METHODIST COLLEGE OF ENGINEERING & TECHNOLOGY

Affiliated to Osmania University - College Code - 1607

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT-I

Basic Computer Organization: Functions of CPU, I/O Units, Memory: Instruction: Instruction Formats- One address, two addresses, zero addresses and three addresses and comparison; addressing modes with numeric examples: Program Control-Status bit conditions, conditional branch instructions, Program Interrupts: Types of Interrupts.

Basic Computer Organization

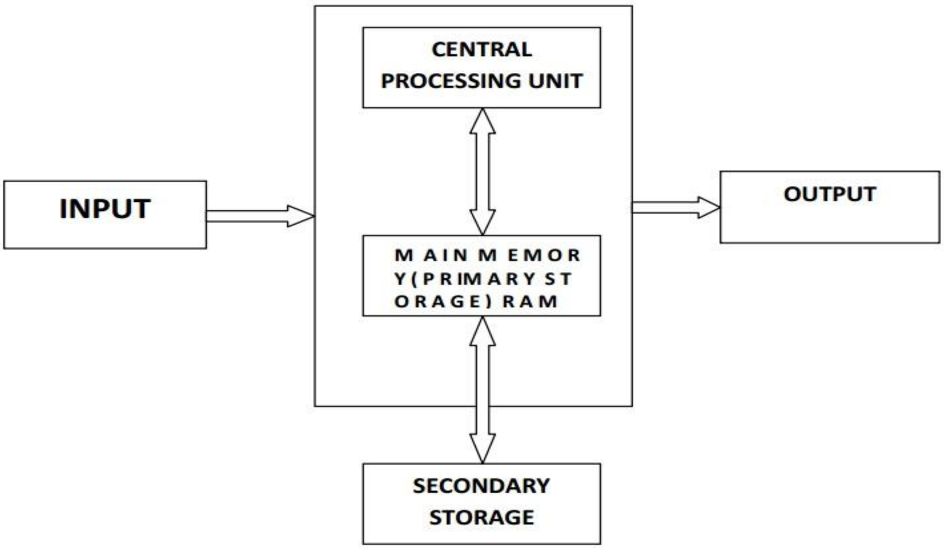


Fig: Block diagram of computer

A standard fully featured desktop configuration has basically four types of featured devices(I/O Units, Memory)

1. Input Devices 2. Output Devices 3. Memory 4. Storage Devices

1. Introduction to CPU
2. CPU
3. The Arithmetic / Logic Unit (ALU)
4. The Control Unit
5. Main Memory
6. External Memory
7. Input / Output Devices
8. The System Bus

CPU OPERATION

The fundamental operation of most CPUs

- To execute a sequence of stored instructions called a program.

1. The program is represented by a series of numbers that are kept in some kind of computer memory.

2. There are four steps that nearly all CPUs use in their operation: fetch, decode, execute, and write back.

3. Fetch:

o Retrieving an instruction from program memory.

o The location in program memory is determined by a program counter (PC)

o After an instruction is fetched, the PC is incremented by the length of the instruction word in terms of memory units.

Decode :

1.The instruction is broken up into parts that have significance to other portions of the CPU.

2.The way in which the numerical instruction value is interpreted is defined by the CPU's instruction set architecture (ISA).

3.Opcode, indicates which operation to perform.

4.The remaining parts of the number usually provide information required for that instruction, such as operands for an addition operation.

5.Such operands may be given as a constant value or as a place to locate a value: a register or a memory address, as determined by some addressing mode.

Execute :

1.During this step, various portions of the CPU are connected so they can perform the desired operation.

2.If, for instance, an addition operation was requested, an arithmetic logic unit (ALU) will be connected to a set of inputs and a set of outputs.

3.The inputs provide the numbers to be added, and the outputs will contain the final sum.

4. If the addition operation produces a result too large for the CPU to handle, an arithmetic overflow flag in a flags register may also be set.

Write back :

1.Simply "writes back" the results of the execute step to some form of memory.

2.Very often the results are written to some internal CPU register for quick access by subsequent instructions.

3.In other cases results may be written to slower, but cheaper and larger, main memory.

Some types of instructions manipulate the program counter rather than directly produce result data.

INPUT DEVICES

Anything that feeds the data into the computer. This data can be in alpha-numeric form which needs to be keyed-in or in its very basic natural form i.e. hear, smell, touch, see; taste & the sixth sense …feel?

Typical input devices are:

1.Keyboard
2. Mouse
3. Joystick
4. Digitizing Tablet
5. Touch Sensitive Screen
6. Light Pen
7. Space Mouse
8.Digital Stills Camera
9. Magnetic Ink Character
10.OpticalMarkReader
Recognition (MICR) (OMR)
11. Image Scanner
12.Bar Codes

13. Magnetic Reader
14.Smart Cards
15.Voice Data Entry
16.Sound Capture
17. Video Capture

The Keyboard is the standard data input and operator control device for a computer. It consists of the standard QWERTY layout with a numeric keypad and additional function keys for control purposes.

The Mouse is a popular input device. You move it across the desk and its movement is shown on the screen by a marker known as a 'cursor'. You will need to click the buttons at the top of the mouse to select an option.

Track ball looks like a mouse, as the roller is on the top with selection buttons on the side. It is also a pointing device used to move the cursor and works like a mouse. For moving the cursor in a particular direction, the user spins the ball in that direction. It is sometimes considered better than a mouse, because it requires little arm movement and less desktop space. It is generally used with Portable computers.

Magnetic Ink Character Recognition (MICR) is used to recognize the magnetically charged characters, mainly found on bank cheques. The magnetically charged characters are written by special ink called magnetic ink. MICR device reads the patterns of these characters and compares them with special patterns stored in memory. Using MICR device, a large volume of cheques can be processed in a day. MICR is widely used by the banking industry for the processing of cheques.

The joystick is a rotary lever. Similar to an aircraft's control stick, it enables you to move within the screen's environment, and is widely used in the computer games industry.

A Digitizing Tablet is a pointing device that facilitates the accurate input of drawings and designs. A drawing can be placed directly on the tablet, and the user traces outlines or inputs coordinate positions with a hand-held stylus.

A Touch Sensitive Screen is a pointing device that enables the user to interact with the computer by touching the screen. There are three types of Touch Screens: pressure-sensitive, capacitive surface and light beam.

A Light Pen is a pointing device shaped like a pen and is connected to a VDU. The tip of the light pen contains a light-sensitive element which, when placed against the screen, detects the light from the screen enabling the computer to identify the location of the pen on the screen. Light pens have the advantage of 'drawing' directly onto the screen, but this can become uncomfortable, and they are not as accurate as digitising tablets.

The Space mouse is different from a normal mouse as it has an X axis, a Y axis and a Z axis. It can be used for developing and moving around 3-D environments.

Digital Stills Cameras capture an image which is stored in memory within the camera. When the memory is full it can be erased and further images captured. The digital images can then be downloaded from the camera to a computer where they can be displayed, manipulated or printed.

The Optical Mark Reader (OMR) can read information in the form of numbers or letters and put it into the computer. The marks have to be precisely located as in multiple choice test papers.

Scanners allow information such as a photo or text to be input into a computer. Scanners are usually either A4 size (flatbed), or hand-held to scan a much smaller area. If text is to be scanned, you would use an Optical Character Recognition (OCR) program to recognise the printed text and then convert it to a digital text file that can be accessed using a computer.

A Bar Code is a pattern printed in lines of differing thickness. The system gives fast and error-free entry of information into the computer. You might have seen bar codes on goods in supermarkets, in libraries and on magazines. Bar codes provide a quick method of recording the sale of items.

Card Reader: This input device reads a magnetic strip on a card. Handy for security reasons, it provides quick identification of the card's owner. This method is used to run bank cash points or to provide quick identification of people entering buildings.

Smart Card: This input device stores data in a microprocessor embedded in the card. This allows information, which can be updated, to be stored on the card. This method is used in store cards which accumulate points for the purchaser, and to store phone numbers for cellular phones.

OUTPUT DEVICES :

Output devices display information in a way that you can you can understand. The most common output device is a monitor. It looks a lot a like a TV and houses the computer screen. The monitor allows you to 'see' what you and the computer are doing together.

Brief of Output Device

Output devices are pieces of equipment that are used to get information or any other response out from computer. These devices display information that has been held or generated within a computer. Output devices display information in a way that you can understand. The most common output device is a monitor.

Types of Output Device

Printing:	Plotter, Printer
Sound	: Speakers
Visual	: Monitor

A Printer is another common part of a computer system. It takes what you see on the computer screen and prints it on paper. There are two types of printers; Impact Printers and Non-Impact Printers.

Speakers are output devices that allow you to hear sound from your computer. Computer speakers are just like stereo speakers. There are usually two of them and they come in various sizes.

MEMORY OR PRIMARY STORAGE :

Purpose of Storage

The fundamental components of a general-purpose computer are arithmetic and logic unit, control circuitry, storage space, and input/output devices. If storage was removed, the device we had would be a simple calculator instead of a computer. The ability to store instructions that form a computer program, and the information that the instructions manipulate is what makes stored program architecture computers versatile.

Primary Storage

Primary storage is directly connected to the central processing unit of the computer. It must be present for the CPU to function correctly, just as in a biological analogy the lungs must be present (for oxygen storage) for the heart to function (to pump and oxygenate the blood). As shown in the diagram, primary storage typically consists of three kinds of storage:

Processors Register

It is the internal to the central processing unit. Registers contain information that the arithmetic and logic unit needs to carry out the current instruction. They are technically the fastest of all forms of computer storage.

Main memory

It contains the programs that are currently being run and the data the programs are operating on. The arithmetic and logic unit can very quickly transfer information between a processor register and locations in main storage, also known as a "memory addresses". In modern computers, electronic solid-state random access memory is used for main storage, and is directly connected to the CPU via a "memory bus" and a "data bus".

Cache memory

It is a special type of internal memory used by many central processing units to increase their performance or "throughput". Some of the information in the main memory is duplicated in the cache memory, which is slightly slower but of much greater capacity than the processor registers, and faster but much smaller than main memory.

Memory

Memory is often used as a shorter synonym for Random Access Memory (RAM). This kind of memory is located on one or more microchips that are physically close to the microprocessor in your computer. Most desktop and notebook computers sold today include at least 512 megabytes of RAM (which is really the minimum to be able to install an operating system). They are upgradeable, so you can add more when your computer runs really slowly.

The more RAM you have, the less frequently the computer has to access instructions and data from the more slowly accessed hard disk form of storage. Memory should be distinguished from storage, or the physical medium that holds the much larger amounts of data that won't fit into RAM and may not be immediately needed there.

Storage devices include hard disks, floppy disks, CDROMs, and tape backup systems. The terms auxiliary storage, auxiliary memory, and secondary memory have also been used for this kind of data repository.

RAM is temporary memory and is erased when you turn off your computer, so remember to save your work to a permanent form of storage space like those mentioned above before exiting programs or turning off your computer.

TYPES OF RAM:

There are two types of RAM used in PCs - Dynamic and Static RAM.

Dynamic RAM (DRAM): The information stored in Dynamic RAM has to be refreshed after every few milliseconds otherwise it will get erased. DRAM has higher storage capacity and is cheaper than Static RAM.

Static RAM (SRAM): The information stored in Static RAM need not be refreshed, but it remains stable as long as power supply is provided. SRAM is costlier but has higher speed than **DRAM.**

Additional kinds of integrated and quickly accessible memory are Read Only Memory (ROM), Programmable ROM (PROM), and Erasable Programmable ROM (EPROM). These are used to keep special programs and data, such as the BIOS, that need to be in your computer all the time. ROM is "built-in" computer memory containing data that normally can only be read, not written to (hence the name read only).

ROM contains the programming that allows your computer to be "booted up" or regenerated each time you turn it on. Unlike a computer's random access memory (RAM), the data in ROM is not lost when the computer power is turned off. The ROM is sustained by a small long life battery in your computer called the CMOS battery. If you ever do the hardware setup procedure with your computer, you effectively will be writing to ROM. It is non volatile, but not suited to storage of large quantities of data because it is expensive to produce. Typically, ROM must also be completely erased before it can be rewritten,

PROM (Programmable Read Only Memory)

A variation of the ROM chip is programmable read only memory. PROM can be programmed to record information using a facility known as PROM-programmer. However once the chip has been programmed the recorded information cannot be changed, i.e. the PROM becomes a ROM and the information can only be read.

EPROM (Erasable Programmable Read Only Memory)

As the name suggests the Erasable Programmable Read Only Memory, information can be erased and the chip programmed a new to record different information using a special PROM-Programmer. When EPROM is in use information can only be read and the information remains on the chip until it is erased.

STORAGE DEVICES

The purpose of storage in a computer is to hold data or information and get that data to the CPU as quickly as possible when it is needed. Computers use disks for storage: hard disks that are located inside the computer, and floppy or compact disks that are used externally.

Computers Method of storing data & information for long term basis i.e. even after PC is switched off.

It is non - volatile

Can be easily removed and moved & attached to some other device

Memory capacity can be extended to a greater extent

- Cheaper than primary memory

Storage Involves Two Processes

a) Writing data b) Reading data

Floppy Disks

The floppy disk drive (FDD) was invented at IBM by Alan Shugart in 1967. The first floppy drives used an 8-inch disk (later called a "diskette" as it got smaller), which evolved into the 5.25-inch disk that was used on the first IBM Personal Computer in August 1981. The 5.25-inch disk held 360 kilobytes compared to the 1.44 megabyte capacity of today's 3.5-inch diskette.

The 5.25-inch disks were dubbed "floppy" because the diskette packaging was a very flexible plastic envelope, unlike the rigid case used to hold today's 3.5-inch diskettes.

By the mid-1980s, the improved designs of the read/write heads, along with improvements in the magnetic recording media, led to the less-flexible, 3.5-inch, 1.44-megabyte (MB) capacity FDD in use today. For a few years, computers had both FDD sizes (3.5-inch and 5.25-inch). But by the mid-1990s, the 5.25-inch version had fallen out of popularity, partly because the diskette's recording surface could easily become contaminated by fingerprints through the open access area.

When you look at a floppy disk, you'll see a plastic case that measures 3 1/2 by 5 inches. Inside that case is a very thin piece of plastic that is coated with microscopic iron particles. This disk is much like the tape inside a video or audio cassette. Basically, a floppy disk drive reads and writes data to a small, circular piece of metal-coated plastic similar to audio cassette tape.

At one end of it is a small metal cover with a rectangular hole in it. That cover can be moved aside to show the flexible disk inside. But never touch the inner disk - you could damage the data that is stored on it. On one side of the floppy disk is a place for a label. On the other side is a silver circle with two holes in it. When the disk is inserted into the disk drive, the drive hooks into those holes to spin the circle. This causes the disk inside to spin at about 300 rpm! At the same time, the silver metal cover on the end is pushed aside so that the head in the disk drive can read and write to the disk.

Floppy disks are the smallest type of storage, holding only 1.44MB.

3.5-inch Diskettes (Floppy Disks) features:

Spin rate: app. 300 revolutions per minute (rpm)

High density (HD) disks more common today than older, double density (DD) disks

Storage Capacity of HD disks is 1.44 MB

Floppy Disk Drive Terminology

Floppy disk - Also called diskette. The common size is 3.5 inches.

Floppy disk drive - The electromechanical device that reads and writes floppy disks.

Track - Concentric ring of data on a side of a disk.

Sector - A subset of a track, similar to wedge or a slice of pie.

It consists of a read/write head and a motor rotating the disk at a high speed of about 300 rotations per minute. It can be fitted inside the cabinet of the computer and from outside, the slit where the disk is to be inserted, is visible. When the disk drive is closed after inserting the floppy inside, the monitor catches the disk through the Central of Disk hub, and then it starts rotating.

There are two read/write heads depending upon the floppy being one sided or two sided. The head consists of a read/write coil wound on a ring of magnetic material. During write operation, when the current passes in one direction, through the coil, the disk surface touching the head is magnetized in one direction. For reading the data, the procedure is reverse. I.e. the magnetized spots on the disk touching the read/write head induce the electronic pulses, which are sent to CPU.

The major parts of a FDD include:

Read/Write Heads: Located on both sides of a diskette, they move together on the same assembly. The heads are not directly opposite each other in an effort to prevent interaction between write operations on each of the two media surfaces. The same head is used for reading and writing, while a second, wider head is used for erasing a track just prior to it being written. This allows the data to be written on a wider "clean slate," without interfering with the analog data on an adjacent track.

Drive Motor: A very small spindle motor engages the metal hub at the center of the diskette, spinning it at either 300 or 360 rotations per minute (RPM).

Stepper Motor: This motor makes a precise number of stepped revolutions to move the read/write head assembly to the proper track position. The read/write head assembly is fastened to the stepper motor shaft.

Mechanical Frame: A system of levers that opens the little protective window on the diskette to allow the read/write heads to touch the dual-sided diskette media. An external button allows the diskette to be ejected, at which point the spring-loaded protective window on the diskette closes.

Circuit Board: Contains all of the electronics to handle the data read from or written to the diskette. It also controls the stepper-motor control circuits used to move the read/write heads to each track, as well as the movement of the read/write heads toward the diskette surface.

Electronic optics check for the presence of an opening in the lower corner of a 3.5-inch diskette (or a notch in the side of a 5.25-inch diskette) to see if the user wants to prevent data from being written on it.

Hard Disks

Your computer uses two types of memory: primary memory which is stored on chips located on the motherboard, and secondary memory that is stored in the hard drive. Primary memory holds all of the essential memory that tells your computer how to be a computer. Secondary memory holds the information that you store in the computer.

Inside the hard disk drive case you will find circular disks that are made from polished steel. On the disks, there are many tracks or cylinders. Within the hard drive, an electronic reading/writing device called the head passes back and forth over the cylinders, reading information from the disk or writing information to it. Hard drives spin at 3600 or more rpm (Revolutions Per Minute) - that means that in one minute, the hard drive spins around over 7200 times!

Optical Storage

Compact Disk Read-Only Memory (CD-ROM)

CD-Recordable (CD-R)/CD-Rewritable (CD-RW)

Digital Video Disk Read-Only Memory (DVD-ROM)

DVD Recordable (DVD-R/DVD Rewritable (DVD-RW)

Photo CD

Optical Storage Devices Data is stored on a reflective surface so it can be read by a beam of laser light. Two Kinds of Optical Storage Devices

CD-ROM (compact disk read-only memory)

DVD-ROM (digital video disk read-only memory)

Compact Disks

Instead of electromagnetism, CDs use pits (microscopic indentations) and lands (flat surfaces) to store information much the same way floppies and hard disks use magnetic and non-magnetic storage. Inside the CD-Rom is a laser that reflects light off of the surface of the disk to an electric eye. The pattern of reflected light (pit) and no reflected light (land) creates a code that represents data.

CDs usually store about 650MB. This is quite a bit more than the 1.44MB that a floppy disk stores. A DVD or Digital Video Disk holds even more information than a CD, because the DVD can store information on two levels, in smaller pits or sometimes on both sides.

Recordable Optical Technologies

CD-Recordable (CD-R)

CD-Rewritable (CD-RW)

PhotoCD

DVD-Recordable (DVD-R)

- DVD-RAM

CD ROM - Compact Disc Read Only Memory.

Unlike magnetic storage device which store data on multiple concentric tracks, all CD formats store data on one physical track, which spirals continuously from the center to the outer edge of the recording area. Data resides on the thin aluminum substrate immediately beneath the label. The data on the CD is recorded as a series of microscopic pits and lands physically embossed on an aluminum substrate. Optical drives use a low power laser to read data from those discs without physical contact between the head and the disc which contributes to the high reliability and permanence of storage device.

To write the data on a CD a higher power laser are used to record the data on a CD. It creates the pits and land on aluminum substrate. The data is stored permanently on the disc. These types of discs are called as WORM (Write Once Read Many). Data written to CD cannot subsequently be deleted or overwritten which can be classified as advantage or disadvantage depending upon the requirement of the user. However if the CD is partially filled then the more data can be added to it later on till it is full. CDs are usually cheap and cost effective in terms of storage capacity and transferring the data.

The CD‘s were further developed where the data could be deleted and re written. These types of CDs are called as CD Rewritable. These types of discs can be used by deleting the data and making the space for new data. These CD‘s can be written and rewritten at least 1000 times.

CD ROM Drive

CD ROM drives are so well standardized and have become so ubiquitous that many treat them as commodity items. Although CD ROM drives differ in reliability, which standards they support and numerous other respects, there are two important performance measures.

Data transfer rate

Average access

Data transfer rate: Data transfer rate means how fast the drive delivers sequential data to the interface. This rate is determined by drive rotation speed, and is rated by a number followed by ‗X‘. All the other things equal, a 32X drive delivers data twice the speed of a 16X drive. Fast data transfer rate is most important when the drive is used to transfer the large file or many sequential smaller files. For example: Gaming video.

CD ROM drive transfers the data at some integer multiple of this basic 150 KB/s 1X rate. Rather than designating drives by actual KB/s output drive manufacturers use a multiple of the standard 1X rate. For example: a 12X drive transfer data at (12*150KB/s) 1800 KB/s and so on.

The data on a CD is saved on tracks, which spirals from the center of the CD to outer edge. The portions of the tracks towards center are shorter than those towards the edge. Moving the data under the head at a constant rate requires spinning the disc faster as the head moves from the center where there is less data per revolution to the edge where there is more data. Hence the rotation rate of the disc changes as it progresses from inner to outer portions of the disc.

CD Writers

CD recordable and CD rewritable drives are collectively called as CD writers or CD burners. They are essentially CD ROM drives with one difference. They have a more powerful laser that, in addition to reading discs, can record data to special CD media.

Pen Drives / Flash Drives

- Pen Drives / Flash Drives are flash memory storage devices.
- They are faster, portable and have a capability of storing large data.
- It consists of a small printed circuit board with a LED encased in a robust plastic
- The male type connector is used to connect to the host PC
- They are also used a MP3 players

Functions of CPU:

The CPU, or Central Processing Unit, is both the heart and brains of every computer. Many of us do not know how important this unit is to the performance of a computer. How many of you have wondered about the basic functions of CPU? This article will answer that question, plus others including: Why it is important to have a good cooling system to keep the CPU at the right temperatures. Why it is so important to keep the CPU from overheating.

Common CPU Terms

Term	What it Means or Does
CPU	Central Processing Unit, or the heart and brains of the computer.
Binary Code	A sequence of ones and zeros, or the language into which your CPU translates all data.
ALU	Arithmetical Logical Unit, responsible for all mathematical and logical operations
Program Counter	Tracks which instructions the CPU should execute next when processing data.
One hertz	The speed at which one operation is performed per second.
Multi-core processor	A CPU with two or more independent cores, so it can do more than one thing at once.

The Four Primary Functions of the CPU

The CPU processes instructions it receives in the process of decoding data. In processing this data, the CPU performs four basic steps:

Fetch: Each instruction is stored in memory and has its own address. The processor takes this address number from the program counter, which is responsible for tracking which instructions the CPU should execute next.

Decode: All programs to be executed are translated to into Assembly instructions. Assembly code must be decoded into binary instructions, which are understandable to your CPU. This step is called decoding.

Execute: While executing instructions the CPU can do one of three things: Do calculations with its ALU, move data from one memory location to another, or jump to a different address.

Store: The CPU must give feedback after executing an instruction, and the output data is written to the memory.

INSTRUCTION FORMATS

The physical and logical structure of computers is normally described in reference manuals provided with the system. Such manuals explain the internal construction of the CPU, including the processor registers available and their logical capabilities. They list all hardware-implemented instructions, specify their binary code format, and provide a precise definition of each instruction. A computer will usually have a variety of instruction code formats. It is the function of the control unit within the CPU to interpret each instruction code and provide the necessary control functions needed to process the instruction.

The format of an instruction is usually depicted in a rectangular box symbolizing the bits of the instruction as they appear in memory words or in a control register. The bits of the instruction are divided into groups called fields. The most common fields found in instruction formats are:

1 An operation code field that specifies the operation to be performed.

An address field that designates a memory address or a processor registers.

A mode field that specifies the way the operand or the effective address is determined.

Other special fields are sometimes employed under certain circumstances, as for example a field that gives the number of shifts in a shift-type instruction.

The operation code field of an instruction is a group of bits that define various processor operations, such as add, subtract, complement, and shift. The bits that define the mode field of an instruction code specify a variety of alternatives for choosing the operands from the given address. The various addressing modes that have been formulated for digital computers are presented in Sec. 5.5. In this section we are concerned with the address field of an instruction format and consider the effect of including multiple address fields is an instruction.

Operations specified by computer instructions are executed on some data stored in memory or processor registers, Operands residing in processor registers are specified with a register address. A register address is a binary number of k bits that defines one of 2^k registers in the CPU. Thus a CPU with 16 processor registers R0 through R15 will have a
register address field of four bits. The binary number 0101, for example, will designate register R5.

Computers may have instructions of several different lengths containing varying number of addresses. The number of address fields in the instruction format of a computer depends on the internal organization of its registers. Most computers fall into one of three types of CPU organizations:

1. Single accumulator organization.
2. General register organization.
3. Stack organization.

An example of an accumulator-type organization is the basic computer presented in Chap. 5. All operations are performed with an implied accumulator register. The instruction format in this type of computer uses one address field. For example, the instruction that specifies an arithmetic addition is defined by an assembly language instruction as ADD.

Where X is the address of the operand. The ADD instruction in this case results in the operation AC ← AC + M[X]. AC is the accumulator register and M[X] symbolizes the memory word located at address X.

An example of a general register type of organization was presented in Fig. 7.1. The instruction format in this type of computer needs three register address fields. Thus the instruction for an arithmetic addition may be written in an assembly language as

ADD R1, R2, R3

To denote the operation R1 ← R2 + R3. The number of address fields in the instruction can be reduced from three to two if the destination register is the same as one of the source registers. Thus the instruction

ADD R1, R2

Would denote the operation R1 ← R1 + R2. Only register addresses for R1 and R2 need be specified in this instruction.

Computers with multiple processor registers use the move instruction with a mnemonic MOV to symbolize a transfer instruction. Thus the instruction

MOV R1, R2

Denotes the transfer R1 ← R2 (or R2 ← R1, depending on the particular computer). Thus transfer-type instructions need two address fields to specify the source and the destination.

General register-type computers employ two or three address fields in their instruction format. Each address field may specify a processor register or a memory word. An instruction symbolized by

ADD R1, X

Would specify the operation R1 ← R + M [X]. It has two address fields, one for register R1 and the other for the memory address X.

The stack-organized CPU was presented in Fig. 8-4. Computers with stack organization would have PUSH and POP instructions which require an address field. Thus the instruction

PUSH X

Will push the word at address X to the top of the stack. The stack pointer is updated automatically. Operation-type instructions do not need an address field in stack-organized computers. This is because the operation is performed on the two items that are on top of the stack. The instruction

ADD

In a stack computer consists of an operation code only with no address field. This operation has the effect of popping the two top numbers from the stack, adding the numbers, and pushing the sum into the stack. There is no need to specify operands with an address field since all operands are implied to be in the stack.

Most computers fall into one of the three types of organizations that have just been described. Some computers combine features from more than one organization structure. For example, the Intel 808-microprocessor has seven CPU registers, one of which is an accumulator register As a consequence; the processor has some of the characteristics of a general register type and some of the characteristics of a accumulator type. All arithmetic and logic instruction, as well as the load and store instructions, use the accumulator register, so these instructions have only one address field. On the other hand, instructions that transfer data among the seven processor registers have a format that contains two register address fields. Moreover, the Intel 8080 processor has a stack pointer and instructions to push and pop from a memory stack. The processor, however, does not have the zero-address-type instructions which are characteristic of a stack-organized CPU.

To illustrate the influence of the number of addresses on computer programs, we will evaluate the arithmetic statement X = (A + B) ∗ (C + D).

Using zero, one, two, or three address instruction. We will use the symbols ADD, SUB, MUL, and DIV for the four arithmetic operations; MOV for the transfer-type operation; and LOAD and STORE for transfers to and from memory and AC register. We will assume that the operands are in memory addresses A, B, C, and D, and the result must be stored in memory at address X.

THREE-ADDRESS INSTRUCTIONS

Computers with three-address instruction formats can use each address field to specify either a processor register or a memory operand. The program in assembly language that evaluates X = (A + B) ∗ (C + D) is shown below, together with comments that explain the register transfer operation of each instruction.

ADD R1, A, B
R1 ← M [A] + M [B]
ADD R2, C, D
R2 ← M [C] + M [D]
MUL X, R1, R2
M [X] ← R1 ∗R2

It is assumed that the computer has two processor registers, R1 and R2. The symbol M [A] denotes the operand at
memory address symbolized by A.

The advantage of the three-address format is that it results in short programs when evaluating arithmetic expressions. The disadvantage is that the binary-coded instructions require too many bits to specify three addresses. An example of a commercial computer that uses three-address instructions is the Cyber 170. The instruction formats in the Cyber computer are restricted to either three register address fields or two register address fields and one memory address field.

TWO-ADDRESS INSTRUCTIONS

Two address instructions are the most common in commercial computers. Here again each address field can specify either a processor register or a memory word. The program to evaluate X = (A + B) ∗ (C + D) is as follows:

MOV R1, A	**R1 ← M [A]**
ADD R1, B	**R1 ← R1 + M [B]**
MOV R2, C	**R2 ← M [C]**
ADD R2, D	**R2 ← R2 + M [D]**
MUL R1, R2	**R1 ← R1*R2**

```
MOV X, R1      M [X] ← R1
```

The MOV instruction moves or transfers the operands to and from memory and processor registers. The first symbol listed in an instruction is assumed to be both a source and the destination where the result of the operation is transferred.

ONE-ADDRESS INSTRUCTIONS

One-address instructions use an implied accumulator (AC) register for all data manipulation. For multiplication and division there is a need for a second register. However, here we will neglect the second and assume that the AC contains the result of tall operations. The program to evaluate X = (A + B) ∗ (C + D) is

```
LOAD   A     AC ← M [A]
ADD    B     AC ← A [C] + M [B]
STORE  T     M [T] ← AC
LOAD   C     AC ← M [C]
ADD    D     AC ← AC + M [D]
MUL    T     AC ← AC ∗ M [T]
STORE  X     M [X] ← AC
```

All operations are done between the AC register and a memory operand. T is the address of a temporary memory location required for storing the intermediate result.

ZERO-ADDRESS INSTRUCTIONS

A stack-organized computer does not use an address field for the instructions ADD and MUL. The PUSH and POP instructions, however, need an address field to specify the operand that communicates with the stack. The following program shows how X = (A + B) ∗ (C + D) will be written for a stack organized computer. (TOS stands for top of stack)

```
PUSH  A     TOS ← A
PUSH  B     TOS ← B
ADD         TOS ← (A + B)
PUSH  C     TOS ← C
PUSH  D     TOS ← D
ADD         TOS ← (C + D)
MUL         TOS ← (C + D) ∗ (A + B)
POP   X     M [X] ← TOS
```

To evaluate arithmetic expressions in a stack computer, it is necessary to convert the expression into reverse Polish notation. The name “zero-address” is given to this type of computer because of the absence of an address field in the computational instructions.

ADDRESSING MODES

The operation field of an instruction specifies the operation to be performed. This operation must be executed on some data stored in computer registers or memory words. The way the operands are chosen during program execution in dependent on the addressing mode of the instruction. The addressing mode of the instruction. The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually referenced. Computers use addressing mode techniques for the purpose of accommodating one or both of the following provisions:

To give programming versatility to the user by providing such facilities as pointers to Memory, counters for loop control, indexing of data, and program relocation

To reduce the number of bits in the addressing field of the instruction.

The availability of the addressing modes gives the experienced assembly language programmer flexibility for writing programs that are more efficient with respect to the number of instructions and execution time.

To understand the various addressing modes to be presented in this section, it is imperative that we understand the basic operation cycle of the computer. The control unit of a computer is designed to go through an instruction cycle that is divided into three major phases:

1. Fetch the instruction from memory
2. Decode the instruction.
3. Execute the instruction.

There is one register in the computer called the program counter of PC that keeps track of the instructions in the program stored in memory. PC holds the address of the instruction to be executed next and is incremented each time an instruction is fetched from memory. The decoding done in step 2 determines the operation to be performed, the addressing mode of the instruction and the location of the operands. The computer then executes the instruction and returns to step 1 to fetch the next instruction in sequence.

In some computers the addressing mode of the instruction is specified with a distinct binary code, just like the operation code is specified. Other computers use a single binary code that designates both the operation and the mode of the instruction. Instructions may be defined with a variety of addressing modes, and sometimes, two or more addressing modes are combined in one instruction.

An example of an instruction format with a distinct addressing mode field is shown in Fig. 1. The operation code specified the operation to be performed. The mode field is sued to locate the operands needed for the operation. There may or may not be an address field in the instruction. If there is an address field, it may designate a memory address or a processor register. Moreover, as discussed in the preceding section, the instruction may have more than one address field, and each address field may be associated with its own particular addressing mode.

Although most addressing modes modify the address field of the instruction, there are two modes that need no address field at all. These are the implied and immediate modes.

Implied Mode: In this mode the operands are specified implicitly in the definition of the instruction. For example, the instruction “complement accumulator” is an implied-mode instruction because the operand in the accumulator register is implied in the definition of the instruction. In fact, all register reference instructions that sue an accumulator are implied-mode instructions.

Op code	Mode	Address

Figure 1: Instruction format with mode field

Zero-address instructions in a stack-organized computer are implied-mode instructions since the operands are implied to be on top of the stack.

Immediate Mode: In this mode the operand is specified in the instruction itself. Inother words, an immediate- mode instruction has an operand field rather than an address field. The operand field contains the actual operand to be used in conjunction with the operation specified in the instruction. Immediate-mode instructions are useful for initializing registers to a constant value.

It was mentioned previously that the address field of an instruction may specify either a memory word or a processor register. When the address field specifies a processor register, the instruction is said to be in the register mode.

Register Mode: In this mode the operands are in registers that reside within the CPU .The particular register is selected from a register field in the instruction. A k-bit field can specify any one of 2^k registers.

Register Indirect Mode: In this mode the instruction specifies a register in the CPU whose contents give the address of the operand in memory. In other words, the selected register contains the address of the operand rather than the operand itself. Before using a register indirect mode instruction, the programmer must ensure that the memory address fo the operand is placed in the processor register with a previous instruction. A reference to the register is then equivalent to specifying a memory address. The advantage of a register indirect mode instruction is that the address field of the instruction sues fewer bits to select a register than would have been required to specify a memory address directly.

Auto increment or Auto decrement Mode: This is similar to the register indirect mode except that the register is incremented or decremented after (or before) its value is used to access memory. When the address stored in the register refers to a table of data in memory, it is necessary to increment or decrement the register after every access to the table. This can be achieved by using the increment or decrement instruction. However, because it is such a common requirement, some computers incorporate a special mode that automatically increments or decrements the content of the register after data access.

The address field of an instruction is used by the control unit in the CPU to obtain the operand from memory. Sometimes the value given in the address field is the address of the operand, but sometimes it is just an address from which the address of the operand is calculated. To differentiate among the various addressing modes it is necessary to distinguish between the address part of the instruction and the effective address used by the control when executing the instruction. The effective address is defined to be the memory address obtained from the computation dictated by the given addressing mode. The effective address is the address of the operand in a computational-type instruction. It is the address where control branches in response to a branch-type instruction. We have already defined two addressing modes in previous chapter.

Direct Address Mode: In this mode the effective address is equal to the address part of the instruction. The operand resides in memory and its address is given directly by the address field of the instruction. In a branch-type instruction the address field specifies the actual branch address.

Indirect Address Mode: In this mode the address field of the instruction gives the address where the effective address is stored in memory. Control fetches the instruction from memory and uses its address part to access memory again to read the effective address.

Relative Address Mode: In this mode the content of the program counter is added to the address part of the instruction in order to obtain the effective address. The address part of the instruction is usually a signed number (in 2's complement representation) which can be either positive or negative. When this number is added to the content of the program counter, the result produces an effective address whose position in memory is relative to the address of the next instruction. To clarify with an example, assume that the program counter contains the number 825 and the address part of the instruction contains the number 24. The instruction at location 825 is read from memory during the fetch phase and the program counter is then incremented by one to 826 + 24 = 850. This is 24 memory locations forward from the address of the next instruction. Relative addressing is often used with branch-type instructions when the branch address is in the area surrounding the instruction word itself. It results in a shorter address field in the instruction format since the relative address can be specified with a smaller number of bits compared to the number of bits required to designate the entire memory address.

Indexed Addressing Mode: In this mode the content of an index register is added to the address part of the instruction to obtain the effective address. The index register is a special CPU register that contains an index value. The address field of the instruction defines the beginning address of a data array in memory. Each operand in the array is stored in memory relative to the beginning address. The distance between the beginning address and the address of the operand is the index value stores in the index register. Any operand in the array can be accessed with the same instruction provided that the index register contains the correct index value. The index register can be incremented to facilitate access to consecutive operands. Note that if an index-type instruction does not include an address field in its format, the instruction converts to the register indirect mode of operation. Some computers dedicate one CPU register to function solely as an index register. This register is involved implicitly when the index-mode instruction is used. In computers with many processor registers, any one of the CPU registers can contain the index number. In such a case the register must be specified explicitly in a register field within the instruction format.

Base Register Addressing Mode: In this mode the content of a base register is added tothe address part of the instruction to obtain the effective address. This is similar to the indexed addressing mode except that the register is now called a base register instead of an index register. The difference between the two modes is in the way they are used rather than in the way that they are computed. An index register is assumed to hold an index number that is relative to the address part of the instruction. A base register is assumed to hold a base address and the address field of the instruction gives a displacement relative to this base address. The base register addressing mode is used in computers to facilitate the relocation of programs in memory. When programs and data are moved from one segment of memory to another, as required in multiprogramming systems, the address values of the base register requires updating to reflect the beginning of a new memory segment.

Numerical Example

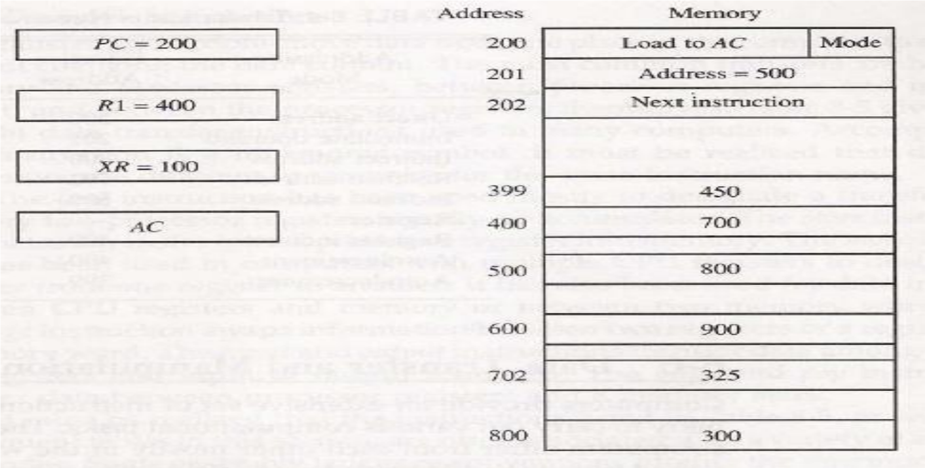
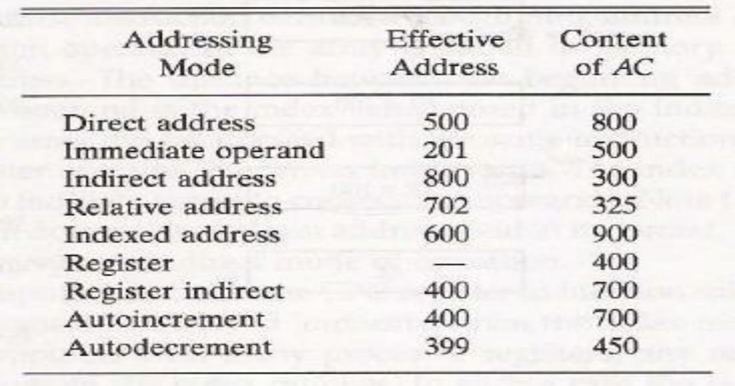


Figure 8-7 Numerical example for addressing modes.

TABLE 8-4 Tabular List of Numerical Example

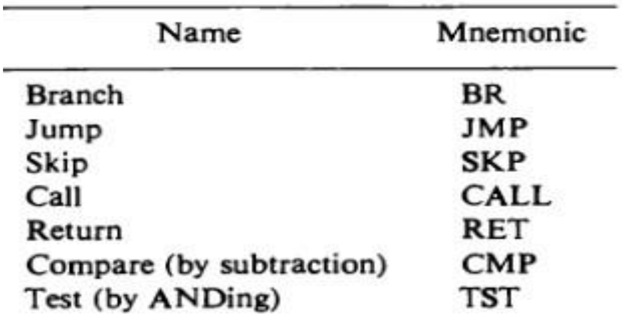


Addressing Mode	Effective Address	Content of AC
Direct address	500	800
Immediate operand	201	500
Indirect address	800	300
Relative address	702	325
Indexed address	600	900
Register	—	400
Register indirect	400	700
Autoincrement	400	700
Autodecrement	399	450

Program Control

Instructions are always stored in successive memory locations. When processed in the CPU, the instructions are fetched from consecutive memory locations and executed

Typical Program Control Instructions



Name	Mnemonic
Branch	BR
Jump	JMP
Skip	SKP
Call	CALL
Return	RET
Compare (by subtraction)	CMP
Test (by ANDing)	TST

Status Bit Conditions

It is sometimes convenient to supplement the ALU circuit in the CPU with a status register where status bit conditions can be stored for further analysis. Status bits are also called condition-code bits or flag bits.

The four status bits are symbolized by C, S, Z, and V. The bits are set or cleared as a result of an operation performed in the ALU.

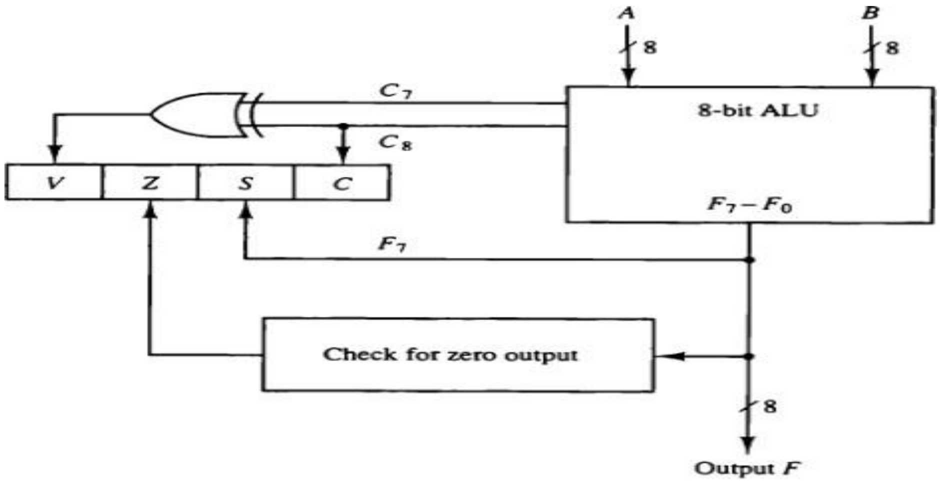
Bit C (carry) is set to 1 if the end cany C8 is 1. It is cleared to 0 if the canyisO.

Bit S (sign) is set to 1 if the highest-order bit F? is 1. It is set to 0 if the bit is 0.

Bit Z (zero) is set to 1 if the output of the ALU contains all 0's. It is cleared to 0 otherwise. In other words, Z = 1 if the output is zero and Z = 0 if the output is not zero.

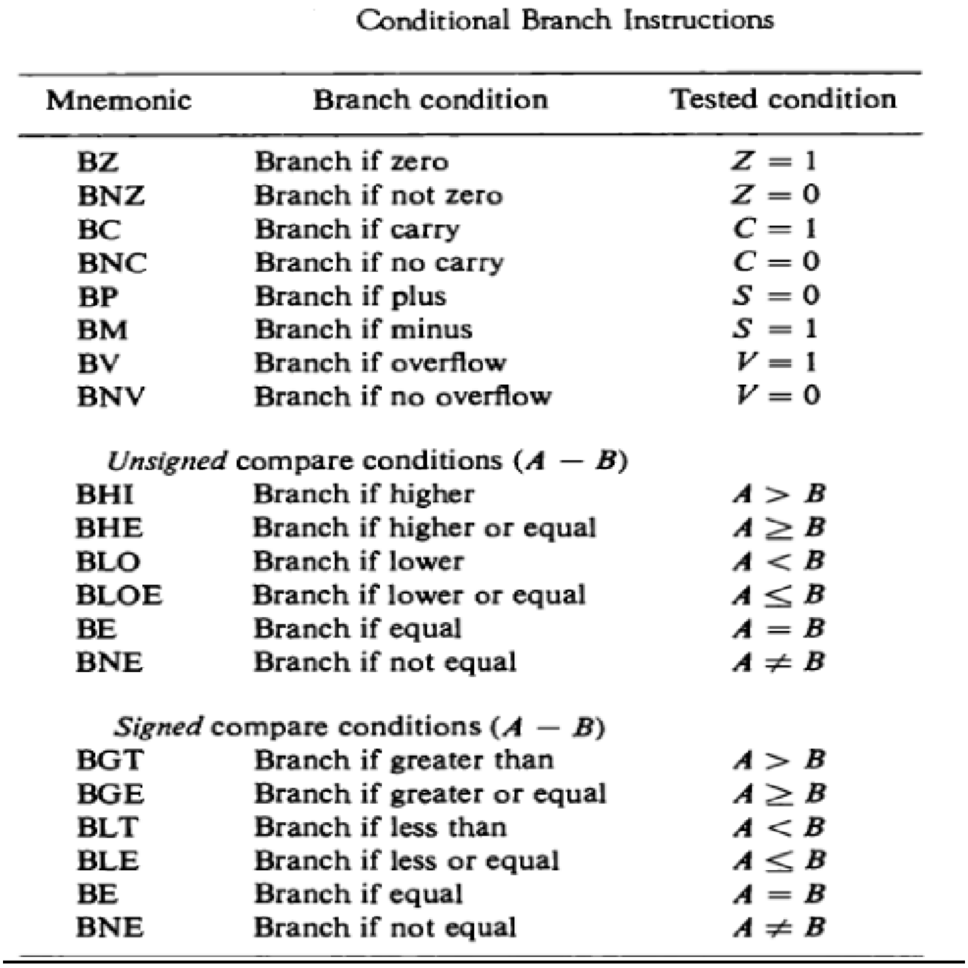
Bit V (overflow) is set to 1 if the exclusive-OR of the last two carries is equal to 1, and Cleared to 0 otherwise. This is the condition for an overflow when negative numbers are In 2's complement.

For the 8-bit ALU, V = 1 if the output is greater than +127 or less than -128.



Status register bits.

Conditional Branch Instructions



Conditional Branch Instructions

Mnemonic	Branch condition	Tested condition
BZ	Branch if zero	$Z = 1$
BNZ	Branch if not zero	$Z = 0$
BC	Branch if carry	$C = 1$
BNC	Branch if no carry	$C = 0$
BP	Branch if plus	$S = 0$
BM	Branch if minus	$S = 1$
BV	Branch if overflow	$V = 1$
BNV	Branch if no overflow	$V = 0$
Unsigned compare conditions ($A - B$)		
BHI	Branch if higher	$A > B$
BHE	Branch if higher or equal	$A \geq B$
BLO	Branch if lower	$A < B$
BLOE	Branch if lower or equal	$A \leq B$
BE	Branch if equal	$A = B$
BNE	Branch if not equal	$A \neq B$
Signed compare conditions ($A - B$)		
BGT	Branch if greater than	$A > B$
BGE	Branch if greater or equal	$A \geq B$
BLT	Branch if less than	$A < B$
BLE	Branch if less or equal	$A \leq B$
BE	Branch if equal	$A = B$
BNE	Branch if not equal	$A \neq B$

Some computers consider the C bit to be a borrow bit after a subtraction operation A — B. A Borrow does not occur if

A ^ B, but a bit must be borrowed from the next most significant Position if A < B. The condition for a borrow is the complement of the carry obtained when the subtraction is done by taking the 2's complement of B. For this reason, a processor that considers the C bit to be a borrow after a subtraction will complement the C bit after adding the 2's complement of the subtrahend and denote this bit a borrow.

Subroutine Call and Return

A subroutine is a self-contained sequence of instructions that performs a given computational task.
The instruction that transfers program control to a subroutine is known by different names. The most common names used are call subroutine, jump to subroutine, branch to subroutine, or branch and save address.
The instruction is executed by performing two operations:
The address of the next instruction available in the program counter (the return address) is Stored in a temporary location so the subroutine knows where to return
Control is transferred to the beginning of the subroutine.
Different computers use a different temporary location for storing the return address.
- Some store the return address in the first memory location of the subroutine, some store it in a fixed location in memory, some store it in a processor register, and some store it in a memory stack. The most efficient way is to store the return address in a memory stack. The advantage of using a stack for the return address is that when a succession of subroutines is called, the sequential return addresses can be pushed into the stack. The return from subroutine instruction causes the stack to pop and the contents of the top of the stack are transferred to the program counter.
A subroutine call is implemented with the following micro operations:
SP ←SP - 1 Decrement stack pointer
M [SP] ←PC Push content of PC onto the stack
PC ← effective address Transfer control to the subroutine
If another subroutine is called by the current subroutine, the new return address is pushed into The stack and so on. The instruction that returns from the last subroutine is implemented by the Micro operations:
PC←M [SP] Pop stack and transfer to PC SP ←SP + 1 Increment stack pointer

Program Interrupt

Program interrupt refers to the transfer of program control from a currently running program to another service program as a result of an external or internal generated request. Control returns to the original program after the service program is executed.The interrupt procedure is, in principle, quite similar to a subroutine call except for three Variations:
The interrupt is usually initiated by an internal or external signal rather than from the Execution of an instruction (except for software interrupt as explained later);
The address of the interrupt service program is determined by the hardware rather than from the address field of an instruction.
An interrupt procedure usually stores all the information
The state of the CPU at the end of the execute cycle (when the interrupt is recognized) is determined From:
The content of the program counter
The content of all processor registers
The content of certain status conditions

- Program status word the collection of all status bit conditions in the CPU is sometimes called a program status word or PSW. The PSW is stored in a separate hardware register and contains the status information that characterizes the state of the CPU.

Types of Interrupts

There are three major types of interrupts that cause a break in the normal execution of a Program. They can be classified as:

1. External interrupts
2. Internal interrupts
3. Software interrupts

External interrupts come from input-output (I/O) devices, from a timing device, from a circuit monitoring the power supply, or from any other external source.

Internal interrupts arise from illegal or erroneous use of an instruction or data. Internal interrupts are also called traps. Examples of interrupts caused by internal error conditions are register overflow, attempt to divide by zero, an invalid operation code, stack overflow, and protection violation.

A software interrupt is initiated by executing an instruction. Software interrupt is a special call instruction that behaves like an interrupt rather than a subroutine call. It can be used by the programmer to initiate an interrupt procedure at any desired point in the program.

****