**Features and usefulness**
ROM	Read only Memory used for BIOS chips, CMOS chips, and Special function Chips.
RAM	Random access memory - FPM RAM, EDO & BEDO RAM, Synchronous DRAM (SDRAM)
DRAM	Dynamic RAM-Actual memory chips on SIMM boards or Motherboard (Main Memory)-DRAM is actually FPM, EDO RAM, or SDRAM on a chip.
SRAM	Static RAM, used as External (L2) CACHE. L2 SRAM is on chips. L2 On-board cache is in the CPU chip. It comes in 3 basic types – Async SRAM, Sync SRAM, and PB SRAM (Pipelined Burst RAM - the fastest).
FPM	Fast page mode DRAM-Some modern computers today use Fast Page Mode DRAM. The difference between FPM DRAM and regular DRAM is in the way the memory is accessed by the controller. When data needed is in the same page or row that the previous data was found, The controller only has to indicate the next column location to access the data. By not having to generate a complete address the memory is accessed a little faster.
EDO RAM	Extended Data Out RAM Memory used on Pentium or later type motherboards. EDO RAM is not designed for 486 or earlier motherboards. EDO RAM is on 72-pin SIMMs. EDO RAM comes in plain EDO and Burst EDO (BEDO RAM) versions. EDO and BEDO RAM are ok in systems with bus speeds up to 66MHz.
SDRAM	Synchronous DRAM (Pentium w/MMX has SDRAM as main memory). SDRAM memory is on 168-pin DIMM chips. SDRAM comes in several types with speeds from 10, 15, 20, and 25 nanoseconds.
FLASH MEMORY	Normally, it is memory on a card. The size of a PCMCIA card.
Cache Memory	Normally, it is memory on a card. The size of a PCMCIA card.

So this is the categorization of the memory. Now we will study each and every memory one by one.

Main Memory

Main memory is the central storage unit in a computer. It is relatively large and fast memory used to stare programs and data during the computer operations.

Integrated circuit RAM chip

static

Cousist of internal flip flops that store the binary into⁴.

Stored into⁴ remains valid as long as power is applied to the unit

Dynamic

It stores into⁴ (binary) in the form of electric charges that are applied to the capacitors

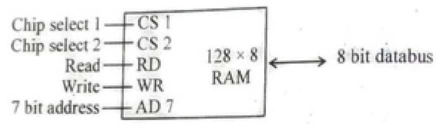
It offers reduced power cousumption and larger storage capacity in a simple memory chip

ROM is also used along with RAM.

It is basically used to store an initial program called a **bootstrap loader.**

The boot strap loader is a program whose functions is to start the computer software operating when power is turned on.

RAM



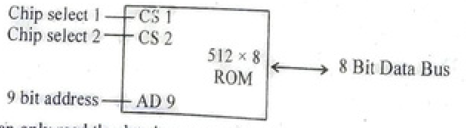
It requires 7 bit address and 8 bit data bus (Di-directional)

Read and write inputs specify the memory operations.

The two chips select control input are fore enabling the chip only when it is selected by the microprocessor.

CS 1	$\overline{CS}$ 2	RD	WR	Memory function	Data Bus
0	0	×	×	Inhibit	Hight In
0	1	×	×	Inhibit	4
1	0	0	0	Inhibit	4
1	0	0	1	Write	Input data to RAM
1	0	1	×	Read	Output from RAM

ROM



Since ROM can only read the data bus can only be in an output mode.

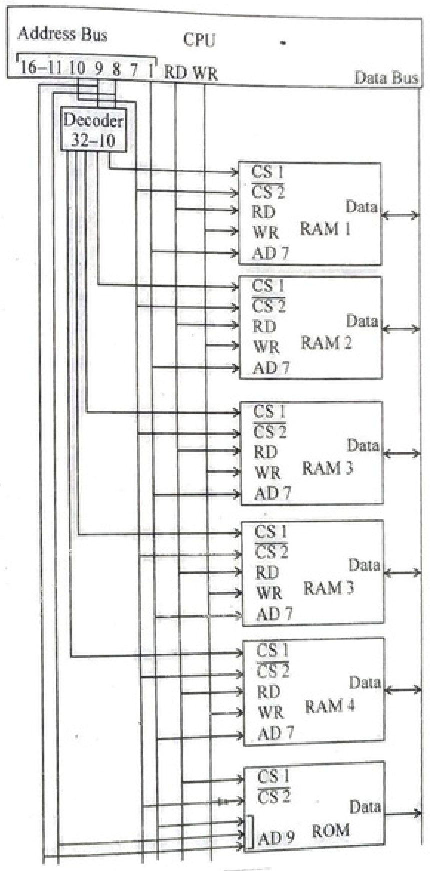
The two chip select inputs must be in CS1 = 1 and $\overline{CS}$ 2 = 0 for the unit to operate often wise the

data bus in a high impedance state.

Memory Connections to CPU

RAM and ROM chips are connected to a CPU through the data and address buses.

The low order lewis in the address bus select the byte within the chips and other lines in through its chip select inputs.



The particular RAM chip selected is determined from line 8 and 9 in the address bus. Th done through a 2 × 4 decoder where input go to the CS 1 inputs in each RAM chip.

When line 8 and 9 = $\underline{0\ 0}$ the first RAM chip is selected

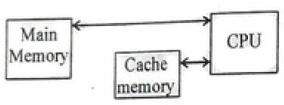
When line 8 and 9 = 01, the second RAM chip is selected

The RD and WR outputs from the microprocessor are applied to the inputs of each RAM c

Selection b/w RAM & ROM is achieved through **bus line 10.**

When bus line 10 has **bit 0 RAM is selected, ROM** when this line **has 1.**

Cache Memory



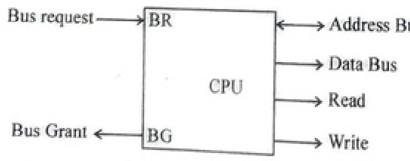
If the active portions of the program and data are placed in a fast small memory the avera memory access time can be reduced fluid reducing the total evections tunnel of program. Such a fa small memory is referred to as a cache memory.

The fundamental idea of cache organization is that by keeping the most frequently access instructions and data in the fast cache memory the average memory access time will approach t access time of the cache.

Direct Memory Access (DMA)

The transfer of data between a fast storage device such as magnetic disk and memory is ofte limited by the speed of the CPU. Removing CPU from the path and letting the peripheral devic manage the memory buses directly would improve the speed of transfer. This transfer technique i called **direct memory access (DMA).** During DMA transfer, the CPU is idle and has no control of the memory buses. ADMA controller takes over the buses to manage the transfer directly b/w the I O device and memory.

(CPU Bus Signal for DMA Transfer)



CPU BUS Signals for DMA Transfer

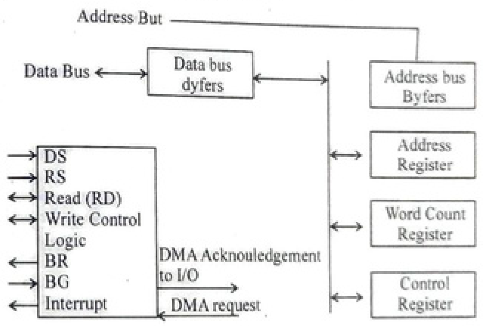
The Bus request input is used by the DMA controller to request the CPU to relinquish control of the Buses. When this input is active the CPU terminates the execution of the current instruction and

places the address bus the data bus and the read and write lines into a high impedance state.

CPU activates the bus grant output to inform the external DMA that the buses are in high impedance state. The DMA that originates the bus request can now take control of the buses to conduct memory transfers without processor intervention. When the DMA terminates the transfer, it disables the bus request line the CPU disables the bus grant, takes control of the buses and returns to its normal operations.

When the DMA takes control of the bus system, it communicates directly with the memory. The transfer can be made in several ways. In DMA bust transfer a black sequence consisting of number of memory words is transferred in a continuous first while the DMA controller is master of the memory buses. This mode of transfer is needed for first devices such as magnetic disks, where data transmission can't be stopped or slowed down unit an entire black is transferred. An Alternative technique called cycle stealing allows the DMA controller to transfer one data word at a time after which it must return control of the buses to the CPU. The CPU merely delays its operations for are memory cycle to allow the direct memory I/O transfer to steal one memory cycle.

DMA Controller



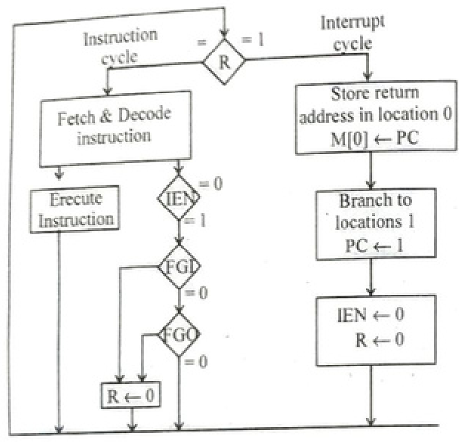
The DMA Unit Communicates with the CPU via the Data Bus & Control lines. The registers in the DMA are selected by the CPU through the address bus by enabling the DS and RS inputs. RD & WR are bidirectional. When BG is 0, the CPU can communicate with DMA Register. But When BG is 1 the CPU has relinquished the buses and the DMA can communicate directly with the memory by specifying an address in the address bus and activating the RD or WR control. DMA communicates with external peripherals thought the request and acknowledge lines.

DMA controller has three registers

Address register - it contains and address to specify the desired location in memory.

Word Count register - It holds the be transferred.

Control register specifies the mode of transfer.

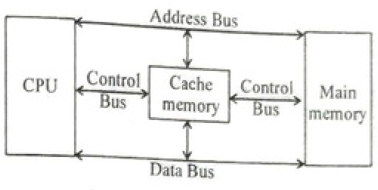

Instruction cycle
= 0
R
= 1
Interrupt cycle
Fetch & Decode instruction
Store return address in location 0 M[0] ← PC
Erecute Instruction
IEN
= 0
= 1
Branch to locations 1 PC ← 1
FGI
= 0
IEN ← 0 R ← 0
FGO
= 0
R ← 0

R = an interrupt flip flop
IEN

Cache Memory

All processes are completed in CPU and all data which is currently stored in main memory but the CPU is usually faster than main memory access with the result that processing speed is limited primarily by the speed of main memory. Then computer work very slowly because process is not provide to the CPU and CPU doesn't process on any data at that mid time when main memory transfers data to CPU is very slow of speed.

Cache memory is an alternative of the low speed of main memory. It is a technique used to recover for the mismatch in operating speeds is to employ an externally fast small cache b/w the CPU and main memory where access time is closed to processor logic clock cycle time.



Cache memory is a very special memory.

Cache memory is small in capacity but speed is almost equal to CPU.

It is used to increase the speed of processing by making current programs and data will able to provide CPU at a rapid rate.

The performance of cache memory is frequently measured in terms of a quantity called **hit ratio.**

$$\text{Hit Ratio} = \frac{\text{Number of hits}}{\text{Total Number of Address Re flued}}$$

When data is transferred from main memory to cache memory is called **mapping.**

Three types of mapping procedures are considered in the organisation of cache memory

Associative mapping

It is simplest & fastest type of mapping.

It work on search method

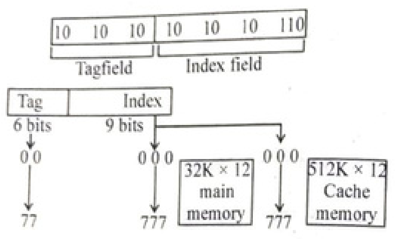
CPU Address (15 bits)

Address	Data
–	–
–	–
–	–
	–

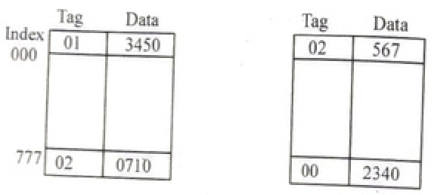
In associative memory both address and data stores in itself. Data can store in associative cache memory in any location and any word from main memory.

Direct Mapping

Associative memories are very expensive so we can use another type of cache organisation. In this technique CPU address of 15 bits is divided into two fields.



Set Associative Mapping



Modes of Transfer

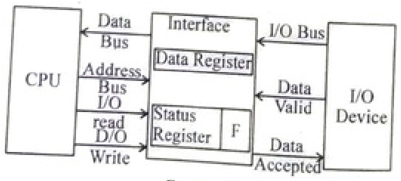
Binary into[4] received from an external device is usually stored in memory for later processing into[4] transferred from the central computer into an exotically device alginates in the memory unit.

Data transfer b/w the central computer and I/O devices may be handled in a variety of modes.

Programmed I/O

Interrupt initiated I/O

Direct memory Access

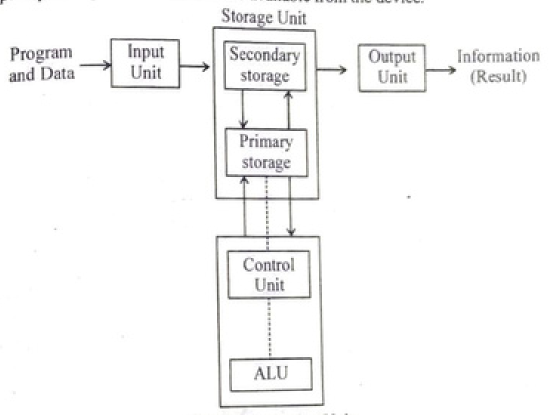


F = Flagbit

Operations are the result of I/O instructions written in the computer program. Each data item transfer is initiated by an instruction in the program. Usually the transfer is to and from a CPU register and peripheral. Other instructions and needed to transfer the data to and from CPU & memory.

Transferring data under program control squares constant monitoring of the peripheral by the CPU. Once a data transfer is initiated the CPU is required to monitor the interface to see when a transfer can again be made.

In programmed I/O method, the CPU stays in a program loop until the I/O unit indicates that it is ready for data transfer. This is a time consuming process since it keeps the process on busy needlessly. It can be avoided by using an interrupt facility and special commands to inform the interface to issue an interrupt request signal when the data are available from the device.

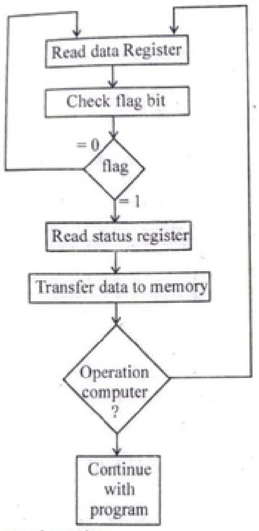


Centrol Processing Unit

The device transfers bytes of data one at a time a they are available when a byte of data is available, the device places it in the I/O bus and enables its data valid line. The interface accepts the byte into its data register and enables the data accepted line. The interface set a bit in the status register that is referred of "F". The device can now disable the data valid line, but it will not transfer another byte until the data accepted line is disabled by the interface.

Interrupt Initiated I/O

An alternative to the CPU constantly monitoring the flag is to let the interface inform the computer when it is ready to transfer data. This mode of transfer uses the interrupt facility. White the CPU is running a program it doesn't check the flog. However, when the flag is set the computer is momentarily interrupted from proceeding with the current program and is informed of the fact that the flag has been set. The CPU deviates from what it is doing to take care of the input on output transfer. After the transfer is completed, the computer returns to the previous program to continue what is was doing before the interrupt.



Flow chart for CPU program to input data

2.4 Arithmetic Logic Unit and its Components

It is the place where actual execution of an instruction takes place during processing operation. Calculations are performed and all comparisons (decisions) are made in the ALU.

However, almost all ALUs are designed to perform the four basic arithmetic operations (add, subtract, multiply and divide) and logic operations or comparison such as, less than, equal to, and greater than.

Control Unit

Manages and coordinates the entire system. It obtains instructions from the program stored in main memory, interprets instructions, and issues signals causing other units of the system to execute them.

Number System

Binary Number System

Conversion from Another Base of Decimal

Eg. 1. $11001_2 = ?_{10}$

Step 1 : Determine column number

Step 2 : Multiply the obtained column value by the digits in cores pending column.

Step 3 : Sum up the products calculated in step 2.

Step 1.	**Step 2.**	**Step 3.**
$2^0 = 1$	$1 \times 1 = 1$	$1 + 0 + 0 + 8 + 16$
$2^1 = 2$	$2 \times 0 = 0$	$= 25_{10}$ **Ans.**
$2^2 = 4$	$4 \times 0 = 0$	
$2^3 = 8$	$8 \times 1 = 8$	
$2^4 = 16$	$16 \times 1 = 16$	

Eg. 2. 47068 = $?_{10}$

$4 \times 8^3 = 2048$
$7 \times 8^2 = 448$
$0 \times 8^1 = 0$
$6 \times 8^0 = 6$

$2048 + 448 + 0 + 6 = 2502_{10}$ **Ans.**

Eg. 3. $1AC_{16} = ?_{10}$

$1 \times 16^2 = 1 \times 256 = 256$
$A \times 16^1 = 10 \times 16 = 160$
$C \times 16^0 = 12 \times 1 = 12$
$= 428_{10}$ **Ans.**

Eg. 4. $4052_7 = ?_{10}$

$4 \times 7^3 = 343 \times 4 = 1372$
$0 \times 7^2 = 0$
$5 \times 7^1 = 35$
$2 \times 7^0 = 2$
$= 1409_{10}$

Eg. 5. $1AC_{13} = ?_{10}$

$1 \times 13^2 = 169$
$(10)\, A \times 13^1 = 130$
$(12)\, C \times 13^0 = 12$
$= 311_{10}$ **Ans.**

CONVERSION FROM DECIMAL TO ANOTHER BASE (Division Remainder)

Eg. 1. $25_{10} = ?_2$

$$\frac{25}{2} = 12 \text{ remainder } 1$$

$$\frac{12}{2} = 6 \text{ remainder } 0$$

$$\frac{6}{2} = 3 \text{ remainder } 0$$

$$\frac{3}{2} = 1 \text{ remainder } 1$$

$$\frac{1}{2} = 0 \text{ remainder } 1$$

$\therefore$ $\boxed{25_{10} = 11001_2}$

Eg. 2. $4210 = ?_2$

```
 2|42
 2|21 0
 2|10 1
 2| 5 0
 2| 2 1
  | 1 0
```

$42_{10} = 101010_2$

Eg. 3. $952_{10} = ?_8$

```
8|952
 |119 0
 | 14 7
 |  1 0
```

$952_{10} = 1670_8$

Eg. 4. $428_{10} = ?_{16}$

```
16|428
  | 26 12
  |  1 10
```

$428_{10} = 1AC_{16}$

Converting From Base other than 10 to Base other than 10

Step 1. Convert the original number to base 10 number.

Step 2. Convert from base 10 to required base.

Eg. 1. $545_6 = ?_4$

Firstly convert 545_6 to base 10

$$5 \times 6^2 = 180$$
$$4 \times 6^1 = 24$$
$$5 \times 6^0 = 5$$
$$= 209_{10}$$
$$545_6 = 209_{10}$$

Now convert to base 4.

$$\frac{209}{4} = 52 \text{ remainder } 1$$

$$\frac{52}{4} = 13 \text{ remainder } 0$$

$$\frac{13}{4} = 3 \text{ remainder } 1$$

$$\frac{3}{4} = 0 \text{ remainder } 3$$

$\boxed{\therefore 545_6 = 3101_4}$

Eg. 2. $101110_2 = ?_8$

Convert into base_{10}

$$1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$$
$$= 32 + 0 + 8 + 4 + 2 + 0$$
$$= 46_{10}$$

$46_{10} = ?_8$

$$\frac{46}{8} = 5 \text{ remainder } 6$$

$$\frac{5}{8} = 0 \text{ remainder } 5$$

$\boxed{\therefore 101110_2 = 56_8}$

Shortcut Method for Binary to Octal

Eg. 1. $101110_2 = ?_8$

I. $\underline{1\ 0\ 1}\ \underline{1\ 1\ 0}$ [Divide binary digits into group of 3]

II. Convert each group into one digit of act at

$$101_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$
$$= 4 + 0 + 1$$
$$= 5_8$$
$$110_2 = 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$$
$$= 4 + 2 + 0$$
$$= 6_8$$

$\boxed{\therefore 101110_2 = 56_8}$

Shortcut for octal to Binary

Eg.1. $562_8 = ?_2$

$5_8 = 101_2$

$6_8 = 110_2$

$2_8 = 010_2$

Eg. 2. $6751_8 = ?_2$

$6_8 = 110$

$7_8 = 111$

$5_8 = 101$

$1_8 = 001$

$\therefore\ 562_8 = 101110010_2$ $\qquad$ $\therefore\ 6751_8 = 11011110100$

Shortcut for Binary to Hexadecimal

Eg. 1. $11010011_2 = ?_{16}$

I. Divide into groups of 4

$\underline{1\ 1\ 0\ 1}\ \underline{0\ 0\ 1\ 1}$

II. Convert each group of 4 digit to 1 Hexadecimal digit

$$\begin{aligned}1101_2 &= 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0\\ &= 8 + 4 \times 0 + 1\\ &= 13 \Rightarrow D_{16}\\ 0011 &= 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0\\ &= 0 + 0 + 2 + 1\\ &= 3_{16}\\ \therefore\quad 11010011_{16} &= D3_{16}\end{aligned}$$

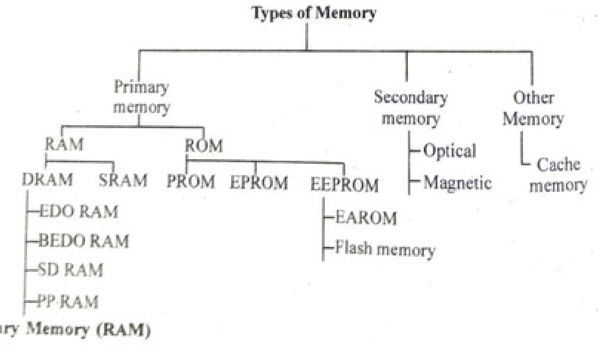
Shortcut for Hexadecimal to Binary

Eg. 1. $2AB_{16} = ?_2$

Convert decimal equivalent of each hexadecimal digit to 4 binary digit

$$\begin{aligned}2 &= 0010_2\\ A &= 10_{16} = 1010_2\\ B &= 11_{16} = 1011_2\\ \therefore\quad 2AB_{16} &= 001010101011_2\end{aligned}$$

Types of Memory



Primary Memory (RAM)

DRAM (Dynamic Random Access Memory) which are inexpensive. They are used essentially for the computer's main memory.

SRAM (Static Random Access Module) Which are fast and costly. SRAM memories are used in particular for the processor's cache memory.

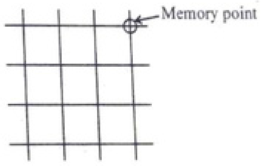
Operation of RAM

RAM comprises of hundreds of small capacitors that store loads. When loaded the logical state of the capacitor is equal to 1, otherwise 0, meaning that each capacitor represents one memory bit.

Given that the capacitor because discharge they must be consflarty recharged at regular intervals, know as the refresh cycles. DRAM memories for eg. require refresh cycles of around 15 nanoseconds.

Each capacitor is coupled wid a transistor (MDS type) enabling recovery or emendurent of the status of capacitor these transistors are arranged in the form of a table (matrix) thus we access a memory box also called memory point via line and column.

Each memory point is thus categorised by an address which corresponds to a row number and a column number this access is not instant and the access time period is known as **latency thine.** Consequently time required for access to data in the memory is equal to cycle time plus latency time.



Read only Memory

It is a class of storage media used in computers and other electronic devices because data stored in ROM can't be modified. It is mainly used to distribute firm ware (S/W that is very closely tied to specific H/W and unlikely to require frequent updates). In its stnic test sense ROM refers only to mask ROM, which is fabricated with the desired data permanently stored in it and thus can never be modified. However, were modern types such as EPROM and flash EPROM can be erased and reprogrammed multiple Tories they are fill described as ROM because the reprogramming process is generally infeftment comparatively slow and often design permit random access writes to individual memory locations.

Memory Module

Memory module is a broad term used to refer to series of dynamic random access memory integrated circuits modules mounted on a printed circuit board and designed for use in personal computers workstations and servers.

SIMM single In line memory module

DIMM Dual In line memory module

Ram bus memory modules are a subset of DIMM, but are usually referred to as R1MMs.

SO-D1MM, small outline D1MM a smaller version of the D1MM used in laptops.

SIMM

Single In-line memory module- It is a type of memory module containing random access memory used in computers from the early 1980s to the late 1990s.

It differs from a dual in line memory module (D1MM) the most predominant form of memory module today in that the contacts on a SIMM are redundant on both sides of the module.

A slender circuit board dedicated to storing memory chips. Each chip is capable of holding 8 to 9 chips per.

72 pins connector.

DIMM

DIMM is the type of circuit board that hold memory chips. DIMMs have 64-bit path because of the peritoneum processor requirements. Because of the new bit path, DIMMs can be installed one at a time unlike SIMMs which on a pinetum would require two be added.

Advantages of DIMM over SIMM DIMM have separate contacts on each side of the board, thereby providing twice as much data as a simple SIMM.

The command address and control signals are buffered on the DIMMs. With heavy memory requirements this will reduce the loading effort of the memory.

2.5 INSTRUCTIONS

An instruction format defines the layout of the bits of an instruction. It must include opcode, zero or more operands and addressing mode for each operand. The instruction length is usually kept in multiple of the character length, or memory transfer length which is usually 8-bits. With this length we will always get an integral number of instructions during a fetch cycle. Once the instruction length is fixed, it is necessary to allocate number of bits for opcode, operand(s) and addressing modes. For an instruction format of a given length, if more number of bits are allocated to opcode field then less number of bits available for addressing. The bits allocation for addressing can be determined by the following factors, which simplifies the task of allocating bits in the instruction.

- Number of addressing modes
- Number of operands
- Number of CPU registers
- Number of register sets
- Address range or number of address lines
- Address granularity (address can refer byte, word or double word)

The Fig. 2 shows the general IA-32 instruction format. The instruction represented by IA-32 format consists of four fields. Opcode field, addressing mode field, displacement field and immediate field. The Opcode field consists of one or two bytes. The addressing mode information is contained in one or two bytes immediately following the opcode. For instructions that involve the use of only one register in generating the effective

OP code	Addressing mode	Displacement	Immediate
1 or 2 bytes	1 or 2 bytes	1 or 4 bytes	1 or 4 bytes

Fig. 2 : 32 instruction format

addres of an operand, only one byte is needed in the addressing mode field. The addressing modes, base with index and base with index and displacement require two registers to generate the effective address of an operand. Hence addressing mode field for these two addressing mode is two bytes.

If a displacement value is used in computing an effective address for a memory operand, it is encoded into either one or four bytes in a field that immediately follows the addressing mode field.

If one of the operand is an immediate value, then it is placed in the last field of an instruction and it occupies either one or four bytes.

The instruction format of a processor also changes according to the address references in the instruction. Let us see various instruction types according to address references.

Three Address instructions:

The three adress instruction can be represented symbolically as

ADD A, B, C,

where A, B, C, are the variables. These variable names are assigned to distinct locations in the memory. In this instruction operands A and B are called source operands and C is called destination operand and ADD is the operation to be performed on the operands. Thus the general instruction format for three address instruction is

Operation Source 1, Source 2, Destination

The number of bits required to represent such instruction include :

1. Bits required to specify the three memory addresses of the three operands. If n-bits are required to specify one memory address, 3n bits are required to specify three memory addresses.
2. Bits required to specify the operation.

Two Address Instructions

The two address instruction can be represented symbolicaly as

ADD A, B

This instruction add the contents of variable A and B and stores the sum in variable destriying its previous contents. Here, operand A is source operand; however operand serves as both source and destination operand. The general instruction format for two address instruction is

Operation Source, Destination

To represent this instruction less number of bits are required as compared to three address instruction. The number of bits required to represent two address instruction include:

1. Bits required to specify the two memory addresses of the two operands, i.e. 2n bits.
2. Bits required to specify the operation.

One Address Instruction

The one address instruction can be represented symbolically as

ADD B

This instruction adds the contents of vairable A into the processor register called accumulator and stores the sum back into the accumulator destroying the previous contents of the accumulator. In this instruction the second operand is assumed implicitly in a unique location accumulator. The general instruction format for one address instruction is

Operation Source

Few more examples of one address instruction are :

LOAD A : This instruction copies the contents of memory location A into the accumulator.

STORE B : This instruction copies the contents of accumulator into memory location B.

In one address instruction, it is important to note that the operand specified in the instruction canbe eighter source operand or destination operand depending on the instruction. For example, in LOAD A instruction, the operand specified in the instruction is a source operand whereas the operand specified in the STORE B instruction is a destination operand. Similarly, in one address instruction the implied operand (accumulator) can be either source or destination depeding on the instruction.

Zero Address Instructions

In these instructions, the locations of all operands are defined implicity. Such instructions are found in machines that store operands in a structre called a pushdown stack.

From above discussion we can easily understand that the instruction with only one address will require less number of bits to represent it, and instruction with three addresses will require more number of bits to represent it. Therefore, to access entire instruction from the memory, the instruction iwth thre addresses requires more memory accesses while instruction with one address requires less memory accesses. The speed of instruction execution is mainly depend on how much memory accesses it requires for the execution. If memory accesses are more, more time is required to execute the instruction. Therefore, the execution time for three address instructions is more than the execution time for one address instructions.

To have a less excution time we have to use instructions with minimum memroy accesses. For this instead of referring the operands from memory it is advised to refer operands from processor registers. When machne level language programs are generated by compilers from high level languages, the intelligent compilers see that the maximum references to the operands lie in the processor registers.

Instruction and Execution Cycle

Every processor has some basic types of instructions such as data transfer instructions arithmetic instructions, logical instructions, branch instructions and so on. To perform a particular task on the computer. it is programmers job to select and write appropriate sequence. This job of programmer is known as **instruction sequencing.** The instructions written in a propoer sequence to execute a particular task is called **program.**

Processor executes a program with the help of program counter (PC). PC holds the address of the instruction to be executed next. To begin execution of a program, the address of its first instruction is placed into the PC. Then, the processor control circuits use the information (address of memory) in the PC to fetch execute instructions, one at a time, in the order of increasing addresses. This is called **straight line sequencing.** During the execution of instruction, the PC is incremented by the length of the current instruction in execution. For example, if currently executing instruction length is 3 bytes, the PC is incremented by 3 so that it points to the instruction to be executed next.

Branching

Everytime it is not possible to store a program in the consecutive memory locations. After execution of decision making instruction we have to follow one of the two program sequences. In such cases we can not use straight line sequencing. Here, we have to use branch instructions to transfer the program control from one straight line sequence to another straingth line sequence of instruction, as shown in following program.

For example, see the program given for operation |A – B|. In this program, we have to check whether A > B or B > A and accordingly we have to perform operation A – B or B – A.

```
        MOV NUM1, R0    : Get the number 1 into R0
        MOV NUM2, R1    : Get the number 2 into R1
        CMP R0,     R1  : NUM1 - NUM 2
        JB NEXT         : IF NUM1 < NUM2
                        : Jump to another program sequence
        SUB R0, R1      : NUM1 ← NUM1 – NUM2
        MOV R2, R1      : Store the result in R2
            ⋮
NEXT    SUB R1, R0      : NUM2 ← NUM2 – NUM1
        MOV R2, R0      : Store the result in R2
```

In the above program we have used JB NEXT instruction to transfer the program control to the instruction SUB R1, R0 if NUM1 is les than NUM2. Thus we have decided to branch the program control after checking the condition. Such branch instructions are called **conditional branch instructions.** In branch instructions the new address called **target address** or **branch target** is loaded into PC and instruction is fetched from the new address, instead of the instruction at the location that follows the branch instruction in sequential address order.

Unconditional Branch Instructions	
Mnemonic	**Instruction**
BR	Branch
JMP	Jump
CALL	Call
RET	Return
SKP	Skip

Table : (a) Unconditional branch Instructions

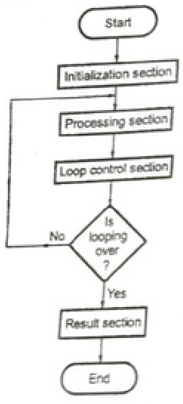
Conditional Branch Instructions		
Mnemonic	**Instruction**	**Tested Condition**
BR	Branch if Zero	Z = 1
BNZ	Branch if not zero	Z = 0
BC	Branch if carry	C = 1
BNC	Branch is no carry	C = 0
BP	Branch if Positive	S = 0
BN	Branch if Negative	S = 1
BV	Branch if overflow	V = 1
BNV	Branch if not overflow	V = 0
BH	Branch if higher	A > B
BL	Branch if Lower	A < B
BHE	Branch if higher or equal	A > B
BE	Branch in equal	A = B
BNE	Branch if not equal	A ≠ B

Table (b) Conditional branch instructions

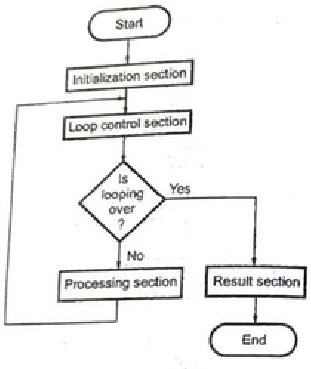
The branch instructions are also called **Jump** instructions. They may be conditional or unconditional. An unconditional branch instruction causes a branch to the specified address without any condition. The conditional branch instruction specifies a condition such as branch if positive or branch of zero. If the condition is met, the program counter is loaded with the branch address and the next instruction is taken from this address if condition is not met, the PC is incremented by the length of the current instruction in execution and the next instruction is taken from the next location in the sequence. The table gives listing of some unconditional branch instructions and conditional branch instructions.

Many times it is necessory to execute certain set of instructions repeatedly to perform a particular task number of times.

For example, to add ten numbers stored in the consecutive memory locations we have to perform addition ten times. A program style of executing certain set of instructions sepeatedly is called **program looping.**



Flowchart 1



Flowchart 2

The program loop is the basic structure which forces the processor to repeat a sequence of instructions. Loops have for sections.

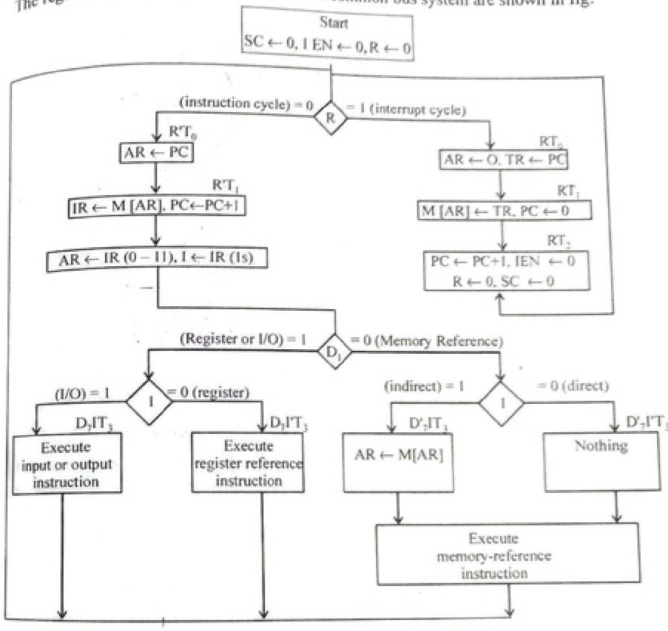
1. Initialization section. 2. Processing section.
3. Loop control section 4. Result section.

1. *The initialization section establishes the starting values of*
   - *loop counters for counting how many times loop is executed,*
   - *Address registers for indexing which give pointers to memory locations and*
   - *Other variables*
2. *The actual data manipulation occurs in the processing section. This is the section which does the work.*
3. *The lóop control section updates counters, indices (pointers) for the next iteration.*
4. *The result section analyzes and stores the results*

Note : *The processor executes intialization section and result section only once, while it may execute processing section and loop control section many times. Thus, the execution time of the loop will be mainly dependent on the execution time of the processing section and loop control section. The flowchart 1 shows typical program loop. The processing section in this flowchart is always executed at least once. If you interchange the position of the processing and loop control section then it is possible that the processing section may not be executed at all, if necessary.*

2.6 CONTROL OF REGISTERS

The registers of the computer connected to a common bus system are shown in fig.



The control inputs of the registers are LD (load), INR (increment) and CLR (clear). Suppose thta we want to derive the gate structure associated with the control inputs of AR. We scan table to find all the statement that change the content of AR.

$$R'T_0 : AR \leftarrow PC$$
$$R'T_2 : AR \leftarrow IR\ (0-11)$$
$$D_7'IT_3 : AR \leftarrow M\ [AR]$$
$$RT_0 : AR \leftarrow 0$$
$$D_5T_4 : AR \leftarrow AR + 1$$

The first three statement specify transfer of information from a register or memory to AR. The content of the source register or memory is placed on the bus and the content of the bus is transferred into AR by enabling its LD contron input. The fourth statement clears AR to 0. The host statement increment AR by 1. The control function can be combined into three Boolean expression as follows;

$$LD\ (AR) = R'\ T_0 + R'T_2\ + D_7'IT_3$$
$$CLR\ (AR) = RT_0$$
$$INR\ (AR) = D_5T_4$$

where LD(AR) is the load input of AR, CLR (AR) is the clear input of AR and INR (AR) in the increment input of AR. The control gate logic associated with AR shown in fig below.

In a similar fashion we can derive the control gates for the other registers as well as the logic needed to control the read and write inputs of memory. The logic gates associated with the read input of memory is derived by scanning table 5-6 to find the statement that specify a real operation. The read operation is recognized from the symbol $\leftarrow$ M[AR]

$Read = R'T_1 + D'7IT_3 + (D_0 + D_1 + D_2 + D_6)\ T_4$

The output of the logic gates that implement the Boolean expression above must be connected to the read input of memory.

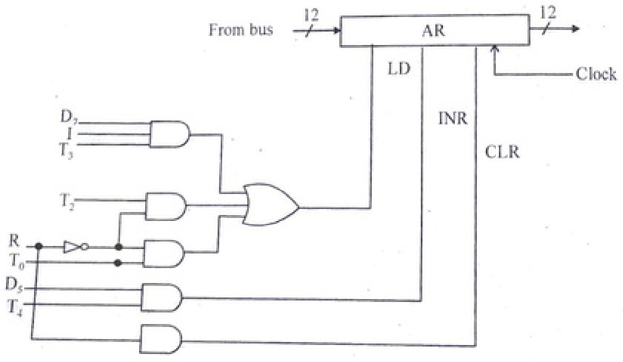


Fig. 3. Control gates associated with AR

2.7 Controlling of Arithmetic Operation

Unsigned Binary Numbers

When all the bits of binary number are used to represent the magnitude of the corresponding hexadecimal or decimal number, the number is called **unsigned binary** number. For example, the smallest 8-bit binary number is 0000 0000, and the largest eight bit binary number is 1111 1111. Since all bits represents magnitude of the number :

$0000\ 0000 \rightarrow (0)_{10}$ or $(00)_H$

to $1111\ 1111 \rightarrow (255)_{10}$ or $(FF)_H$

Therefore, an 8-bit unsigned binary number can represents any number from 0 to 225. The 16-bit unsigned binary number represents the total range from

$0000\ 0000\ 0000\ 0000 \rightarrow (0)10$ or (0000)H

to $1111\ 1111\ 1111\ 1111 \rightarrow (65{,}535)_{10}$ or $(FFFF)_H$

Sign Numbers

Unsigned numbers represent only positive numbersl. To represent both positive and negative numbers i.e. sign numbers various techniques are used because computer does not have provision to represent negative sign. In this section we will see techiniques to represents signed integer numbers

- Sign-magnitude representation
- 1's complement
- 2's complement

Sign-Magnitude Representation

In signed numbers it is necessary to represent negative as well as positive numbers. Fig. shows the sign magnitude format for 8-bit signed number. In sign-magnitude representation the most significant bit (lefmost bit) is used to represent sign of the number. if the most significant bit is 0, the number is positive, and if the most significant bit is 1, the number is negative. The remaining bits of the number represent magnitude.

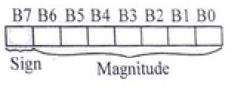


Fig. 4 : Sign magnitude format

Here are some examples of sign-magnitude numbers.

1. +6 = 0 0 0 0 0 1 1 0
2. –14 = 1 0 0 0 1 1 1 0
3. +24 = 0 0 0 1 1 0 0 0
4. –64 = 1 1 0 0 0 0 0 0

In case of unsigned 8-bit binary numbers the decimal range is 0 to 225. For sign magnitude 8-bit numbers the largest magnitude is reduced from 255 to 127 because we need to represent both positive and negative numbers.

Maximum positive number 0111 1111 = + 127

Maximum negative number 1111 1111 = – 127

Drawbacks of sign-magnitude representation

- For addition and subtraction, it is necessary to consider signs of both the numbers and their relative magnitudes in order to carry out the required arithmetic operation.
- There are two representations of zeros :

$$+0_{10} = 0000\ 0000$$
$$-0_{10} = 1000\ 0000$$

Due to the two representations of zero, it is more difficult to test for zero operation frequently perfomed by the computer.

One's Complement Representation

In the 1's complement representation, negative numbers are obtained by complementing each bit of the corresponding positive number. Thus, the representation for –4 is obtained by complementating each bit in the vector 0100 to yield 1011. The operation of obtaining the 1's complement of a number is equivalent to substracting that number from 2ⁿ – 1, that is, from 1111, in the case of the 4-bit numbers.

■ **Example : Find 1's complement of $(1101)_2$.**

Solution :

```
1 1 0 1 ← Number
0 0 1 0 ← 1's complement
```

■ **Example : Find 1's complement of 1011 1001.**

Solution :

```
1 0 1 1  1 0 0 1 ← number
0 1 0 0  0 1 1 0 ← 1's complement
```

Two's Complement Representation

The two's complement number is obtained by subtracting corresponding positive number from 2ⁿ. Hence, the 2's complement number is obtained by adding 1 to the 1's complement number.

■ **Example : Find 2's complement of $(1001)_2$.**

Solution :

```
  1 0 0 1 ← number
  0 1 1 0 ← 1's complement
+       1
  0 1 1 1 ← 2's complement
```

■ **Example : Find 2's complement of $(1\ 0\ 1\ 0\ \ 0\ 0\ 1\ 1)_2$.**

Solution :

```
  1 0 1 0   0 0 1 1  ← number
  0 1 0 1   1 1 0 0  ← 1's complement
+                 1
  0 1 0 1   1 1 0 1  ← 2's complement
```

Table show representation of 4-bit integer numbers in sign-magnitude, 1's complement and 2's complement form.

Decimal	Representations		
	Sign-magnitude	1's complement	2's complement
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
–0	1000	1111	–
–1	1001	1110	1111
–2	1010	1101	1110
–3	1011	1100	1101
–4	1100	1011	1100
–5	1101	1010	1011
–6	1110	1001	1010
–7	1111	1000	1001
–8	–	–	1000

Table : Representation of 4-bit integer numbers in different forms

Advantages of 2's complement representation

- From Table we can note that there are district + 0 and – 0 representations in both the sign magnitude and 1's complement systems, but the 2's complement system has only a + 0 representation.
- For 4-bit numbers, the value – 8 is representable only in the 2's complement system and not in other systems.
- It is more efficient for logic circuit implementation, and one often used in computers, for addition and subtraction operations.

Sign Extension

When we add two binary numbers, the result may extend by 1-bit i.e. the resulted binary number may need one more bit if there is a carry after addition of most significant bit. To represent such a result in correct format we have to allocate the additional bit for representing the magnitude of the numbers which are to be added. Thisis illustrated in Fig.

```
  0 0 1 0 0 0  (+ 8)
+ 0 0 1 0 0 1  (+ 9)
  0 1 0 0 0 1  (+ 17)
```

In case of positive binary number we extend the bit by appending bit-0 (sign bit for positive number) to the left of the most-significant bit of the number. In case of negative binary number can extend the bit by appending bit-1 (sign bit for negative number) to the left of the most-significant bit the number. This is shown below.

In both the case sign-bit of the original number is extended and hence such extension is known as sign-extension of the number.

1. Original number represented using 6-bits : $\boxed{0}$ 1 0 1 0 0 (+20)
   ↖Sign-bit

   Same number represented using 7-bits : $\boxed{0}$ 0 1 0 1 0 0 (+20)
   ↖Sign-extension

   Same number represented using 8-bts : $\boxed{0}\boxed{0}\boxed{0}$ 1 0 1 0 0 (+20)
   ↖Sign-extension

2. Original number represented using 6-bits : $\boxed{1}$ 0 1 1 0 0 (–20)
   in 2's complement form
   ↖Sign-bit

   Same number represented using 7-bits : $\boxed{1}\boxed{1}$ 0 1 1 0 0 (–20)
   in 2's complement from
   ↖Sign-extension

   Same number represented using : $\boxed{1}\boxed{1}\boxed{1}$ 0 1 1 0 0 (–20)
   8-bits in 2's complement from
   ↖Sign-extension

For representing negative numbers, there are two more formats used : signed-1's complement representation and signed-2's complement representation.

Binary Addition and Substraction

We can relate addition and subtraction operations of numbers by the following relationship :

$$(\pm A) - (+B) = (\pm A) + (-B) \text{ and}$$
$$(\pm A) - (-B) = (\pm A) + (+B)$$

Therefore, we can change subtraction operation to an addition operation by changing the sign of the subtrahend. Hence, the binary addition is the key to binary subtraction, multiplication, and division. So, let us see rules for binary addition.

Rules for Binary Addition

A B	Sum	Carry
0 + 0	0	0
0 + 1	1	0
1 + 0	1	0
1 + 1	0	1

Binary Arithmetic- Negative Numbers in 1's Complement Form

Case 1 (Both positive) : Add $(28)_{10}$ and $(15)_{10}$

:

$$\begin{array}{r|c|l}
2 & 28 & 0 \uparrow \text{LSD} \\ \hline
2 & 14 & 0 \\ \hline
2 & 7 & 1 \\ \hline
2 & 3 & 1 \\ \hline
2 & 1 & 1 \ \text{MSD} \\ \hline
 & 0 &
\end{array}
\qquad
\begin{array}{r|c|l}
2 & 15 & 0 \uparrow \text{LSD} \\ \hline
2 & 7 & 0 \\ \hline
2 & 3 & 1 \\ \hline
2 & 1 & 1 \ \text{MSD} \\ \hline
 & 0 &
\end{array}$$

$\therefore$ $(011100)_2 \rightarrow (28)_{10}\,(01111)_2 \rightarrow (15)_{10}$

Addition of 28 and 15

$$\begin{array}{rccccccc|l}
 & & & 1 & 1 & 1 & & & \leftarrow \textbf{Carry} \\
+ & 0 & 0 & 1 & 1 & 1 & 0 & 0 & (28)_{10} \\
 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & (15)_{10} \\ \hline
 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & (43)_{10}
\end{array}$$

Note : Here, the magnitude of greaternumber is 5- bit; however, the magnitude of the result is 6-bit. Therefore, the numbers are sign-extended to 7-bits.

Case 2 (Smaller Negative) : Add $(28)_{10}$ and $(-15)_{10}$

We have $(011100)_2 \rightarrow (28)_{10}$ and
$(01111)_2 \rightarrow (15)_{10}$
$\therefore$ $(10000)_2 \rightarrow$ 1's complement of 15

Addition of 28 and – 15 :

$$\begin{array}{lrcccccc|l}
 & + & 0 & 1 & 1 & 1 & 0 & 0 & (28)_{10} \\
\text{sign extension} \rightarrow & & 1 & 1 & 0 & 0 & 0 & 0 & (-15)_{10} \\ \hline
\text{Carry} \rightarrow & 1 & 0 & 0 & 1 & 1 & 0 & 0 & \\
 & \llcorner & & & & & \rightarrow & 1 & \text{Add end-around carry} \\ \hline
 & & 0 & 0 & 1 & 1 & 0 & 1 & (13)_{10}
\end{array}$$

Case 3 (Greater negative) : Add $(-28)_{10}$ and $(15)_{10}$

We have $(0111100)_2 \rightarrow (28)_{10}$ and
$(01111)_2 \rightarrow (15)_{10}$
$\therefore$ $(100011)_2 \rightarrow (-28)_{10}$

Addition of (– 28) and 15

Addition of – 28 and – 15

Sign-extension →	1 1 0 0 1 0 0	(–28)
+ Sign-extension →	1 1 1 0 0 0 1	(–15)
Ignore carry →	1 0 1 0 1 0 1	(–43) Result is in 2's complement form

Verification : 1 0 1 0 1 0 1

0 1 0 1 0 1 0

+ 1

0 1 0 1 0 1 1 → (43)

Addition and Subtraction with Signed Magnitude Data

In a signed Magnitude data :

Case 1 : When the signs of two numbers say A and B are identical, add the two magnitudes and attach the sign of A to the result.

Case 2 : When the signs of A and B are different compare the magnitudes and subtract the smaller number from larger. Then attach the sign of larger number (in magnitude) to the result. If two magnitudes are equal subtract B from A and make the sign of the result positive.

The Fig. shows the hardware for signed magnitude addition and subtractrion. It consists of two registers A and B to hold the magnitudes of the numbers, two flip flowps A_s and B_s to hold the corresponding signs, complementer, parallel adder, flip-flop E to save output carry and flip-flop AVF to hold the overflow bit when A and B are added. The result operation is transferred into A and A_s. Thus A and A_s together form an accumulator register.

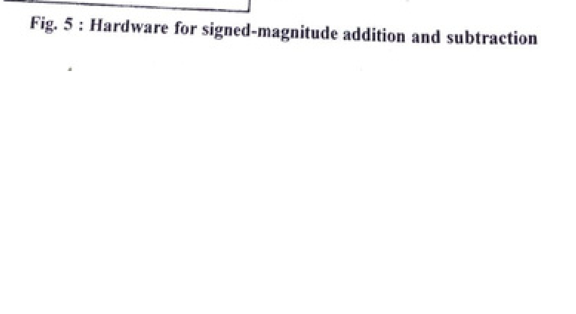


Fig. 5 : Hardware for signed-magnitude addition and subtraction

The parallel adder performs the addition of A and B. The sum output of the adder is applied to the input of the A register. The complementer consists of exclusive OR gates. It provides an output B when M = 0 and the complement of B when M = 1. Thus when M = 0 we have operation A ← A + B and when M = 1, C_{in} = 1 and we have operation A ← A + $\overline{B}$ + 1. Thus, when M = 1, we have A plus the 2's Complement of B, which is equivalent to the subtraction A – B.

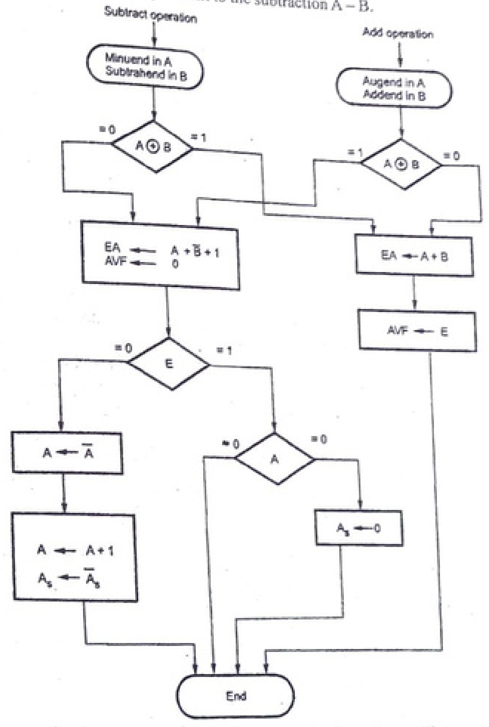


Fig. 6 : Flowchart for add and subtract operation

The Fig. shows flowchart for the hardware algorithm presented in Fig. The signs of A and B are compared by an exclusive- OR gate. If the output of gate is 0, the signs are identical, if it is 1, the

```
                  + 1 0 0 0 1 1     (-28)10
sign extension →    0 0 1 1 1 1     (15)10
                    -----------     -------
                    1 1 0 0 1 0     (-13)10   Result is in 1's
                                              complement form
```

```
Verification : 1 1 0 0 1 0

               0 0 1 1 0 1   → (13)10
```

Case 4 (Both Negative) : Add $(-28)_{10}$ and $(-15)_{10}$

We have $(011100)_2 \rightarrow (28)_{10}$ and

$(01111)_2 \rightarrow (15)_{10}$

∴ $(10000)_2 \rightarrow$ 1's complement of 15

$(100011)_2 \rightarrow$ 1's complement of 28

Addition of (–28) and (–15)

```
Sign-extension →      1 1 0 0 0 1 1
                    +
Sign-extension →      1 1 1 0 0 0 0
                      -------------
Carry →           [1] 1 0 1 0 0 1 1
                   |
                   └────────────→ 1   Add end-around carry
                      -------------
                      1 0 1 0 1 0 0   (-43) Result is in 1's
                                      complement form
```

```
Verification : 1 0 1 0 1 0 0

               0 1 0 1 0 1 1      → (43)10
```

Note :

- Here, the magnitude of greater number is 5-bit; however, the magnitude of the result is 6-bit. Therefore, the numbers are sign-extended to 7-bits.
- For proper result we suggest to use 1 sign-bit extension to the number having greater magnitude and represent the number having smaller magnitude with extended number of bits.

Binary Arithmetic-Negative Numbers in 2's Complement form

***Case I (Both Positive)* : Add $(28)_{10}$ and $(15)_{10}$**

We have $(011100)_2 \rightarrow (28)_{10}$ and

$(01111)_2 \rightarrow (15)1_0$

Addition of 28 and 15

```
Sign-extension →    0 0 1 1 1 0 0      (28)10
                  +
Sign-extension →    0 0 0 1 1 1 1      (15)10
                    -------------      -------
Carry →             0 1 0 1 0 1 1      (43)10
```

Case 2 (Smaller Negative) : Add $(28)_{10}$ and $(-15)_{10}$

We have $(011100)_2 \rightarrow (28)_{10}$ and

$(01111)_2 \rightarrow (15)_{10}$

∴ $(10001)_2 \rightarrow$ 2's complement of 15

Addition of 28 and – 15

```
Sign-extension →    0 0 1 1 1 0 0      (28)10
                  +
Sign-extension →    1 1 1 0 0 0 1      (15)10
                    -------------      -------
Ignore carry →      0 0 0 1 1 0 1      (13)10
```

Case 3 (Greater Negative) : Add $(-28)_{10}$ and $(15)_{10}$

We have $(011100)_2 \rightarrow (28)_{10}$ and

$(01111)_2 \rightarrow (15)_{10}$

∴ $(100100)_2 \rightarrow$ 2's complement of 28

Addition of (–28) and (15)

```
Sign-extension →    1 1 0 0 1 0 0      (-28)10
                  +
Sign-extension →    0 0 0 1 1 1 1      (15)10
                    -------------      -------
                    1 1 1 0 0 1 1      (-13)10   Result is in 2's
                                                 complement form
```

```
Verification : 1 1 1 0 0 1 1

               0 0 0 1 1 0 0
             +             1
               -------------
               0 0 0 1 1 0 1 → (13)10
```

Case 4 (Both Negative) : Add $(-28)_{10}$ and $(-15)_{10}$

We have $(011100)_2 \rightarrow (28)_{10}$ and

$(01111)_2 \rightarrow (15)_{10}$

∴ $(100100)_2 \rightarrow$ 2's Complement of 28

$(10001)_2 \rightarrow$ 2's complement of 15 signs are different. For an add operation, identical signs tell that the magnitudes be added. For a **subtract** operation, different signs tell that the magnitudes be added. The microoperation $EA \leftarrow A + B$ adds the magnitudes and stores the carry if any in E register. The carry in E after the addition constitutes an overflow if it is equal to 1. The value of E is transferred into the add-overflow flip-flop AVF.

The two magnitudes are subtracted if the signs are different for an add operation or identical for a subtract operation. Their magnitude are subtracted by adding A to the 2's complement of B. No overflow can occur if the numbers are subtracted so AVF is cleared to 0. A logic 1 in E indicates that $A \geq B$ and the number in A is the correct result. If this numbers is zero (in case $A = B$), the sign of must be made positive to avoid a negative zero. A logic 0 in E indicates that $A < B$ and it is necessary to take 2's complement of the result, i.e., $A \leftarrow \bar{A} + 1$ When $A < B$, the sign of the result is the ocmplement of the original sign of A. it is then necessary to complement A_s to obtain the correct sign. However, in other paths of flowchart the sign of the result is same as the sign of A so no change in A_s is required. The final result is available in register A and its sign in A_s. The value in AVF indicates overflow condition and the value in E is ignored.

Addition and Subtraction with Signed 2's Complement Data

The Fig. Shows hardware for adding and subtracting two binary numbers in signed 2's complement representation. The sum is obtained by adding the contents of A and B registers including their sign bits. The overflow bit V is set to 1 if the exclusive OR of the last two carries is 1, and it is clearedto 0 otherwise. The subtraction operation is accomplished by adding the contents of A to the 2's complement of B.

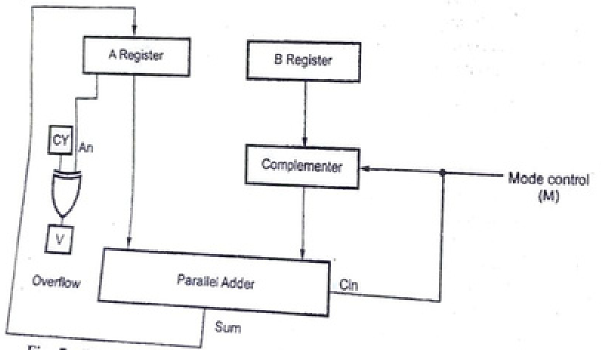


Fig. 7 : Hardware for signed - 2's complement addition and subtraction

When the signs of two numbers are same, the overflow may occur. Thus overflow must be checked and if it occurs, the result in register A is erroneous. The Fig. shows the flowchart for adding and subtracting numbers in signed 2's complement representation.

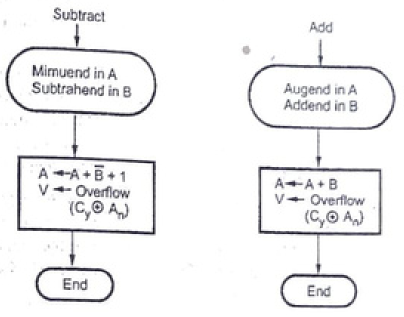


Fig. 8 : Flowchart for adding and subtracting number in signed 2's complement representation

Carry Look Ahead Adder

The n-bit adder discussed in the last section is implemented using full adder stages. In which the carry output of each full adder stage is connected to the carry input of the next higher order stage. Therefore, the sum and carry outputs of any stage cannot be produced until the input carry occurs; this leads to a time delay in the addition process. This delay is known as **carry propagation delay,** which can best explained by considering the following addition.

$$\begin{array}{r} 0\ 1\ 0\ 1 \\ +\ 0\ 0\ 1\ 1 \\ \hline 1\ 0\ 0\ 0 \end{array}$$

Addition of the LSB position produces a carry into the second position. This carry when added to the bits of the second position (stage), produces a carry into the third position. The latter carry, when added to the bits of the third position, produces a carry into the last position. The key thing to notice in this example is that the sum bit generated inthe last position (MSB) depends on the carry that was generated by the addition in the previous positions. This means that, adder will not produce correct result until LSB carry has propagated through the intermediate full adders. This represents a time delay that depends on the propagation delay produced in an each full adder. For example, if each full ader is considered to have a propagation delay of 30 ns, then S_3 willnot reach its correct value until 90 ns after LSB carry is generated. Therefore, total time required to perform addition is $90 + 30 = 120$ ns.

Obviously, this situation becomes much worse if we extend the adder circuit to add a greater registers are shifted to the right one bit, so that the C bit goes into A_{n-1}, A_0 goes into Q_{n-1} and Q_0 is lost. If bit 0 (Q_0) is 0, then no addition is performed only shift operation is carried out.

3. Steps 1 and 2 are repeated n times to get the desired result in the A and Q registers.

A flowchart for multiplication operation is shown in Fig.

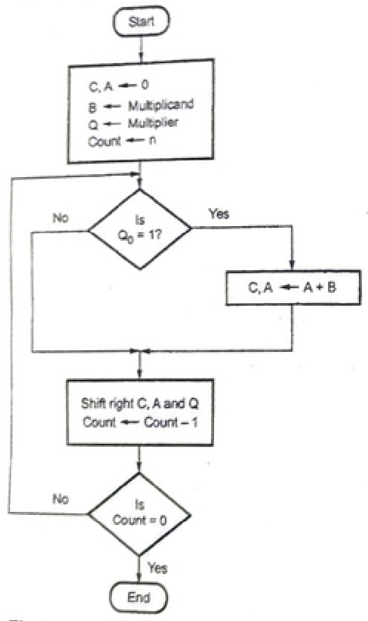


Fig. 15 : Flowchart for multiplication operation

Let us see one example.

Consider 4-bit multiplier and multiplied :

Multiplier = 1 1 0 1 and

Multiplicand = 1 0 1 1

Fig. shows operations involved and their results in the multiplication process.

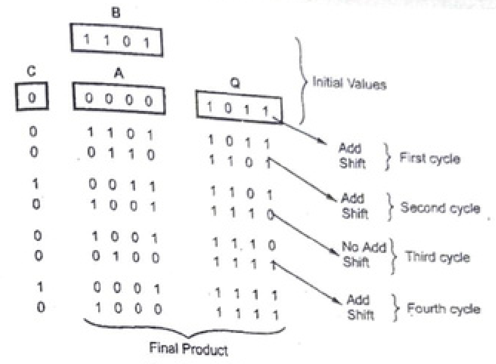


Fig. 16 : Multiplication process

Complement Multiplication Algorithms

In this section we discuss multiplication of 2's complement signed operands. Here also we follow the same strategy that to accumulate partial products by adding versions of multiplicand (either positive or negative) as selected by the multiplier bit. It is important to note that multiplicand when we add negative multiplicand we must extend the sign bit value of the multiplicand to the left as far as product will extend. Refer fig. The sign extension of the multiplicand is shown bold. We now discuss a techinique which works equally well for both negative and positive multiplier, called **Booth algorithm.**

Booth's Algorithm

A powerful algorithm for signed number multiplecation is a Booth's algorithm, which generates a 2n bit product and treats both positive and negative numbers uniformly.

This algorithm suggest that we can reduce the number of operations required for multiplication by representing multiplier as a difference between two numbers. For example, multiplier 0 0 1 1 1 0 (14) can be represented as follows.

$$\begin{array}{rr} 0\,1\,0\,0\,0 & (16) \\ -\;0\,0\,0\,1\,0 & (2) \\ \hline 0\,0\,1\,1\,1\,0 & (14) \end{array}$$

Therefore, the product can be computed by adding 2^4 times the multiplicand to the 2's complement of 2^1 times the multiplicand. In simple notations, we can describe the sequence of required operations by **recoding** the preceding multiplier as

0 + 1 0 0 – 1 0

In genera, for Booth's algorithm recoding scheme can be given as :

–1 times the shifted multiplicand is selected when moving from 0 to 1, +1 times the shifted number of bits. If the adder were handling 16-bit numbers, the carry propagation delay could be 480 ns.

One method of speeding up this process by eliminating inter stage carry delay is called **look ahead-carry addition.** This method untilizes logic gates to look at the lower order bits of the augend and addend to see if a higher order carry isto be generated. It uses two functions : Carry generate and carry propagate.

Consider the circuit of the full adder shown in Fig. Her, we define two functions carry generate and carry propagate.

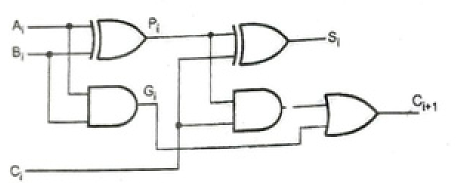


Fig. 9 : Full adder circuit

$$P_i = A_i \oplus B_i$$

$$G_i = A_i B_i \qquad \text{(Refer Appendix A for details)}$$

The output sum and carry can be expressed as

$$S_i = P_i \oplus C_i$$

$$C_i + 1 = G_i + P_i C_i$$

G_i is called a carry generate and it produces on carry when both A_i and B_i are one, regardles of the input carry. P_i is called a carry propagate because it is term associated with the propagation of the carry from C_i to C_{i+1}. Now C_{i+1} can be expressed as a sum of products function of the P and G outputs of all the preceding stages. For example, the carriers in a four stage carry lookahead adder are defined as follows :

$$C_1 = G_0 + P_0 C_{in}$$

$$C_2 = G_1 + P_1 C_1 = G_1 + P_1 G_0 + P_1 P_0 C_{in}$$

$$C_3 = G_2 + P_2 C_2 = G_2 + P_2 G_1 + P_2 + P_1 G_0 + P_2 P_1 P_0 C_{in}$$

$$C_4 = G_3 + P_3 C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_{in}$$

Fig. shows the general form of a carry lookahead adder circuit designed in this way.

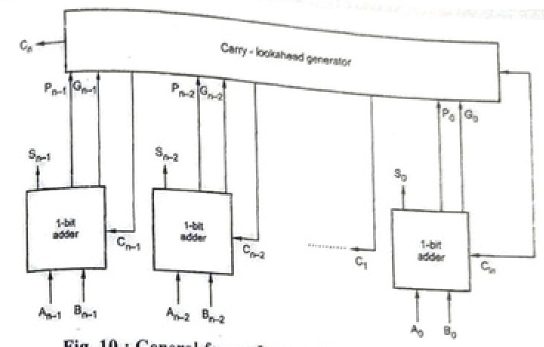


Fig. 10 : General form of a carry lookahead adder circuit

We can further simplify the design by noting that the sum equation of stage i.

$S_i = A_i \oplus B_i \oplus C_i$ as $S_i = P_i \oplus G_i \oplus C_i$ (Refer Appendix-A for details.)

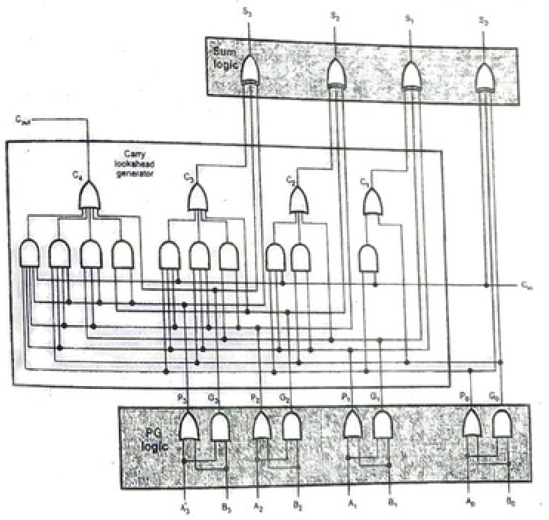


Fig.11

Serial Adder

The shift registers with added circuitry can be used to design a serial adder. The Fig. shows the logic diagram of serial adder. It consists of two right shift register, full adder and D-flip-flop. A full adder is used to perform bit by bit addition and D-flip-Flip is used to store the carry output generated after addition. This carry is used as carry input for the next addition. Initially, the 'D-Flip-Flip is cealred and addition starts with the least significant bits of both register. After each clock pulse data within the right shift registers are shifted right, 1-bit and we get bits from next digit and carry of previous addition as new inputs for the full adder. The sum bit of the full adder is connected as a serial input of shift register A. Thus the result of serial addition gets stored in the register A. The new number can be added to the contents of register A by loading a new number into register B and repeating the process of serial addition.

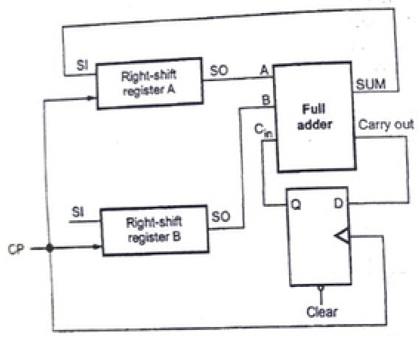


Fig. 12 : Serial adder

Multiplication Algorithm

The multiplication is a complex operation than addition and subtraction. It can be performed in hardware or software. A wide variety of algorithms have been used in various computers. For simplicity we will first see the mulitplication algorithms for unsigned (postive) integers, and then we will see the multiplication algorithm for signed numbers.

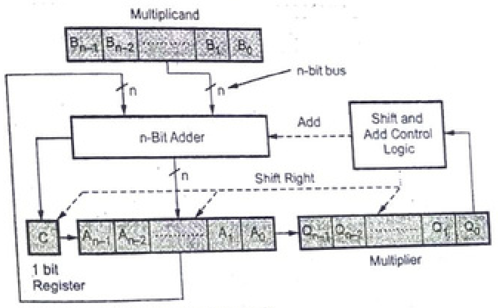
Fig. shows the usual algorithm for multiplying positive numbers by hand. Looking at this algorithms we can note following points :

```
   ×1101   (13) Multiplicand
    1000    (9) Multiplier
    ────
    1101 ┐
   0000  │
  0000   ├ Partial Products
 1101    ┘
 ───────
 110101     Final Product (117)
```

Fig. 13 : Manual multiplication algorithm

- Multiplication process involves generation of partial products, one for each digit in the multiplier. These partial products are then summed to produce the final product.
- In the binary system the partial products are easily defined. When the multiplier bit is 0, the partial prduct is 0, and when the multiplier is 1, the partial prudct is the multiplicand.
- The final product is produced by summing the partial products. Before summing operation each successive partial product is shifted one position to the left relative to the preceding partial product, as shwon in the Fig.
- The product of two n-digit numbers can be accommodated in 2n digits, so the product of the two 4-bit numbers inthis example fits into 8-bits, as shown.

Fig. shows the implementation of mannual multiplication approach. It consists of n-bit binary adder, shift and add control logic and four registers, A, B, C and Q. As shown in the Fig. multiplier and multiplicand are loaded into register Q and register B, respectively and C are initially set to 0.


Multiplicand
B_{n-1} B_{n-2} B_1 B_0
n-bit bus
n
n
n-Bit Adder
Add
Shift and Add Control Logic
n
Shift Right
C
A_{n-1} A_{n-2} A_1 A_0
Q_{n-1} Q_{n-2} Q_1 Q_0
1 bit Register
Multiplier

Note : Dotted lines indicate control signals

Fig. 14 : Hardware implementation of unsigned binary multiplication

Multiplication Operation Steps

1. Bit 0 of multiplier operand (Q_0 of Q register) is checked.
2. If bit 0 (Q_0) isone then multiplicand and partial product are added and all bits of C, A and Q

multiplicand is selected when moving from 1 to 0, and 0 times the shifted multiplicand is selected for none of the above case, as multiplier is scanned from **right to left.**

We have to assume an implied 0 to right of the multipleir LSB. This is illustrated in the following examples.

■ **Example : Recode the multiplier 1 0 1 1 0 0 for Booth's multiplication.**

Solution :

```
                 ←Implied zero
  1 0 1  1 0 0 0   Multiplier
 -1 +1 0 -1 0 0    Recoded multiplier
```

■ **Example : Recode the multiplier 0 1 1 0 0 1 for Booth's multiplication.**

Solution :

```
                 ←Implied zero
  0  1  1  0  0  1 0   Multiplier
 +1  0 -1  0 +1 -1     Recoded multiplier
```

The Fig. shows the Booth's multiplication. As shown in the Fig. whenever multiplicand is multiplied by –1, its 2's complement is taken as a partial result.

Multiplier : 0 0 1 1 0 0 Multiplicand : 0 1 0 0 1 1.

Recoded multiplier : 0 + 1 0 – 1 0 0

Multiplication :

```
              0 10 011
          × 0+1 0-1 0 0
   ---------------------
   0 0 0 0 0 00 00 000
   0 0 0 0 0 00 00 00
   1 1 1 1 1 01 10 1      ← 2's complement of the
   0 0 0 0 0 00 00          multiplicand
   0 0 0 1 0 01 1
   0 0 0 0 0 00 0
   ---------------------
   0 0 0 0 1 11 00 100
```

Fig. : Booth's multiplication

The same algorithm can be sued for negative multiplier. This is illustrated in the following example.

■ ***Example : Multiply 0 1 1 1 0 (+14) and 1 1 0 1 1 (–5)***

Solution :

```
 0  1  1  1  0   (+14) Multiplicand
 1  1  0  1  1   (–5) Multiplier
 0 -1  1  0 -1   Recoded Mutliplier
```

Multiplication :

```
            0   1  1   1  0
          × 0  -1 +1   0 -1
 ---------------------------
 1 1 1 1 1 1   0 0   1 0   2's complement of the
 0 0 0 0 0 0   0 0   0     multiplicand
 0 0 0 0 1 1   1 0
 1 1 1 0 0 1   0
 0 0 0 0 0 0
 ---------------------------
 1 1 1 0 1 1   1 0   1 0   (–70)
```

The same algorithm also can be used for negative multiplier and negative multiplicand. This is illustrated in the following example.

■ **Example : Explain the following pair of signed 2's complement numbers.**

Multipolicad : 1 1 0 0 1 1 (– 13)

Multiplier : 1 0 1 1 0 0 (–20)

Solution :

```
  1  0 1  1 0 0  Multiplier
 -1 +1 0 -1 0 0  Recoded Multiplier
```

Multiplication :

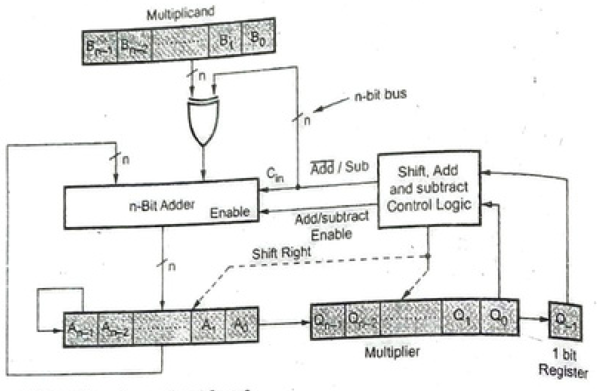
```
                      1  1  0  0  1  1   Multiplicand
                   × -1 +1  0 -1  0  0   Recoded Multiplier
 ----------------------------------------
 0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  1  1  0  1            ← 2's complement of the
                                            multiplicand
 0  0  0  0  0  0  0  0  0
 1  1  1  1  0  0  1  1
 0  0  0  1  1  0  1                     ← 2's complement of the
                                            multiplicand
 ----------------------------------------
 0  0  0  1  0  0  0  0  0  1  0  0      (260)
```

Hardware Implementation

The Booth's algorithm can be implemented as shown in the Fig. The circuit is similar to circuit for positive number multiplication. It consists of n-bit adder, shift, add subtract control logic and four registers, A, B, Q and Q_{-1}. As shown in the Fig. multiplier and multiplicand are loaded into register Q and register B, respectively, and register A and Q_{-1} are initially set to 0.

The n-bit adder performs addition of two inputs. One input is the A register and other input is multiplecand. In case of addition, $\overline{\text{Add}}$/sub line is 0, therefore $C_{in} = 0$ and multiplicand is directly applied as a second input to the n-bit adder. In case of subtraction, $\overline{\text{Add}}$/sub line is 1, therefore C_{in} =

1 and multiplicand is complemented and then applied to the n-bit adder. As a result, the 2's complement of multiplicand is added in the A register.



I Initial settings : A ← 0 and Q_{-1} = 0

Fig. 17 : Hardware implementation of signed binary multiplication

The shift, add and subtract control logic scans bits Q_{10} and Q_{-1} one at a time and generates the control signals as shown in the Table. If the two bits are same (–1 or 0 – 0), then all of the bits of the A, Q, and Q_{-1} registers are shifted to right 1 bit without addition or subtraction (Add/subtract Enable = 0). If the two bits are differ, then the multiplicand (B-register) is added to or subtracted from the A register, depending on the status of bits. If bits are $Q_0 = 0$ and $Q_{-1} = 1$ then multiplicand is added and if bits are $Q_0 = 1$ and $Q_{-1} = 0$ then multiplicand is subtracted. After addition or subtraction right shift occurs such that the leftmost bit of A (A_{n-1}) is not only shifted into A_{n-2}, but also remains in A_{n-1}. This is required to preserve the sign of the number in A and Q. It is known as an arithmeic shift, since it preserves the sign bit.

Q_0	Q_{-1}	$\overline{\text{Add}}$/sub	Add/Subtract Enable	Shift
0	0	×	0	1
0	1	0	1	1
1	0	1	1	1
1	1	×	0	×

Table : Truth table for shift, add and subtract control logic

The sequence of events in Booth's algorithm can be explained with the help of flowchart shown in Fig.

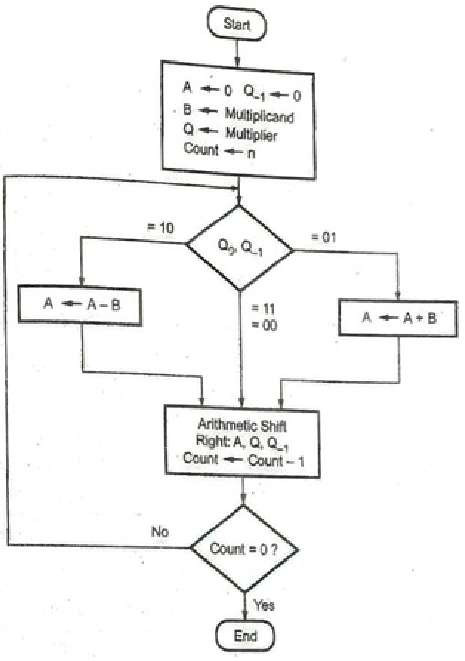


Fig. 18 : Booth's algorithm for signed multiplication

Let us see the multiplication of 4-bit numbers, 5 and 4 with all possible combinations.

Case 1 : Both Positive (5 × 4)

Multiplicand (B) ← 0 1 0 1 (5)		Multiplicand (Q) ← 0 1 0 0 (4)		
Steps	**A**	**Q**	**Q_{-1}**	**Operation**
	0 0 0 0	0 1 0 0	0	Initial
Step 1 :	0 0 0 0	0 0 1 0	0	Arithmetic shift right

Step 2 :	0000	0001	0	Arithmetic shift right
Step 3 :	1011	0001	0	A ← A − B
	1101	1000	1	Arithmetic shift right
Step 4 :	0010	1000	1	A ← A + B
	0001	0100	0	Arithmetic shift right
Result :	0001	0100	= + 20	

Case 2 : Negative Multiplier (5 × –4)

Multiplicand (B) ← 0 1 0 1 (5)		Multiplicand (Q) ← 1 1 0 0 (–4)		
Steps	A	Q	Q_{-1}	**Operation**
	0000	1100	0	Initial
Step 1 :	0000	0110	0	Arithmetic shift right
Step 2 :	0000	0011	0	Arithmetic shift right
Step 3 :	1011	0011	0	A ← A − B
	1101	1001	1	Arithmetic shift right
Step 4 :	1110	1100	1	Arithmetic shift right
Result :	1110	1100	= – 20	(2's complement of 20)

Case 3 : Negative Multiplicand (–5 × 4)

Multiplicand (B) ← 1 0 1 1 (–5)		Multiplicand (Q) ← 0 1 0 0 (–4)		
Steps	A	Q	Q_{-1}	**Operation**
	0000	0100	0	Initial
Step 1 :	0000	0010	0	Arithmetic shift right
Step 2 :	0000	0001	0	Arithmetic shift right
Step 3 :	0101	0001	0	A ← A − B
	0010	1000	1	Arithmetic shift right
Step 4 :	1101	1000	1	A ← A − B
	1110	1100	0	Arithmetic shift right
Result :	1110	1100	= – 20	

Case 4 : Both Negative (–5 × 4)

Multiplicand (B) ← 1 0 1 1 (–5)		Multiplicand (Q) ← 1 1 0 0 (–4)		
Steps	A	Q	Q_{-1}	**Operation**
	0000	1100	0	Initial
Step 1 :	0000	0110	0	Arithmetic shift right
Step 2 :	0000	0011	0	Arithmetic shift right
Step 3 :	0101	0011	0	A ← A − B
	0010	1001	1	Arithmetic shift right
Step 4 :	0001	0100	1	Arithmetic shift right
Result :	0001	0100	= + 20	

Modified Booth's Algorithm

To speed up the multiplication process in the Booth's algorithm a technique called bit pair recoding is used. It is also called **modified Booth's algorithm.** It halves the maximum number of summands. Inthis techinique, the Booth-recoded multiplier bits are grouped in pairs. Then each pair is represented by its equivalent single bit multiplier reducing total number of multiplier bits to half. For example pair (+1 –1) is equivalent to the pair (0 + 1). That is, instead of adding – 1 times multiplicand t shifted position i to –1 times the multiplicand at position i + 1, the same result is obtaned by adding +1 times multiplicand at position i. Similarly, (+1 0) is equivalent to (0 + 2), (–1 +1) is equivalent to (0 –1), and so on. By replacing pairs with their equivalents we can get bit pair recoded multiplier. But instead of deriving bit-pair recoded multiplier from Booth recoded multiplier one can directly derive it from original multiplier. The bit9pair recoding at multiplier can be directly derived from Table. The table shows the bit pair order for all possible multiplier bit options.

Multiplier bit-pair		Multiplier bit on the right	Bit-pair recoded multiplier bit at position i
i + 1	i	i – 1	
0	0	0	0
0	0	1	+1
0	1	0	+1
0	1	1	+2
1	0	0	–2
1	0	1	–1
1	1	0	–1
1	1	1	0

■ **Example : Find the bit-pair code for multiplier.**

1 1 0 1 0.

Solution : By referring Table we can derive bit-pair code as follows :

```
 1   1   1   0   1   0   0  ← Implied 0 to
 └───────┴───────┴───────┘     right of LSB
     0      -1      -2
```

■ **Example : Multiply given signed 2's complement numbers using bit-pair recoding**

A = 110101 multiplicand (–11)

B = 011011 multiplier (+27)

Solution : Let us find the bit-pair code for multipler.

```
 0   1   1   0   1   1   0  ← Implied 0 to
 └───────┴───────┴───────┘     right of LSB
    +2      -1      -1
```

Multiplication :

```
                    1  1  0  1  0  1
                 ×    +2    -1    -1
 ──────────────────────────────────
 0 0 0 0 0 0 0 0 1  0  1  1  ← 2's complement of
                                the multiplicand
 0 0 0 0 0 0 1 0 1  1        ← 2's complement of
                                the multiplicand
 1 1 1 0 1 0 1 0             ← Multiplicand × (+2)
 ──────────────────────────────────
 1 1 1 0 1 1 0 1 0  1  1  1    (–297)
```

Array Multiplier

The multiplication process for bianry numbers is similar to the decimal numbers. Actually binary multiplication is simple than decimal multiplication since it involves only 1s and os.

Rules for binary Multiplication

$0 \times 0 = 0$

$0 \times 1 = 0$

$1 \times 0 = 0$

$1 \times 1 = 1$

■ **Example : Multiply 011_2 by 110_2 using binary multiplication method.**

Solution :

```
        0 1 1
      × 1 1 0
      ───────
        0 0 0
+     0 1 1 0  ← shift left
+   0 1 1 0 0  ← shift left
    ─────────
    1 0 0 1 0
```

We have seen how multiplication takes place in binary numbers. It uses n shits and adds to multiply n-bit binary number. The combination logic circuit implemented to perform such multiplication is called combinational multiplier or **array multiplier.**

Let us generalize the multiplication process for a 2 × 2 multiplier for two unsigned 2-bit numbers : multiplicand $A = A_1 A_0$ and multiplier $B = B_1 B_0$. The Fig. shows how the multiplication process is carried out.

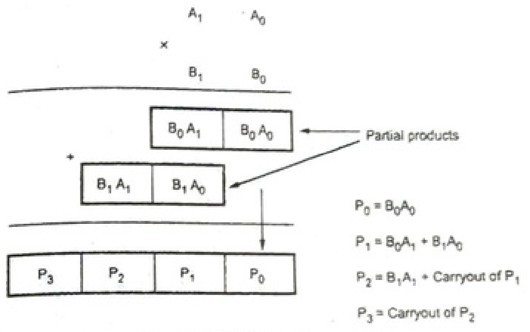


Fig. 19 : Multiplication process

The multiplication processinvolves multiplication (product) of 2-bit number and differtion of 2-bt number. The multiplication of 2-bit number. The multiplication of 2-bits can be implemented using 2-input AND gate whereas addition of 2-bits can be implemented using half adder. Such an complementation of 2 × 2 multiplier is shown in the Fig.

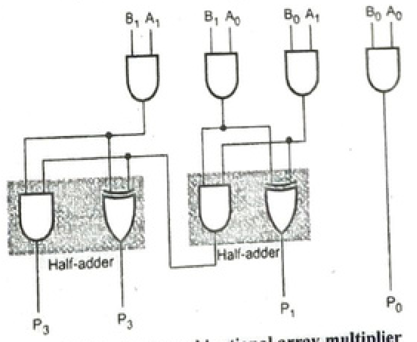
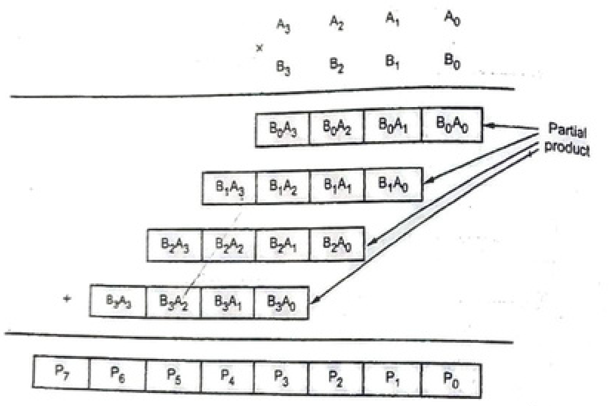


Fig. 20 : 2 × 2 bit combinational array multiplier

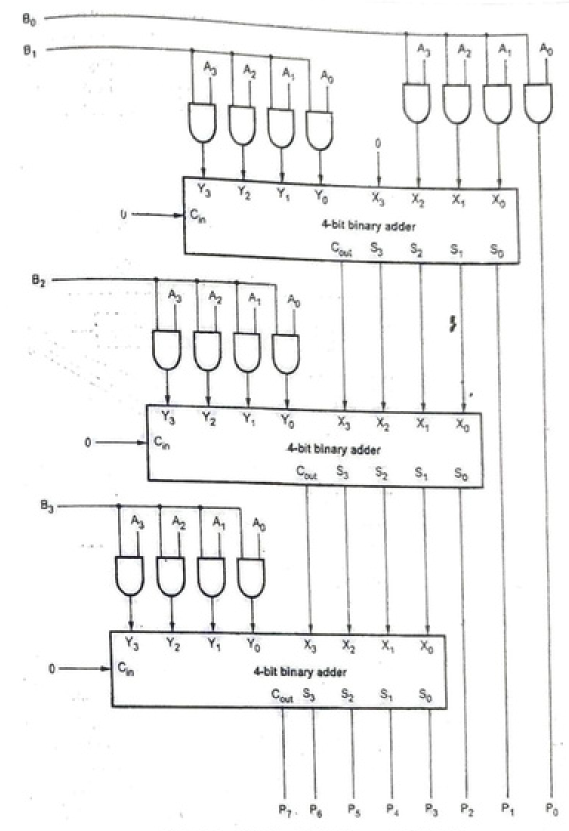
The Fig. shows the multiplication process for a 4 × 4 multiplier for two unsigned integers: multiplicand A = A_3 A_2 A_1 A_0 and multiplier B = B_3 B_2 B_1 B_0. As shown in Fig. each shifted multiplicand which is multiplied by either 0 or 1 depedning on the corresponding multiplier bit is called partial product. Each partial product consists of four product component. The product is represented by a product term surrounded by rectangular box. The final 8-bit product is obtained by adding all partial products.



A_3 A_2 A_1 A_0
×
B_3 B_2 B_1 B_0
B_0A_3 B_0A_2 B_0A_1 B_0A_0
Partial product
B_1A_3 B_1A_2 B_1A_1 B_1A_0
B_2A_3 B_2A_2 B_2A_1 B_2A_0
+
B_3A_3 B_3A_2 B_3A_1 B_3A_0
P_7 P_6 P_5 P_4 P_3 P_2 P_1 P_0

Fig. 21 : 4 bit by 4-bit binary multiplication

The first partial product is formed by multiplying B_0 by A_3 A_2 A_1 A_0. The second partial product is formed by multiplying B_1 by A_3 A_2 A_1 A_0. The third partial is formed by multiplying B_2 by A_3 A_2 A_1 A_0 and the fourth partial product is formed by multiplying A_3 A_2 A_1 A_0. The multiplication of two bits such as B_0 and A_0 produces a 1 if both are 1; otherwise, it produces a 0. This is identicalto AND operation. Therefore, the initial products can be implemented with AND gates as shown in Fig. The 4-bit initial products are added using 4-bit parallel adder. During addition of first partial product, three most significant bits of it are added to the second partial product. As we only three bits from first partial product, the fourth (most significant bit) is considered. This is illustrated in Fig. The three most significant bits and carry out (treated as significant bit) of first partial sum are then added to the third partial product. Finally three most significant bits and carry out (treated as most significant bit) of second sum are added to the fourth partial product. The carryout and third sum represents five most significant bits of the product. Least significant bits of first and second partial sum represents P_1 and P_2' respectively, and product B_0A_0 respresents P_0.



B_0
B_1
A_3 A_2 A_1 A_0
0
Y_3 Y_2 Y_1 Y_0 X_3 X_2 X_1 X_0
C_{in}
4-bit binary adder
C_{out} S_3 S_2 S_1 S_0
B_2
B_3
P_7 P_6 P_5 P_4 P_3 P_2 P_1 P_0

Fig. 22 : 4-bit by 4-bit binary multiplier

Division Algorithm

The division is more complex than multiplication. For simplicity we will see division for positive numbers. Fig shows the usual algorithm for dividing positive numbers by. It shows examples of decimal division and the binary coded division of the same value.

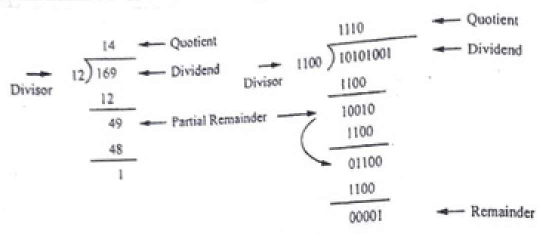


Fig. 23 : Division examples

In both the division, division process is same, only in binary division quotient bits are 0 and 1. We will now see the binary division process in detail. First, the bitrs of the dividend are examined from left to right, until the set of bits examined represents a number greater than or equal tothe divisor; this is referred to as the divisor being able to divide the number. Until this condition occurs, 0s are placed in the quotient from left to right. When the condition is satisfied, a 1 is placed inthe quotient and the divisor is subtracted from the partial dividend. The result is referred to as a **partial remainder.** From this point onwards, the division process follows repetition of steps. In each repetiotion cycle, additional bits from the dividend are brought down to the partial remainder until the result to produce a new partial remainder. The process continues until all the bits of the dividend are brought down and result is still less than the divisor.

Restoring Division

Fig. shows the hardware for implementation of division process described in the previous section. It consists of n + 1-bit binary adder, shift, add and subtract control logic and registers A, B and Q. As

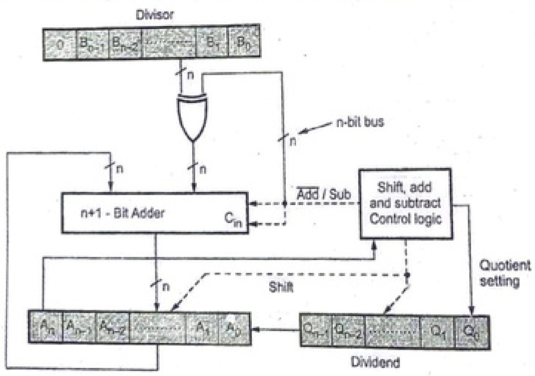


Fig. 24 : Hardware to implement binary division

shown in the Fig. divisor and dividend are loaded into register B and register Q, respectively. Register A is initially set to zero. The division operation is then carried out. After the division is complete the n-bit quotient is in register Q and the remainder is in register A.

Division Operation Steps

1. Shift A and Q left one binary position.
2. Subtract divisor from A and place answer back in A ($A \leftarrow A - B$).
3. If the sign bit of A is 1, set Q_0 to 0 and add divisor back to A (that is, restore A) Otherwise, set Q_0 to 1.
4. Repeat steps 1, 2 and 3 n times.

A flowchart for division operation is as shown in Fig. (See fig. on need page)

Let us see one example. Consider 4-bit dividend and 2-bit divisor :

Dividend = 1 0 1 0

Divisor = 0 0 1 1

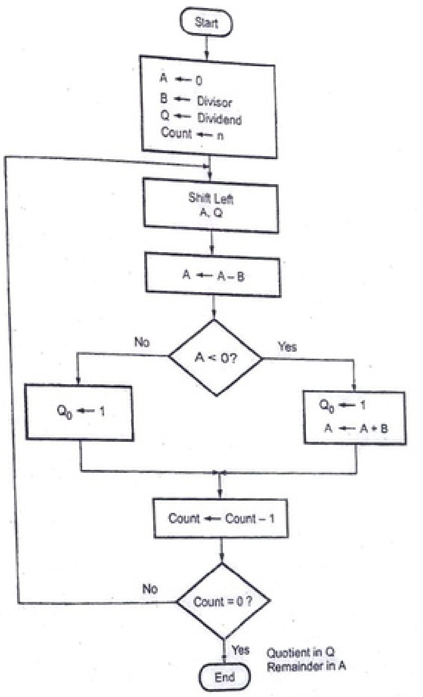


Fig. 25 : Flowchart for restoring division operation

Fig. show steps involved in the above binary division.

The division algorithm just discussed needs restoring register A after each unsuccessful subtraction (Subtraction is said to be unsuccessful if the result is negative). Therefore it is referred to as **restoring division algorithm.** This algorithm is improved, giving non-restoring division algorithm. Consider the sequence of operations that takes place after the subtraction operation in the **restoring algorithm.**

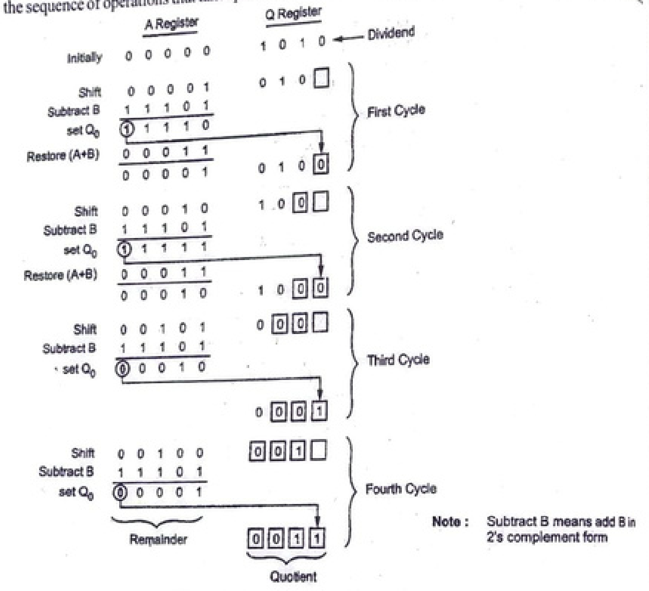


Fig. 26 : A restoring division example

If A is positive	If A is negative
Shift left and subtract divisor $\rightarrow$ 2A – B	Restore $\rightarrow$ A + B Shift left and subtract divisor $\rightarrow$ 2(A + B) – B = 2A + B

Looking at the above operations we can write following steps for non-restoring algorithm.

Step 1: If the sign of A 0, shift A and Q left one bit position and subtract divisor from A; otherwise, shift A and Q left and add divisor to A.

If the sign of A is 0, set Q_0 to 1; otherwise set Q_0 to 0.

Step 2: Repeat steps 1 and 2 for n times.

Step 3: If the sign of A is 1, add divisor to A.

Note : Step 3 is required to leave the proper positive remainder in A at the end of in cycles.

Non Restoring Division

A flowchart for non-restoring division operation is as shown in Fig. Let us see one example. Consider 4-bit dividend and 2-bit divisor :

Dividend = 1 0 1 0

Divisor = 0 0 1 1

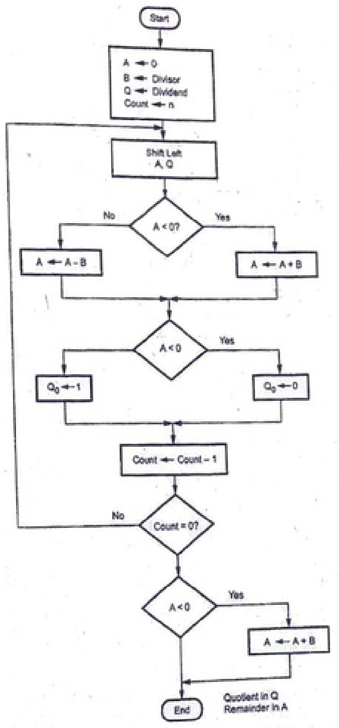


Fig. 27 : Flowchart for non-restoring division operation

Fig. shows steps involved in the non-restoring binary division.

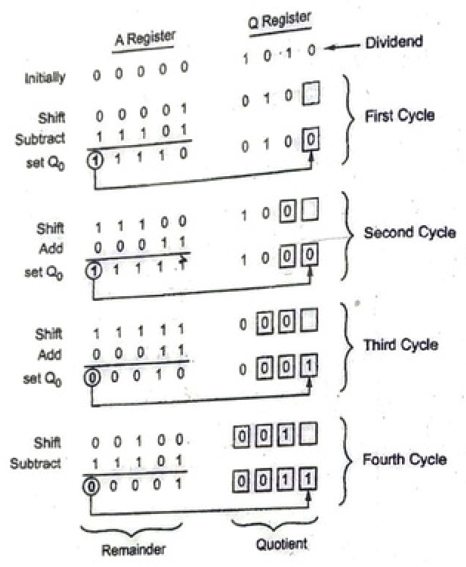


Fig. 28 : A non restoring division example

In the above example after 4 cycles register A is positive and hence step 3 is not required. Let us see another example where we need step 3. Consider 4-bit dividend and 2-bit divisor.

Dividend = 1 0 1 1

Divisor = 0 1 0 1

Fig. shows steps involved in the nonrestoring binary division.

The hardware shown in Fig. can also be used to perform non-restoring algorithm. There is no simple algorithm for signed division. In signed division, the operands are preprocessed to transform them into positive values. Then using one of the algorithms just discussed quotients and remainders are calculated. The quotients and remainders are then transformed to the correct signed values.

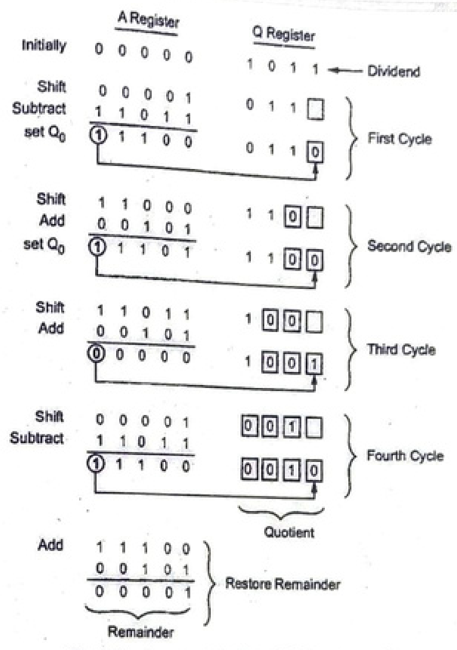


Fig. 29 : A non-restoring division example

Comparison between Restoring and Non-Restoring Division Algorithm

S.No.	Restoring	Non-Restoring
1.	Needs restoring of register A if the result of subtraction is negative.	Does not need restoring.
2.	In each cycle content of register A is first shifted left and then divisor is subtracted from it.	In each cycle content of register A is first shifted left and then divisor is added or subtracted with the content of register A depending on the sign of A.
3.	Does not need restoring of remainder.	Needs restoring of remainder if remainder is negative.
4.	Slower algorithm.	Faster algorithm.

Floating Point Arithmetic Operations

In this section we are going to see general procedures for addition, subtraction, multiplication and division of floating point numbers.

Addition and Subtraction

Consider two floating point numbers :

$X = m_1 . r^{e1}$ and

$Y = m_2 . r^{e2}$ Assume : $e_1 \geq e_2$

Let us see the rules for addition and subtraction.

Step 1: Select the number with a smaller exponent and shift its mantissa right, a number of steps equal to the difference in exponents $|e_2 - e_1|$. For example, if the numbers are 1.75×10^2 and 6.8×10^4, then the number 1.75×10^2 is selected and converted to 0.0175×10^4.

Step 2: Set the exponent of the result equal to the larger exponent.

Step 3: Perform addition/subtraction on the mantissas and determine the sign of the result.

Step 4: Normalize the result, if necessary.

Let us perform the addition and subtraction of single precision numbers using above rules.

■ **Example : Add single precision floating point numbers A and B where**

A = 44900000 H and B = 42A00000H

Solution : Step 1 : Represent numbers in single precision format

A = 0 1000 1001 0010 000 0

B = 0 1000 0101 0100 000 0

Exponent for A = 1 0 0 0 1 0 0 1 = 37

∴ Actual exponent = 137 – 127 (Bias) = 10

Exponent for B = 1 0 0 0 0 1 0 1 = 133

∴ Actual exponent = 133 – 127 (Bias) = 6

∴ Number B has smaller exponent with difference 4. Hence its mantissa is shifted right by 4 bits as shown below.

Step 2 : Shift Mantissa

Shifted Mantissa of B = 0 0 0 0 0 1 0 0 0

Step 3 : Add Mantissas

```
Mantissa of      A = 0 0 1 0 0 0 0 0      .....0
Mantissa of      B = 0 0 0 0 0 1 0 0      .....0
                   ---------------
Mantissa of Result = 0 0 1 0 0 1 0 0      .....0
```

As both numbers are positive, sign of the result is positive.

∴ Result = 0 1 0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0

= 44920000 H

We have seen addition process for floating point numbers. Similar process is used for subtraction of floating point numbers. In subtraction, two mantissas are subtracted instead of addition and the sign of greater mantissa is assigned to the result.

■ ***Example : Subtract single point precision floating point numbers A and B (A – B), where A = 44900000 H and B = 42A00000 H.***

Solution : Step 1 : Represent numbers in single precision format

```
A = 0    1 0 0 0 1 0 0 1          0 0 1 0 0 0 0     ....0
```

```
B = 0    1 0 0 0 0 1 0 1          0 1 0 0 0 0 0     ....0
```

Exponent for A = 10001001 = 137

∴ Actual exponent = 137 – 127 (bias) = 10

Exponent for B = 1000 0101 = 133

∴ Actual exponent = 133 – 127 (Bias)

= 6

∴ Number B has smaller exponent with difference 4. Hence its mantissa is shifted right by 4 bits as shown below :

Step 2 : Shift Mantissa

Shifted Mantissa of B = 0 0 0 0 0 1 0 0 0

Step 3 : Subtract mantissa

```
Mantissa of A = 0 0 1 0 0 0 0 0 ....0
Mantissa of B = 0 0 0 0 0 1 0 0 ....0
                ---------------
                0 0 0 1 1 1 0 0 ....0
```

Mantissa for A is greater than mantisa for B therefore sign of result is sign of A.

∴ Result = 0 1 0 0 0 1 0 0 1   0 0 0 1 1 1 0 0 0

= 448E 0 0 0 0 H

Problems in Floating Point Arithmetic

Mantissa Overflow : The addition of two mantissas of the same sign may result in a carryout of the most significant bit. If so, the mantissa is shifted right and the exponent is incremented.

Mantissa Underflow : In the process of aligning mantissas, digits may flow off the right end of the mantissa. In such case trunction methods such as chopping, rounding are used.

Exponent Overflow : Exponent overflow occurs when a positive exponent exceeds the maximum possible exponent value. In some systems this may be designated as $+\infty$ or $-\infty$.

Exponent Underflow : Exponent underflow occurs when a negative exponent exceeds the maximum possible exponent value. In such cases, the number is designated as zero.

Flowchart and Algorthm for Floating Point Addition and Subtraction

Fig. shows flowchart for floating point addition and subtraction.

Algorithm : Referring the flowchart given in Fig. it is possible to write algorithm for floating point addition as follows.

```
Declare registers :  AM [0 : M –1]  ; For Mantissa of A
                     BM [0 : M –1]  ; For mantissa of B
                     AE [0 : E –1]  ; For exponent of A
                     BE [0 : E –1]  ; For exponent of B
                      E [0 : E –1]  ; For temporary storage of exponent
Step 1 : Read Numbers :      AM ← Mantissa of A
                             BM ← Mantissa of B
                             AE ← Exponent of A
                             BE ← Exponent of B
Step 2 : Compare the two exponents :
                             E = AE – BE
```

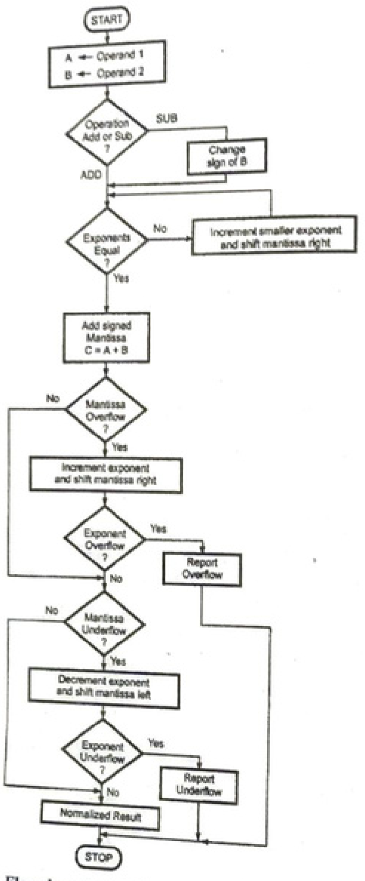


Fig. 30 : Flowchart for floating point addition and subtraction

```
Step 3 : Equalize
                              if E  < 0 then
                                   right shift AM;
                                   E = E + 1 and go to step 3;
                              if E  > 0 them
                                   right shift BM;
                                   E = E – 1 go to step 3;
Step 4 : Add Mantissas :      AM  = AM + BM and E = max (AE, BE)
Step 5 : Check exponent overflow :
    if overflow occurs during mantissa addition then
        if E = Emax then Report overflow
        else
            right shift AM
            E = E + 1 go to stop
Step 6 : Adjust for zero result
        If AM = 0 then E = 0; go to stop
Step 7 : Normalize result
        if AM normalized then go to stop
Step 8 : Check exponent underflow
        If E > Emin then
            left shift AM;
            E = E – 1 go to step 7
Step 9 : Stop
```

Implementing Floating-Point Operations

The Fig. shows the hardware implementation for the addition and subtraction of 32-bit floating point operands that have the single precision format, i.e. 1-bit for sign, 8-bits for signed exponent and 23-bits for mantissa. As per the rules of addition/subtraction of floating point number we have to first compare the exponents of numbers to determine a number with a smaller exponent and then to determine how for to shift the mantissa that number so that its exponent matches with the other number. The shift count value is determined by the 8-bit subtractor. The inputs for the 8-bit subtractor as exponent values of two number and the operation performed is $E_A - E_B$. The subtraction gives difference, n. This difference, is sent to the SHIFTER unit. The sign of the difference that results from comparing exponents determines which mantissa is to be shifted. If the sign as 0, then $E_A \geq E_B$ and input to SWAP network is 0. This disables swapping and mantissa M_B is sent to the SHFTER. If then sign is 1, the $E_A - E_B$ and input to SWAP network is 1. In this case swapping is enabled and mantissa M_A is sent to the SHIFTER. The SHIFTER unit shifts the given mantissa n positions to the right.

The two way multiplexer is used to set the exponent fo the result (E) equal to the larger exponent, i.e. step 5. Multiplexer has two inputs E_A and E_B and its output is based on the sign of the difference resulting from comparing exponents in step 1. The output of multiplexer is

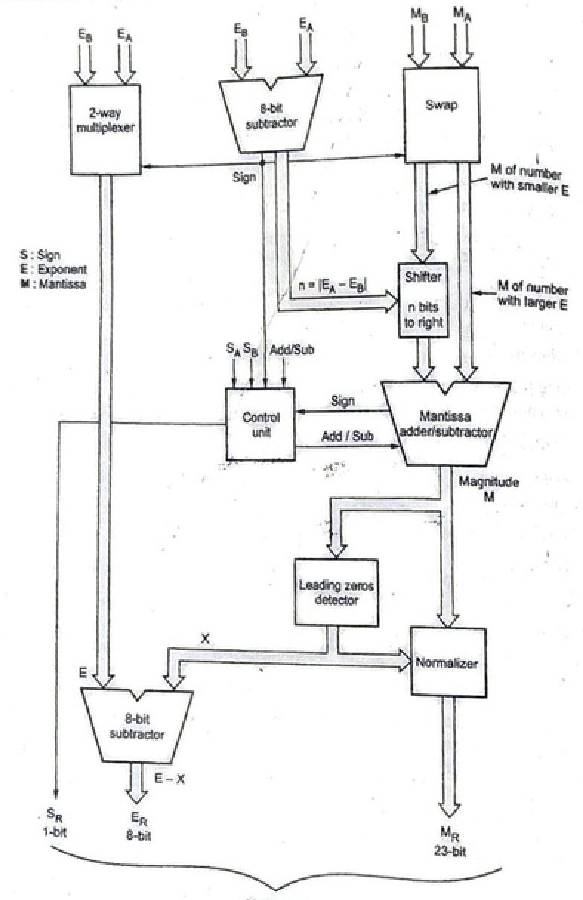


Fig. 31

$$E = E_A \quad \text{if } E_A \geq E_B$$

or

$$E = E_B \quad E_A < E_B$$

The third step is to perform addition or subtraction onthe mantissas and determine the sign of the result. The control logic is used to determine whether the mantissas are to be added or subtracted. It decides this by checking the signs of the operands (S_A and S_B) and the operation (Add or Subtract) that is to be performed on the operands. The control logic is also responsible for determining the sign of the result (S_R). The control logic determines the sign of the result by checking the resulted sign of mantissa adder/subtractor, sign from the exponent comparison, signs of the operands and the operation to be performed.

In step 4, the result of the mantissa (M) is normalized. The normalized value is truncated to generate the 23-bit mantissa, MR, of the result. The leadding zeros detector determines the number of bit shifts, X, to be applied to mantissa (M). The value X is then subtracted from the tentative resulted exponent E to generate the true result exponent E_B.

Multiplication and Division

In floating point arithmetic, multiplication and division are somewhat easier than addition and subtraction because an alignment of mantissas is not required in multiplication and division.

Let us see rules of multiplication.

1. Add the exponents and subtraction bias (127, in case of single precision numbers and 1023, in case of double precision numbers).
2. Multiply the mantissas and determine the sign of the result.
3. Normalize the result.
4. Divide the mantissas and determine the sign of the result.
5. Normalize the result.

Fig. shows the flowchart for floating point multiplication.

Let us see rules of division

1. Subtract exponents and add bias (127 in case of single precision numbers and 1023 in case double precision numbers).
2. Divide the mantissas and determine the sign of the result.
3. Normalize the result.

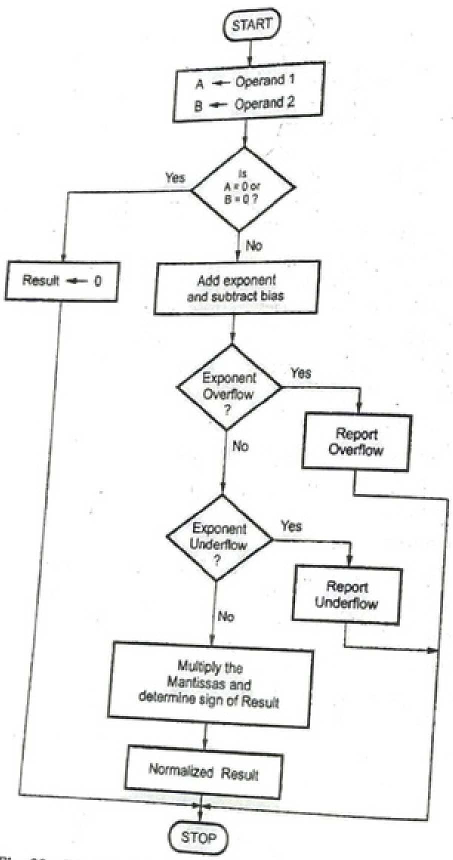


Fig. 32 : Flowchart for floating point multiplication

Fig. shows the flowchart for floating point division.

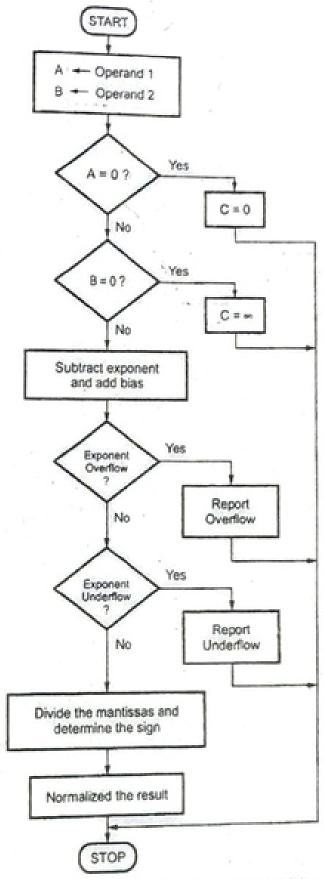


Fig. 33 : Flowchart for floating point division

2.8 Classification of Computers

General purpose computers come in many sizes and capabilities. Traditionally, computers were classification size, processing speed and cost. Based on these factors, computers were classified as microcomputers, minicomputers, mainframes and supercomputers. However, with the rapidly changing technology classification is no more relevant. The problem is that, computer technology is changing so fast that few months of introduction of a new computer in the market, new models of computers are introduction have much higher performance and cost less. Hence, a recently introduced small system can output from models of a few years ago, and a new minicomputer can do the work of an earlier mainframe, at a cost. Hence, today computers are classified based on their mode of use. According to this classification computers are classified as notebook computers, personal computers, workstations, mainframe supercomputers, and clients and serves. In this chapter, you will learn about these types of computers.

1. Work stations

Workstation is a powerful desktop computer, which is designed to meet the computing needs of engineers, and other professionals, who need greater processing power, larger storage, and better graphics display facility than what PCs provide. For example, workstations are commonly used for computer aided design (CAD), simulation of complex scientific and engineering problems, visualization of the results of simulation, and for multimedia applications, such as for creating special audio visual effects for television programmes and movies.

A workstation looks very much like PC, and is typically used by only one person at a time, just like a PC. The following characteristics are often used to differentiate between the two :

1. **Processing power :** The processing power of a workstation is several times (typically 5 to 10 times) more than that of an average power of a PC.
2. **Storage capacity :** Workstations have large main memory (Typically 0.5 GB to a few GB) as compared to PCs, which have few tens or hundreds of MB of main memory. The hard disk capacity of a workstation is also much more (typicaily severalties of GB) as compared to that of PCs (typically few GB).
3. **Display Facility :** Most workstations have a large screen (21 inch or more) monitor capable of displaying high resolution graphics. Hence, the color and graphics adapter card, which is optional for PCs, is available by default in workstations. PCs normally use monitors having smaller screen (19 inch or less).
4. **Processor design :** PCs normally use CPUs based on CISC technology, whereas workstations use CPUs based on RISC technology. Popular RISC processors used in workstations are ALPHa (used in DEC-AlPHA workstations), RIOS (used in IBM workstations), SPARC (used in SUN workstations), and PA-RISC (used in HP workstations).
5. **Operating system :** Unlike PCs which can run any of the five major OSs-MS-DOS, MS-Windows, Windows-NT, Linux and Unix, all workstations generally run the Unix operating system or a variation of it, such as AIX (used in IBM workstations), Solaris (used in SUN workstations), an HPUX (usd in HP workstations). Also, unlike most operating systems for PCs, which are single user oriented, workstation's operating system is designed to enable is to support a multi user environment.
6. **Network interface card :** Most workstations have built-in hardware to connect to a Local Area Network (LAN). This facility is normally optional in case of a PC.

Workstations generally cost form a few lakhs to few tens of lakhs of rupees, depending on the configuration. The biggest manufacturer of workstations is Sun microsystems. Other manufacturers of workstations include IBM, DEC, Hewlett Packard (HP), and Silicon Graphics.

2. Mainframe systems

There are several organizations, such as banks, insurance companies, hospitals, railways, etc., that need on line processing of large number of transactions, and require computer systems, which have massive data storage and processing capabilities. Mainframe systems are computer systems, which are mainly used for handling the information processing needs of such organizations. They are also used in such environments, in which a large number of users need to share a common computing facility, such as in research groups, educational institutions, engineering firms etc.

By its very nature of usage, a mainframe system is housed in a central location, with several user terminals connected to it. The user terminals act as access stations, and may or may not be located in the same building in which the mainframe system is located. A typical configuration of a mainframe system consists of the following components (See Figure 3).

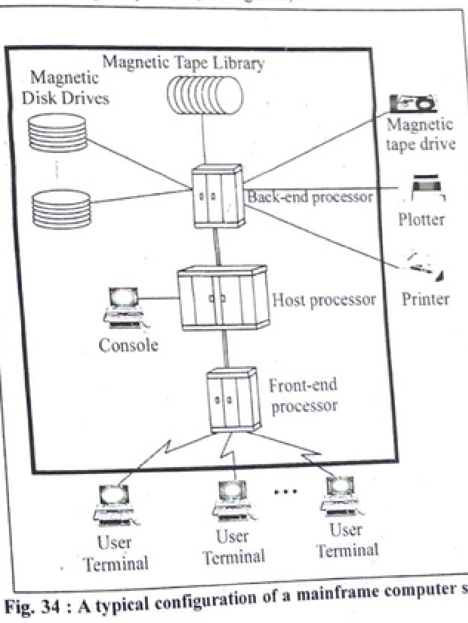


Fig. 34 : A typical configuration of a mainframe computer system

1. **Host, Front-end, and back end computers :** A mainframe computer system is usually composed of several computers (subordinate computers), in addition to the mainframe, or host computer. The host computer carries out most of the computations, and has direct control over all the other computers. The other computers relieve the host computer of certain routine processing requirements, For example, a front-end computer is used for handling communications to and from all the user terminals connected to the computer system, thus relieving the host computer of communications related processing requirements. Similarly, a back-end computer is used to handle data I/O operations, thus relieving the host computer of locating a particular I/O device and transferring data to or from it. As shown in the figure, the host and other computers are located in the systems room, to which entry is restricted to system administrators and maintenance staff.
2. **Console(s) :** One or more console terminals are also located in the system room. These terminals are directly connected to the host computer, and are mainly used by the system administrators to monitor the health of the system, or to perform some system administration activities, such as changing the configuration of the system, installing new software on the system, taking system backup, etc.
3. **Storage devices :** For large volume data storage, a mainframe system has several magnetic disk drives (located in the systems room), which are directly connected to the back end computer. All data to and from these magnetic disks, are accessed by the hot computer, via the back end computer. In addition, a mainframe system also has a few tape drives and a magnetic tape library, for restoration or backup of data, from or to magnetic tapes. The tape library is located in the systems room, and is used by the system administrators, to take regular backups of the data from magnetic disks on to magnetic tapes. The tape drives are located in the users room, so that users who bring their input data on tape or want to take their output data on tape can have access to the tape drives.
4. **User terminals :** User terminals, which act as access stations, are used by the users to work on the system. In the figure, although all the used terminals are shown to be located in the user room, some of them may be located at geographically distributed locations. Since mainframe systems allow multiple users to simultaneously used to system (through the user terminals), their operating systems support multiprogramming with timesharing. This enables all the users to get good response time, and an illusion that their jobs are being attended to by the system.
5. **Output devices :** The user terminals serve the purpose of soft copy output devices. However, for hard copy outputs, a mainframe system usually has several printers and one or more plotters, connected to the back end computer. These output devices are also located in the user room, so that they are accessible to the users, for taking their outputs.

It is unlikely that you would find two mainframe systems configured in exactly the same way. The configuration of a mainframe system depends a lot on the types of usage and the kind of users it is meant for. The example of Fig. 3 is just one possible configuration.

Mainframe systems are much bigger and several times more expensive than workstations. A

typical mainframe system looks like a row of large file cabinets, and needs a large room, with closely monitored humidity and temperature. A mainframe system may cost anywhere from a few tens of lakhs to a few crores of rupees, depending on the configuration. A mainframe system having smaller configuration (slower host and subordinate computers, lesser storage space, and fewer user terminals) is often referred to as a minicomputer system. However, there is no well-defined boundary for differentiating between the two. Two major vendors of mainframe systems are IBM and DEC.

3. SUPERCOMPUTERS

Supercomputers are the most powerful and expensive computers available at a given time. They are primarily used for processing complex scientific applications, which require enormous processing power. Some of the supercomputing applications (applications that need supercomputers for processing) are as follows :

1. Petroleum industry used supercomputers to analyze volumes of seismic data, which are gathered during oil-seeking explorations, to identify areas where there is possibility of getting petroleum products inside the earth. This helps in more detailed imaging of underground geological structures, so that the expensive resources for drilling oil wells and extraction of petroleum products from them can be more effectively channelized to those areas, where the analysis results show better possibility of getting petroleum deposits.
2. Aerospace industry used supercomputers to simulate airflow around an aircraft at different speeds and altitude. This helps in producing an effective aerodynamic design, to develop aircrafts with superior performance.
3. Automobile industry uses supercomputers to do crash simulation of the design of an automobile, before it is released for manufacturing. Doing crash simulation of an automobile on computer screen is less expensive more revealing, and safer than crashing a real model of the automobile. This helps in producing better automobile designs, which are safer to ride.
4. Structural mechanics industry used supercomputers to solve complex structure engineering problems, which designers of various types of civil and mechanical structures need to deal with to ensure safety, reliability, and cost effectiveness. For example, the designer of a large bridge has to ensure that the bridge must work in various atmospheric conditions and pressures from wind, velocity, etc. and under different load conditions. Actual construction and testing of such expensive structures is prohibitive in most cases.
5. Meteorological centers use supercomputers for weather forecasting. In weather forecasting, weather data supplied by a world wide network o space satellites, airplanes, and ground stations are fed into supercomputers, these data are analyzed by a series of computer programs to arrive at forecasts. This analysis involves solving of complex mathematical equations which model the atmosphere and climate processes.

There are many more supercomputing applications. It is not possible to cover all of them here. All these applications are impractical, if not impossible, on mainframe systems.

Super computers use multiprocessing and parallel processing technologies to solve complex problems faster. That is they used multiple processors, and parallel processing enables a complex problem to be divided into smaller problems, which can be assigned to different processors of the system and processed in parallel. A parallel program is written in a manner to break up the original problem into smaller computational modules, which can be allocated to different processors, and multiple processors can work independently and cooperate to solve the problem. Hence, if the original problem takes 100 hours to process on a single-processor system and if it can be broken into 100 smaller computational modules it can be theoretically solved in about 1 hour by using a supercomputer which has 100 processors. Since modern supercomputers employ parallel processing technology, they are also known as parallel computers or parallel processing systems. Moreover, modern supercomputers employ hundreds of processors, and are also known as massively parallel processors.

Like mainframe systems, supercomputers also support multiprogramming, which allows simultaneous access of the system by multiple users. However the main difference between these two types of systems are oriented to input/output bound applications. In processor-bound applications the amount of work that can be performed by the computer system is limited primarily by the speed of the CPU(s), whereas in I/O-bound applications, the amount of work that can be performed by the computer system is limited primarily by the speeds of the I/O devices. Supercomputers are rarely used for processing input/output intensive applications, such as payroll processing, accounting, or record keeping applications. To do so would waste an expensive and relatively rare resource.

Leaders in the development of supercomputers include Cray Research Company, IBM Silicon Graphics, Fujitsu, and Intel. Realizing the importance of supercomputer in the advancement of a nation, some national efforts of building supercomputers indigenously have also been initiated. This has resulted in the development of RARAM series of supercomputers by the Centre for Development of Advanced Computing (C-DAC), Anupam series of supercomputers by the Bhabha Atomic Research centre (BARC), and PACE series of supercomputers by the Defence Research and Development organization (DRDO), Hyderabad. The PARAM 1000 supercomputer (Shown in fig. 4) has been developed by C-DAC.

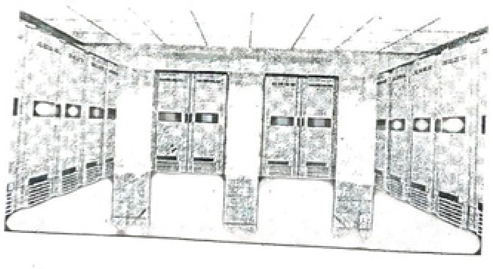


Fig. 35 : C-DAC's PARAM 10000 supercomputer [reproduced with permission from C-DAC]

Because computer technology changes so quickly, the advanced capabilities of a supercomputer today, may become the standard features next year, and next year's supercomputer will be vastly more powerful than today's Supercomputers can typically cost from few tens of lakhs to few tens of crores of rupees, depending on their processing capability and configuration.

4. CLIENTS AND SERVERS

This is purely role-based classification of computer systems. With the increased popularity of computer networks, it has become possible to interconnect several computers, which can communicate and interact with each other over the network. In such a computing environment, there are several resources/services, which can be shared among multiple users for cost effective usage, and can be best managed/offered centrally. A few examples of such resources/services are :

1. **File server :** It provides a central storage facility, to store files of several users on the network.
2. **Database server :** It manages a centralized database, and enables several users on the network to have shared access to the same database.
3. **Print server :** It manages one or more printers and accepts and processes print requests from any user in the network.
4. **Name server :** It translates names into network addresses, enabling different computers on the network to communicate with each other.

In these cases, it is usual to have one process, which "owns" the resource or service and is in charge of managing it. This process accepts requests from other processes, which want to use the resource or service. The process that owns the resource and does this management is called a server process, and the computer on which the ever process runs is called a server computer, because it services requests for use of the resource. Other processes, which send service requests to the server, are called client processes, and the computers on which the client processes run are called client computers. Note that there may be multiple client computers which send service requests to the same server computer. A generic client server computing environment is shown in Figure 5.

In a client server computing environment, it is common for one server to use the services of another server, and hence, to be both a client and a server at the same time. For example, let us assume that a client server computing environment has clients, a file server and a disk block server. Any client can send a file access request to the file server. On receiving such a request the file serve checks the access rights, etc. of the user, but does not actually read/write the file blocks itself. Instead, it sends a request to the disk block server for accessing the requested data block. The disk block server returns to the file server, the requested data blocks. The file server then extracts the actual data from the data blocks and returns it to the client. In this scenario, the file server is both a server and a client. It is a server for the clients but a client for the disk block server. Hence, the concept of client and server computers is purely role based, and may change dynamically, as the role of a computer changes.

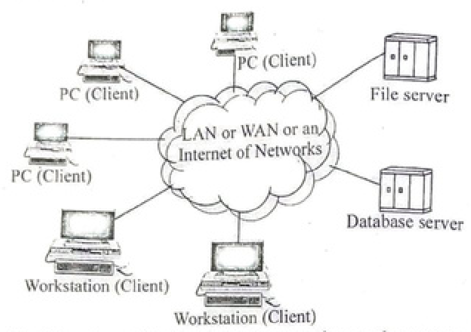


Fig. 36 : A generic client server computing environment.

Figure 6 : Summarizes and presents a relative comparison of different types of computers, based on some key features.

5. Notebook computers

Notebook computers are potable computers, which are mainly meant for use by people who need power wherever they go. As the name implies, notebook computers are approximately of the size of an inch notebook, and can easily fit inside a briefcase. Since they have to be carried along they are also weight, weighing around 2 Kg. They are also known as Laptop PCs (Laptop personal computers), because as powerful as a PC, and their size and weight allows them to be used by comfortably placing them on .

As shown in figure 1, a notebook computer uses an almost full-size keyboard, a small flat screen liquefaction color display, and a trackball (instead of a mouse, because notebook computers are often used without. They also have a hard disk, a floppy disk drive, and a CD-ROM drive. The display screen in foldable in that, when not in use, it can be folded to flush with the keyboard, to convert the system into notebook form in use, the display screen is folded open, as shown in the figure. Many models o notebook computers plugged into a "dock" on a docking station (a personal computer or a workstation), to take advantage of the machine's big monitor, storage space, and other peripherals such as a printer. Many models of computers can also be connected to a network, to enable them to download data (read files) from other computers on the network, as and when such a need arises, or to access the Internet. Since notebook computers are meant to be mobile and used from anywhere efforts are being made to provide wireless connectivity to these systems with other stationary computers.

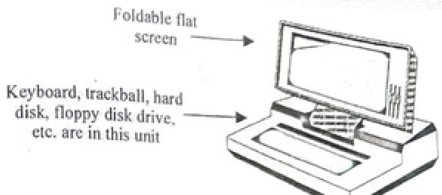

Foldable flat screen
Keyboard, trackball, hard disk, floppy disk drive, etc. are in this unit

Fig. 37 : A notebook computer

Notebook computers are designed to be used even at places where there is no power point available to connect them with a power source (for example while traveling in a train or aeroplane). Hence, they are designed to operate with chargeable batteries. With a fully charged battery, a notebook computer can be used for a few hours.

Notebook computers normally run MS-DOS or WINDOWS operating system. They are mostly used for word processing, spreadsheet computing, data entry, and preparing presentation materials, while a person is traveling. They are also used for making presentations, by plugging them into an LCD (Liquid crystal display) projection system, when presentations are to be made at a location away from ones office.

The processing capability of notebook computer is normally as good as an ordinary PC (personal computer) because both use the same type of processor, such as an Inter Pentium processor. However, a notebook computer generally has lesser hard disk storage than a PC, to keep its total weight to around 2 Kg. Notebook computers are typically more expensive (2 to 3 times) than a normal PC.

6. Personal Computer (PCs)

A PC is a non-portable, general-purpose computer, which can easily fit on a normal size office table (leaving some writing space to keep writing pads and other office stationary), and is generally designed to be used by one person at a time (single user-oriented). As the name implies, PCs were mainly designed to meat the personal computing needs of individuals, either is their working places or at their homes. In fact, PCs have changed the work culture and work habits of numerous organizations and individuals. An increasing proportion of office work now involves the used of computer. Hence, today PCs are found on the working desks of several employees, in any organization PCs are also providing employees with flexible working environments. Those employees, who could not work during traditional office hours due to personal reasons, can now work part of the time in the office and the remainder of the time at home, by having a PC in their homes. Several individuals also keep a PC in their homes to run a business in their homes. PCs are also used, both by children and adults, for education and entertainment. Hence, PCs are now very common everywhere, and can be found in offices, classrooms, homes, hospitals, shops, clinics etc.

The configuration of PCs varies from one PC to another, depending on their usage. However, the most commonly used configuration consists of a system unit, a monitor (display screen), a keyboard, and a mouse. The system unit, which is the form of a box, consists of the main circuit board (consisting of CPU, memory etc), the hard disk storage, the floppy disk drive, the CD-ROM drive any special add on cards (such as network interface card), and ports for connecting peripheral devices (such as printer).

The two most commonly used models of PCs are the desktop model and the tower model. As shown in Figure 2 in the desktop model, the monitor is positioned on top of the system unit, whereas is the tower model, the system unit is designed to stand by the side of the monitor. Hence in the tower model, the system unit can be positioned even on the floor beside or under the working desk, to preserve desk space. Although the desktop model was more popular few years ago, the tower model is gaining popularity now.

A PC generally employs several chips (CPU chip RAM chips, ROM chips, I/O handing chips, etc) on a main circuit board, called a system board, or motherboard. The motherboard is what distinguishes one PC from another. Often PCs are distinguished by the main component of the motherboard, that is the microprocessor chip, which is used as their CPU.

The popular operating systems for PCs are MS-DOS, MS-Windows, Windows-NT, Linux and Unix. An OS called OS/2 was also used on IBM PCs few years ago, and the Apple's Macintosh PCs run the Apple's propriety OS called macintosh OS and Apple's version of UNIx called A/UX. Most of these operating systems enable the used to switch between task. This capability is known as multitasking a single user variation of the multiprogramming concept. Multitasking eases user operation and saves lots of time, when a user has to switch between two or more applications, while performing a job. For example, let us assume that a user is using a word processor to create an annual report, and he/she needs to do some arithmetic calculations on the computer, and include the calculated results in the report. Without multitasking, the user would have to close the annual report file and the word processing application, open the calculator application, make the necessary calculations, write down the results, close the calculator application, and reopen the word processing application and the annual report file. With multitasking, the user simply opens the calculator application, makes the necessary calculations, and switches back to the annual report file, to continue working on it.

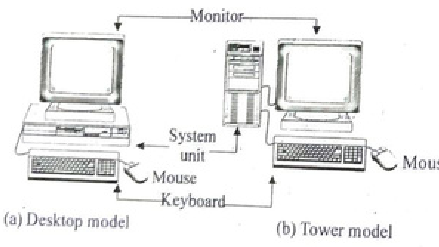


Fig. 38 : The two most commonly used models of PCs.

PCs generally cost from a few tens of thousands to about a lakh of rupees, depending on the configuration. A few of the major PC manufacturers are IBM, Apple, Compaq, Dell, Zenith, Siemens, Toshiba, and Hewlett Packard.

7. Tablet

A tablet computer or simply tablet is a mobile computer with display, circuitry and battery in a single unit. Tablest came equipped with sensors, including camerras, a microphone, an accelerometer and a touchscreen with finture or stylus gestures substituting for the use of computer mouse and keyboard. Tablet may include physical buttons (for example : to control basic features such as speaker volume and power) and ports (for network communications and to charge the battery). They usually feature on screen pop up virtual keyboard for typing.

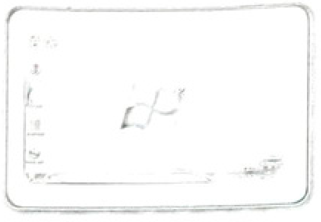


Fig. 39 : A tablet

Tablets are typically largr than smart phone or personal digital assistants at 7 inches (18 cm) or larger measured diagonally.

8. Palmtop Computer

A palmtop PC was on about pocket calculator sized battery powered PC in a horizontal clamshell design with integrated keyboad and display. It could be used like a sub notebook but was light enough to be comfotablyused handheld as well. Most palmtop PCs were small enough to be stored in a user's shit or jacket pocket.

Most palmtop PCs were based on a static hardware design for law power consumption and instant on/off without a need to reboot.

The first palmtop PC was the DIP pocket PC aka Atari Portfolio in 1989.

Fig. 40 : A palmtop computer

EXERCISES

Very Short Answer Type Questions (2 Marks each)

1. A CPU consists of .... (Raj. B.C.A. 2013)
2. RAM stands for .... (Raj. B.C.A. 2013)
3. Main memory stores .... (Raj. B.C.A. 2013)

4. USB stands for .... (Raj. B.C.A. 2011)
5. A storage device can be .... (Raj. B.C.A. 2010)

Short Answer Type Questions (4 Marks each)

1. Short note on SRAM and DRAM. (Raj. B.C.A. 2011)
2. What is instruction and execution cycle ? (Raj. B.C.A. 2012)
3. What do you mean by motherboard ? (Raj. B.C.A. 2012)
4. What is an IC ? (Raj. B.C.A. 2012)
5. Explain the primary and secondary memory. (Raj. B.C.A. 2011)

Long Answer Type Questions (12 Marks each)

1. What is an instruction ? Explain the types of instruction ? (Raj. B.C.A. 2012)
2. Explain the primary and secondary memory. (Raj. B.C.A. 2011)
3. What is virtual memory ? Write advantages of virtual memory ?(Raj. B.C.A. 2012, 2011)
4. Write short note on the following :
   (a) DMA
   (b) Instruction Cycle
   (c) Client server computer
5. What is register ? Define operation of control register briefly. (Raj. B.C.A. 2011)
6. What is cache memory ? Why is it called high speed memory ? Explain cache hit and cache mis.
7. Define the classification of computers ?
8. Discuss the various phases of Instruction cycle. (Raj. B.C.A. 2009)
9. Draw a block diagram to illustrate the basic organization of a computer system and explain the function of the logic units. (Raj. B.C.A. 2006)
10. Explain the execution cycle within CPU ? (Raj. B.C.A. 2006)

□□□