Number Systems ,Base Conversions:-

1- Decimal Numbers.
2- Binary Numbers.
3- Octal Numbers.
4- Hexadecimal Numbers.

1. ***Decimal Numbers***: - Decimal number system contains 10 digits: 0,1,2,3,4,5,6,7,8,9; and that is why its base is 10.
   Decimal weight … …10^4 10^3 10^2 10^1 10^0. 10^{-1} 10^{-2} 10^{-3} ….

   Example (1):$(345)_{10}$
   $300+40+5=10^2*3+10^1*4+10^0*5=345=(345)_{10}$
   ↓ ↓ ↓
   3 4 5

   Example (2):$(234)_{10}$
   $200+30+4=10^2*2+10^1*3+10^0*4=234=(234)_{10}$
   ↓ ↓ ↓
   2 3 4

   Example (3): **$23.5 = (23.5)_{10}$**
   $10^1*2 + 10^0*3 + 10^{-1}*5 = 20+3+0.5=23.5$
   Where $10^0=1$

   Example (4): **$45.5 = (45.5)_{10}$**
   $10^1*4 + 10^0*5 + 10^{-1}*5 = 40+5+0.5=45.5$

2. ***Binary Numbers:***The binary number system its two digits a base-two system. The two binary digits (bits) are 1 and 0 (1, 0).

   Binary weight : ….. 2^3 2^2 $2^1 2^0 . 2^{-1}$ 2^{-2} 2^{-3} …..
   Weight value ….. 84 21 . 0.5 0.25 0.125 …..

   A. Binary – to – Decimal Conversion:
   *where **$2^0=1$**

   Example (5):**1 1 0 1 1 0 1**
   1 1 0 1 1 0 1
   $2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0 = 2^6*1+2^5*1+2^4*0+2^3*1+2^2*1+2^1*0+2^0*1$
   $= 64*1+32*1+16*0+8*1+4*1+2*0+1*1$
   $=64+32+0+8+4+0+1 \quad 109 \Rightarrow (109)_{10}$

Example (6):

10 0 1

$2^3\ 2^2 2^1\ 2^0 = 2^3*1 + 2^2*0 + 2^1*0 + 2^0*1$

$= 8*1 + 4*0 + 2*0 + 1*1$

$(9) \Rightarrow (9)_{10}\ 8+0+0+1$

***The Fractional binary number 0.1011**

Example (7):

0.1 0 1 1

$2^{-1}\ 2^{-2}\ 2^{-3}\ 2^{-4} = 1*2^{-1} + 0*2^{-2} + 1*2^{-3} + 1*2^{-4}$

$= 0.5 + 0 + 0.125 + 0.0625 \rightarrow (0.6875)_1$

Example (8):

0.1 1 0 0

$2^{-1}\ 2^{-2}\ 2^{-3}\ 2^{-4} = 1*2^{-1} + 1*2^{-2} + 0*2^{-3} + 0*2^{-4}$

$= 0.5 + 0.25 + 0 + 0 \rightarrow (0.75)_1$

B. <u>Decimal – to – Binary Conversion:</u>

1- Convert a decimal number to binary using the repeated division -by-2. This method repeatedly divides a decimal number by 2 and records the quotient and remainder

2- Convert a decimal fraction to binary using the repeated Multiplication – by – 2 methods.

Example (9):

Number $(58)_{10}$ ==== $(111010)_2$

58	**2**		**mod**	**LSB**
29	**2**	→	**0**	↑
14	**2**	→	**1**	
7	**2**	→	**0**	
3	**2**	→	**1**	
1	**2**	→	**1**	
0		→	**1**	
				MSB

$\Rightarrow$ **$(111010)_2$**

Example (10):
Number $(0.3125)_{10}$ ======$(0101)_2$

0.3125 *2 | carry MSB
0.6250*2 | $\rightarrow$ 0
0.2500*2 | $\rightarrow$ 1 $\Longrightarrow$ $(0101)_2$
0.5000*2 | $\rightarrow$ 0
0.0000 | $\rightarrow$ 1 $\downarrow$
LSB

3. ***Octal Numbers:*** The octal number system is composed of eight digits, which are 0, 1, 2, 3, 4, 5, 6, and 7.
   To count above 7, begin another column and start over:
   10, 11, 12, 13, 14, 15, 16, and 17.
   20, 21, 22, 23, 24, 25, 26, and 27.
30, 31, … … … … …37.

   Octal Weight … … $8^3\ 8^2\ 8^1\ 8^0.8^{-1}\ 8^{-2}\ 8^{-3}$…….

A. Octal – to – Decimal conversion:
Example (11):

Octal number $(2374)_8$ ==== $(1276)_{10}$

$(2374)_8 = 8^3 * 2+8^2 *3+8^1*7+*8^0* 4$
$= 512*2+64*3+8*7+1*4$
$= 1024+192+56+4$
$= (1276)_{10}$

B.Decimal – to – Octal Conversion:
Example (12):
Decimal number $(359)_{10}$ =======$(547)_8$

359 | 8 **mod LSB**
44 | 8 $\rightarrow$ 7 $\uparrow$
5 | 8 $\rightarrow$ 4 $\Longrightarrow$ $(547)_8$
0 | $\rightarrow$ 5
MSB

C. Octal – to – Binary Conversion:
Octal digit can be represented by a 3-bit binary number.
Octal digit binary
0 1 2 3 4 5 6 7

000 001 010 011 100 101 110 111

***Example* (13):**

$(25)_8$ $(2\quad 5)_8$

$(010\quad 101)_2$

***Example* (14):**

$(140)_8$ $(1\quad 4\quad 0)_8$

$(001100\ 000)_2$

D. <u>Binary – to – Octal Conversion:</u>

Conversion binary number to octal number is start with right – most groups of three bits and moving from right to left.

***Example* (15):**

$$\mathbf{(110101)_2}$$

$$\begin{array}{cc} \underline{\mathbf{110}} & \underline{\mathbf{101}} \\ | & | \\ \mathbf{6} & \mathbf{5} \end{array} \quad \mathbf{(6\ 5)_8}$$

***Examples* (16):**

$$\mathbf{(101111001)_2}$$

$$\begin{array}{ccc} \underline{\mathbf{101}} & \underline{\mathbf{111}} & \underline{\mathbf{001}} \\ | & | & | \\ \mathbf{5} & \mathbf{7} & \mathbf{1} \end{array} \quad \mathbf{(5\ 7\ 1)_8}$$

4.<u>Hexadecimal Numbers:</u> The hexadecimal number system has a base of sixteen; it is composed of 16 digits and alphabetic characters.

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

A. Binary – to – Hexadecimal conversion:
4-bit groups, starting at the right-most bit.

Example(17): **$(1100101001010111)_2$ =======$(CA57)_{16}$**

1100 1010 0101 0111

C A 5 7

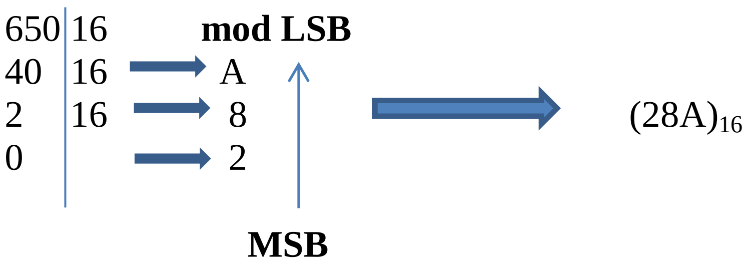
B. Hexadecimal – to – Binary Conversion:
***Example (18)*: $(10A4)_{16}$=========$(1000010100100)_2$**

1 0 A 4
0001 0000 1010 0100

C.*Hexadecimal – to –Decimal Conversion:*By two method
*** First method:**
***Example (19)*:** $(A85)_{16}$====$(2693)_{10}$
1- Convert to binary number.
2- Convert from binary number to decimal number.
A 8 5
1010 1000 0101 =
$2^{11}*1+2^{10}*0+2^9*1+2^8*0+2^7*1+2^6*0+2^5*0+2^4*0+2^3*0+2^2*1+2^1*0+2^0*1=$
$2^{11}+2^9+2^7+2^2+2^0=2048+512+128+4+1=2693= (2693)_{10}$

*** Second method:**
***Example(20)*:** $(E5)_{16}$ ========$(229)_{10}$
$(E5)_{16}=16^1*E+16^0*5$
$16*14+1*5 =224+5=229= (229)_{10}$

D.*Decimal – to – Hexadecimal Conversion:*
Convert the decimal number 650 to hexadecimal by repeated division by 16.
***Example(21)* :**
$(650)_{10}$=====$(28A)_{16}$



Microcomputer Architecture

The word ***computer*** comes from the word ***(compute)*** the word compute means to ***(calculate)*** or to ***(count)***, computer is an electronic device that manipulates information or ***(data).*** It has ability to store, retrieval, and process data.

Advantages of computer system: -
1- Store and retrieve large quantities of data.
2-The speed is faster than in any other form of data processing.
3-A single computer can perform a wide variety of activities as directed by a set of instructions (program).
4-Once data and instructions are fed into the computer, processing is continuous with a minimum of human intervention.
5-Data and programs may be stored inside the computer indefinite and be retrieved quickly.
6- Accuracy is greater than any other system.

***A* computer system has three main components**: a **Central Processing Unit** (CPU) or processor, a **Memory Unit** and **Input/output Units (**devices). In any microcomputer system, the component which actually processes data is entirely contained on a single chip called M**icroprocessor** (MPU). This MPU can be programmed using assembly language.
The main **internal hardware** features of a computer are the **processor**, **memory** and **registers** (registers are special processor components for holding address and data).

The **external hardware** features are the computer Input/Output devices such as keyboard, monitor…

Software consists of the operating system (O.S) and various programs and data files stored on disk.

Personal Computer (PC) Components:-
The main component of the PC is **System Board** (or motherboard). It contains the processor, main memory, connectors, and expansion slots for optional cards. The slots and connectors provide access to such components as ROM, RAM, hard disk, CD-ROM drive, additional memory, video unit, keyboard, mouse, parallel and serial device, sound adapter and cache memory (the processor use high speed cache memory to decrease its need to access the slower main memory). A bus with wires attached to the system board connects the components. It transfers data between the processor, memory and external devices.

A. The processor

The CPU or processor acts as the controller of all actions or services provided by the system. The operations of a CPU can be reduced to three basic steps: **fetch**, **decode**, and **execute**. Each step includes intermediate steps, some of which are:

1- Fetch the next instruction:
- Place it in a holding area called a queue.
- Decode the instruction.

2- Decode the instruction
- Perform address translation.
- Fetch operand from memory.

3- Execute the instruction.
- Perform the required calculation.
- Store results in memory or register.
- Set status flag attached to the CPU.

System Bus:-

The components of the computer system must communicate witheach other and with the outside world. Although it may be possibleto connect each component to the CPU separately as a practicalmatter this would require too many physical connects. To keep thenumber of connections manageable, the processor is connected tomemory and all peripherals using a bus.

A Busis a bunch of wires, and electrical path on the printed IC to which everything in the system is connected.

There are three types of Bus:

1- Address Bus (AB): the width of AB determines the amountof physical memory addressable by the processor.

2- Data Bus (DB): the width of DB indicates the size of the data transferred between the processor and memory or I/O device.

3- Control Bus (CB): consists of a set of control signals, typical control signals includes memory read, memory write, I/O read, I/O write, interrupt acknowledge, bus request.

These control signals indicates the type of action taking place on the system bus.

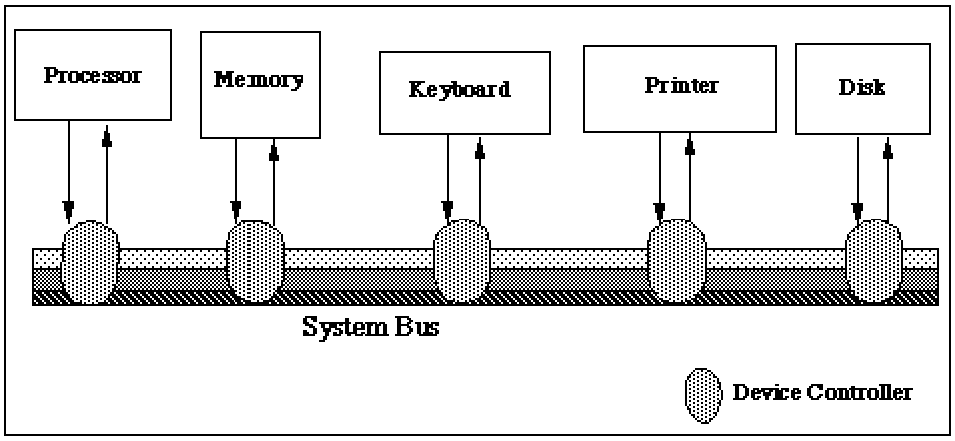


***Figure (1) The system bus**:*
the processor communicates with all devices via the system bus

The CPU is divided into two general parts. **Arithmetic Logic Unit** (ALU) and **Control Unit** (CU).
- The **ALU** carry Arithmetic, logical, and shifting operations.
- The **CU** fetches data and instruction, and decodes addresses for the ALU.

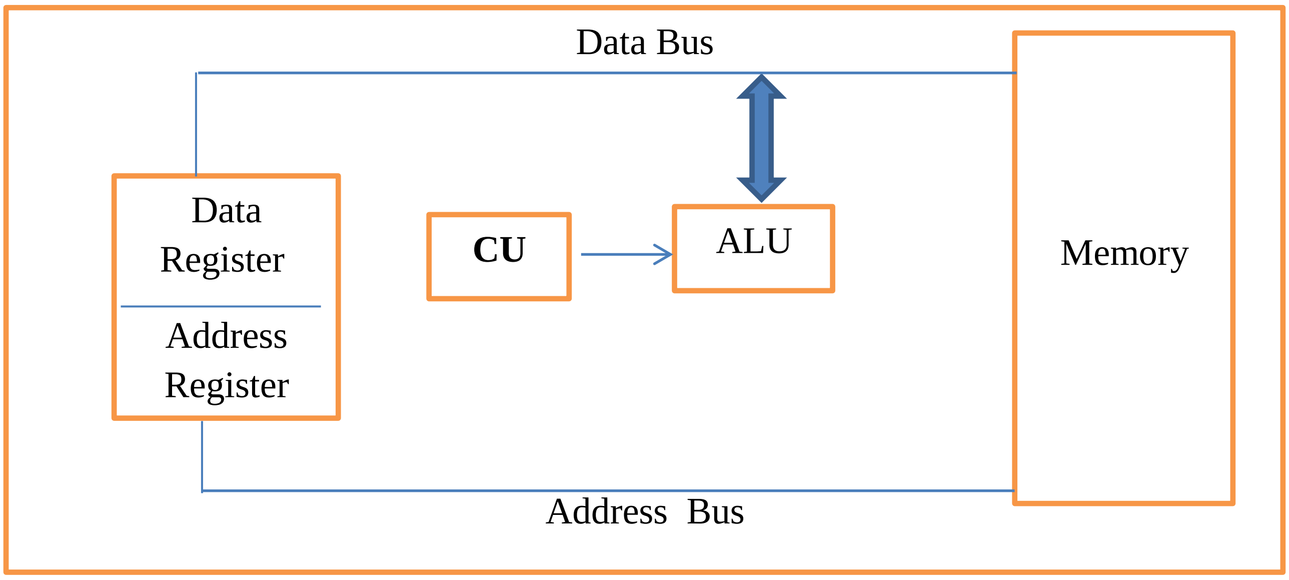


***Figure* (2): A Block Diagram of the Simple CPU**

Register:-Registers are devices capable of storing information, receiving data from other areas within the computer and transferring information as directed by the control unit, it is used for temporary storage of data or instruction and the most important register are :
- **Program counter** (PC): hold address of the next instruction
-**Instruction register** (IR): Hold instruction while it is decoded and Executed

- Address register (AR): holds the address of memory location.

-**PSW** (Processor Status Word): collection of bits called Flags or Condition Codes. They typical used to indicate a Zero result, a Negative result, a Carry, an Overflow and so on.

-SP (Stack Pointer): Stack may consist of a set of internal registers or a portion of main memory. It is used for temporarily storing important information while subroutines are being executed. The top of the stack is the last information put onto the stack.

The instruction is brought in from the memory and placed in the IR. The Control Unit then decodes the instruction direct its execution. At the same time the CU sets the PC/IP to the address of the next instruction.

B- Memory (Main Memory)

The memory of a computer system consist of tiny electronic switches, with each switch set in one of two states: open or close.

It is however more convenient to think of these states **as 0 and 1**.Thus each switch can represent a binary digit or bit, as it is known, the memory unit consists of millions of such bits, bits are organized into groups of eight bits called **byte**. Memory can be viewed as consisting of an ordered sequence of bytes. Each byte in this memory can be identified by its sequence number starting with 0, as shown in Figure 3. This is referred to as memory address of the byte. Such memory is called **byte addressable memory.** The memory address space of a system is determined by the address bus width of the CPU used in the system.

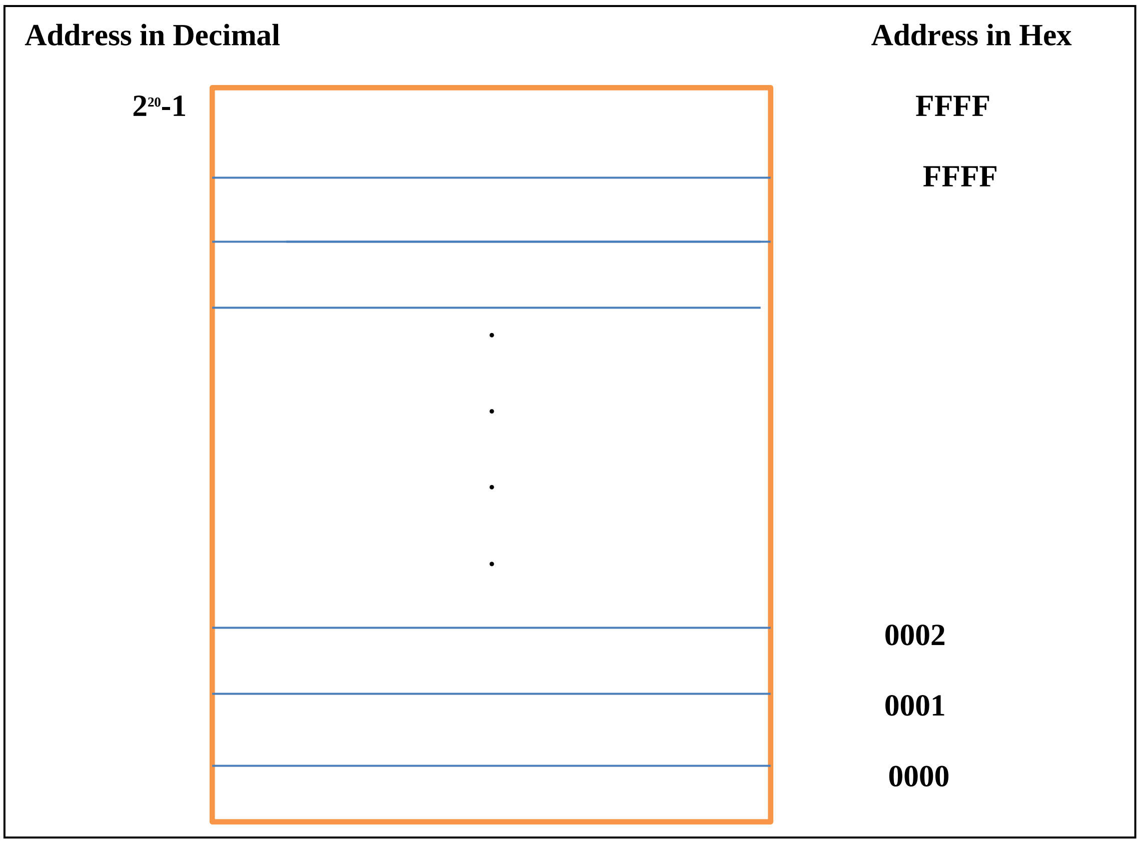


Figure 3: Logical view of the system memory

Two basic memory operations:-

The memory unit supports two fundamental operations: Read and Write. The read operation read a previously stored data and the write operation stores a value in memory.See Figure 4

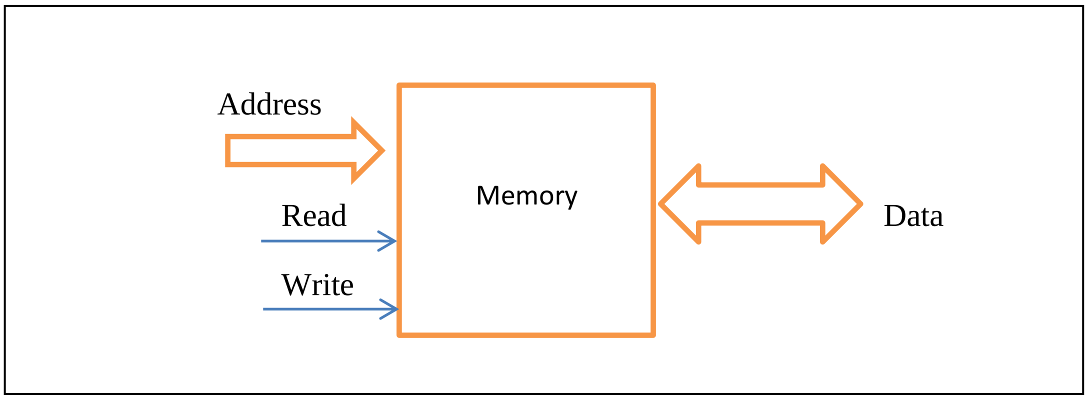


Figure 4: Block diagram of system memory

Steps in a typical <u>read cycle</u>:

1- Place the address of the location to be read on the address bus.
2- Activate the memory read control signal on the control bus.
3- Wait for the memory to retrieve the data from the address memory location.
4- Read the data from the data bus.
5- Drop the memory read control signal to terminate the read cycle.

Steps in a typical <u>write cycle</u>:

1- Place the address of the location to be written on the address bus.
2- Place the data to be written on the data bus.
3- Activate the memory write control signal on the control bus.
4- Wait for the memory to store the data at the address location.
5- Drop the memory write control signal to terminate the write cycle.

Addresses: group of bits which are arranged sequentially in memory, to enable direct access, a number called address is associated with each group. Addresses start at 0 and increase for successive groups. The term location refers to a group of bits with a unique address. Table 1 represents Bit, Byte, and Larger units.

Table1: Bit, Byte, and Larger units.

Name	Number of Byte
Bit	0 or 1
Byte	is a group of bits used to represent a character, typically 8-bit.
Word	2-byte (16-bit)
Double Word	4-byte (32-bits)
Quadword	8-byte (64-byte)
Paragraph	16-byte (128-bit)
Kilo Byte (KB)	The number 2^{10}=1024=1 KB thus 640K=640*1024=655360 bytes)
Megabyte (MB)	(1024*1024) byte or 1,048,576 byte) approximately 1,000,000 bytes
Gigabyte (GB)	(1024*1024*10240byte) or (1,073,741,824 byte), approximately 1,000,000,000 bytes.
Terabyte (TB)	Approximately 1,000,000,000,000 bytes.

Memory chips:

Memory chips have two main properties that determine their application, storage capacity or size and access time or speed. A memory chip contains a number of locations, each of which stores one or more bits of data known as its bit width. The storage capacity of a memory chip is the product of the number of locations and the bit width. For example, a chip with 512 locations and a 2-bit data width has a memory size of 512×2=1024 bits.

Since the standard unit of data is a byte (8 bits), the above storage capacity is normally given as 1024/8 =128 bytes.

The number of locations may be obtained from the address width of the chip. For example, a chip with 10 address lines has 2^{10}= 1024 or 1 k locations. Given an 8-bit data width, a 10- bit address chip has a memory size of 2^{10} ×8 = 1024×8 = 1k ×1 byte = 1 KB.

The computer's word size can be expressed in bytes as well as in bits.

For example, a word size of 8-bit is also a word size of one byte; a word size of 16- bit is a word size of two byte. Computers are often described in terms of their word size, such as an 8-bit computer, a 16-bit computer and so on.

For example, a 16-bit computer is one in which the instruction data are stored in memory as 16-bit units, and processed by the CPU in 16-bit units. The word size also indicates the size of the data. Bus which carries data between the CPU and memory and between the CPU and I/O devices. To access the memory, to store or retrieve a single word of information, it is necessary to have a unique address. The word address is the number that identifies the location of a word in a memory.

Each word stored in a memory device has a unique address. Addresses are always expressed as binary number, although hexadecimal and decimal numbers are often used for convenience.
The second properties of memory chips is access time, access time is the speed with which a location within the memory chip may be made a variable to the data bus. It is defend as the time interval between the instant that an address is sent to the memory chip and the instant that the data stored in to the location appears on the data bus. Access time is given in nanosecond (ns) and varies from 25 ns to the relatively slow 200 ns.

NOTS:

- The large computer (mainframes) have word-sizes that are usually in the 32-to-64 –bits range.

- Mini computers have a word sizes from 8-to-32-bits range.

- Microcomputers have a word sizes from 4-to-32-bits range.

In general a computer with a larger word size can execute programs of instruction at a fast rate because more data and more instruction are stuffed into one word. The larger word sizes, however, mean more lines making up the data bus, and therefore more interconnections between the CPU and memory and I/O devices.
The word size is 4-bit therefore there are 4-data I/P lines and 4data O/P lines.
This memory has 32 different words, and therefore has 32 different addresses (storage location) from (00000) to(11111). Thus, we need a 5 address I/P lines.

Memory capacity = number of memory storage
Location ×size of each word
= (number of word) × (number of bits per word)
= m (word)*n (bits)
= m*n bits

The capacity of memory depends on two parameters, the number of words (m) and the number of bits per word (n).
Every bit added to the length of address will double the number of words in the memory.
The increase in the number of bits per bits requires that an increase the length of data I/P and data O/P lines.

EX:-
A certain memory chip is specified as 2K×8:
1. How many words can be stored on this chip?
2. What is the words size?
3. How many total bits can this chip store?

SOL:-
1. 2K =2 × 1024 = 2048 words
2. The word size is 8-bits (1 byte).
3. Capacity = 2048 × 8 = 16 KB. Memory chip

EX:- A certain memory chip is specified as 2K × 16
1. How many words can be stored on this chip?
2. What is the words size?
3. How many total bits can this chip store?

SOL:-
1. 2K = 2 × 1024 = 2048 words
2. The word size is 16-bits(2 byte).
3. Capacity = 2048 * 16 = 32KB.

EX:- Which memory stores the most number of bits:
2MG × 8 memory or 2MG × 16 memory?

SOL:-
1. Capacity =(2 × 1024 ×1024) × 8 = 16,777,216 bits.
2. Capacity =(2 × 1024 ×1024) ×16= 33,554,432 bits.

EX:- Which memory stores the most number of bits:
4MG × 8 memory or 2MG × 16 ?

SOL:-
1. Capacity = (4 × 1024 ×1024) × 8 =33,554,432 bits.
2. Capacity = (2 × 1024 × 1024) ×16= 33,554,432 bits.

EX:- A certain memory has a capacity of 4K × 8
1. How many data I/P & data O/P lines?
2. How many word address line?
3. What is its capacity in byte?

SOL:-
1. 8each line:- data I/P lines = data O/P lines =8
2. 4 × 1024 = 4096 words

Thus, there are 4096 memory add nesses
2^8 =4096 2^8 =2^{12}
So X=12 it required a 12 bit address line
3. The capacity = (4 ×1024) × 8= 32,768 bit = 32,769/8 =4096 byte
(Since 1byte = 8 bit).

EX: - the a certain memory has a capacity of 4K×16
1. How many data I/P & data O/P lines?
2. How many word address lines?
3. What is its capacity in byte?

SOL:-
1. 16 each one.
Data I/P lines = data O/P lines =16
2. 4 × 1024 = 20496 words
Thus, there are 4096 memory addresses.
4096 = 2^{12}
It is require a 12-bit address line.
3. Capacity = (4 × 1024) × 16 = 65,536 bit
= 65,536 / 8 = 8.192 byte

Types of memory
The memory unit can be implemented using a variety of memory chips- different speeds, different manufacturing technology, and different sizes.

1- Read Only Memories (ROM):
ROMs allow only read operation to be performed. This memory is non-volatile. Most ROMs are programmed and cannot be altered.
This type of ROM is cheaper to manufacture than other types of ROM. The program that controls the standard I/O functions (called BIOS) is kept in ROM, configuration software.
Other types of ROM include:
A- Programmable ROM (PROM).
B- Erasable PROM (EPROM) is read only memory that can be Re-programmed using special equipment.
C- EAPROM, Electrically Alterable Programmable ROM is a Read Only Memory that is electrically reprogrammable.

2- Read/Write Memory (RAM):-
Read/Write memory is commonly referred to as Random Access Memory (RAM); it is divided into static and dynamic.

A-Static RAM (SRAM): used for implementing CPU registers and used for special high speed memory called cache memory this greatly improves system performance. **Static RAM** keeps its value without having to be refreshed.
B-Dynamic RAM (DRAM), the bulk of main memory in a typical computer system consists of dynamic ram. DRAM is where programmed, data are kept when a program is running. It must be refreshed with in less than a millisecond or losses its contents.

* The differences between RAM and ROM :-

RAM	ROM
• Stand for Random-Access Memory	• Stand for Read Only Memory
• Read / Write Memory	• Read Only Memory
• Sending data (writing) to RAM memory address is called destructive write because the new data erases whatever was there before.	• Sending data to ROM memory address is in effective because the contents of ROM cannot changed (write not allowed) because this memory for read only.
• Form of primary storage for holding temporary data and instruction	• Form of primary storage for holding permanent data and instruction
• Volatile :program and data are erased when the power is off	• Permanent :program and data are intact even power is off
• Type of RAM is ➢ Static RAM ➢ Dynamic RAM	• Type of ROM is ➢ PROM ➢ EPROM ➢ EAPROM

3-Cache Memory: many modern computer applications (Microsoft office 98, for example) are very complex and have huge numbers of instructions it takes considerable RAM capacity (usually a minimum 16MB) to store the entire instruction set. Or you may be using an application that exceeds your RAM. In that case, your computer has to go into secondary storage to retrieve the instruction. To alleviate this problem, software is often written in smaller blocks of instruction. As need, these blocks can be brought from secondary storage into RAM; this is still slow however, cache memory is the place closer to the CPU where the computer can temporarily store those blocks used most often. Those used less often remain in RAM until they are transferred to cache; those used infrequently stay stored in secondary storage. Cache memory is faster than RAM. Because, the instructions travel a shorter distance to the CPU.

4- **Flash memory:** is one such device. This memory can be accessed like RAM (read and written), but is non-volatile i.e. it is a form of permanent storage. At the time of writing flash memory is available in the 512Mb to 1GB range. One disadvantage of current NVRAMs is that they cannot be written to, as quickly as ordinary RAM. However, they are much faster to access than disk storage systems and they consume less power, so that in small portable computer systems they offer an alternative low-powered option to disk storage. However, NVRAMs are more expensive than disk storage devices.

5- Virtual Memory: If your computer lacks the random access memory (RAM) needed to run a program or operation, Windows uses virtual memory to compensate. Virtual memory combines your computer's RAM with temporary space on your hard disk. When RAM runs low, virtual memory moves data from RAM to a space called a *paging file*. Moving data to and from the paging file frees up RAM to complete its work. The more RAM your computer has, the faster your programs will generally run. If a lack of RAM is slowing your computer, you might be tempted to increase virtual memory to compensate. However, your computer can read data from RAM much more quickly than from a hard disk, so adding RAM is a better solution.

-Bios: Short for (**B**asic **I**nput / **O**utput **S**ystem), Bios is a chip located on all computer motherboard that contains instructions and setup for how your system should boot and how it operates. To the right is a picture of what a **BIOS** chip may look like in your computer. The BIOS includes instructions on how load basic computer hardware and includes a test referred to as a **POST** (**P**ower **O**n **S**elf **T**est) that helps verify the computer meets requirements to boot up properly, if the computer does not pass the **POST**, you will receive a combination of beeps indicating what is malfunctioning within the computer.

In most PCs, the **BIOS** have four main functions:-

1-POST: - Test computer hardware, ensuring hardware is properly functioning before starting process of loading operation system. Additional information on the POST can be found on our POST/Beep Code Page.

2- Bootstrap Loader:- Process of location the operating system. If capable operation system located, BIOS will pass the control to it.

3-BIOS:-Software/Drives that interface between the operating system and your hardware. When running DOS or Windows you are using complete BIOS support.

4-BIOS/CMOS Setup:- Configuration program that allows you to configure hardware setting including system setting such as computer password, time, and data.

***Secondary Storage (Backing Storage)**

Secondary storage is designed to store very large amounts of data for extended periods of time .secondary storage can have memory capacity of gigabyte or more; only small portions of the data are placed in primary storage at any one time. Secondary storage **has the following Characteristics:**

1-it is non-volatile

2-it takes much more time to retrieve data from secondary storage than it does from RAM

3-it much more cost-effective than primary storage

4-it can take place on a variety of media each with its own technology, as is cussed below:

a) Magnetic tape

b) Magnetic disc

c) Magnetic diskette (floppy disc)

d) Optical discs

A- Features of the Magnetic tapes

1- It is1/4 inch wide and 300, 1200, 2400, or 3600 feet long.

2- It has a plastic base, coated with magnetic able material on one side.

3- Data is stored in tracks; there are 7 or 9 tracks (depending on the tape unit) which run the length of the tape. The data is recorded so that one character is recorded across the 7 or 9 tracks.

4- The density of recording can vary between 2

5- It is serial access device.

6- The tape is reusable i.e. it can be overwritten

7- The same tape can be used for input and output. The tape can be writing protected.

B-Features of magnetic disks (hard disks)

1. Disks are randomly accessed.

2. Disks are of size and shape similar to a long-playing record

3. The surfaces of each disk are of magnetic able material.

4. Each disk surface is divided into a number of concentric tracks (typically 200).

5. Disks are placed on pack and each pack may have 6 or 11 disks and is used as a single unit.

6. The latest models of disk packs can store many hundreds of megabytes of data (i.e. hundreds of millions of characters).

C-Features of floppy disks

1- A pliable disk permanently sealed with a rigid, protective plastic envelope

2- They have random access facility.

3- Data are stored in concentric tracks

4- The floppy disks sizes are 8, 5 1/4, 3 1/2 inch.

5- Storing capacity of 3 1/2 inch disks is 1.44 megabytes i.e. one million four hundred thousand characters.

Floppy Disk (1.44MB)

D-Features of optical disks

1- This is a random access device.

2- Data is written into the disk by burning a permanent pattern into the surface of the disk by means of high precision laser beam.

3- Data is read by using the laser at lower intensity and detecting the pattern reflected from its beam by the surface of the disk there are many types of optical disks:

a- Compact disk read-only memory (CD-ROM) storing devices feature high capacity, low cost. It has become popular for recorded music as well as information (such as books) a variant is the digital video disk (DVD), used for movies.

b- Write once, read many (WORM) disk can be written.

c- Rewritable CD is a less common technology that allows the disk to be written upon and written up to 1.000 times.

Hard Disk Performance: Several basic parameters determine the performance of a given hard disk drive. A seek operation is the movement of the read/write head to the desired track.

1- Seek Time: A seeks time is the movement of the read\write head to the desired track. The seek time is the average time for this operation to be performed. Typically, hard disk drives have an average seek time of several milliseconds, depending on the particular drive.

2- Latency Time: The latency period is the time takes for the desired sector to spin under the head once the head is positioned over the desired track. Latency time depend on the constant rotational speed of the disk.

• The sums of average seek time and the average latency time is the access time for the disk drive.

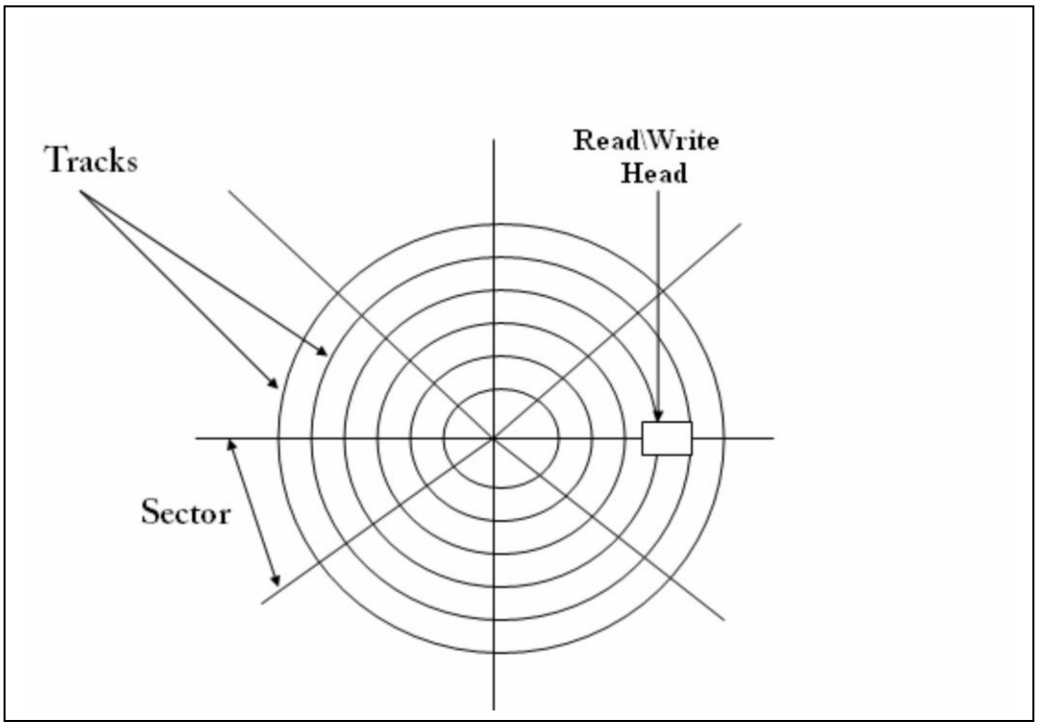


Figure (5): Hard Disk

Hard Disk

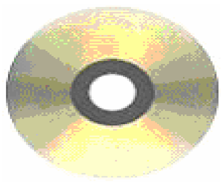
The Difference between Internal and External hard disks

Internal hard disks are located inside your main computer unit, while external hard disks are joined to the main computer unit via a lead which you plug into the back of your computer unit. Some external hard disks will plug into the USB port (connector) located at the back of your computer. Other external hard disks require the installation of a special card within your computer which allows the connection of the external hard disk to the computer unit.

CD-ROM (Compact Disk Read-Only Memory) is another form of secondary storage that is increasing in popularity. It is a low-cost storage medium with a very large capacity. Unlike disk storage, CD-ROM is a **WORM** (Write Once Read Many times) device i.e. it is a **read only** storage device. This means that like ROM, the disk comes with information already stored on it. Thus one of the main uses of CD-ROM is to disseminate information such as library catalogues, reports, manuals, journals, directories and software. It has also become a very popular medium for computer games. Many software vendors and computer manufacturers such as Sun and Apple distribute their software and manuals on CD-ROM. Many publishers now use CD-ROM especially for educational material and it is possible to buy encyclopedia and history texts in CD-ROM form. The CD-ROM has sufficient capacity not only to store the written text, but also video and audio material which require large amounts of storage, for example, a digital version of a passport size photograph requires up to a megabyte of storage.
CD-ROM uses the same technology as the compact audio disk or CD and such disks can also be used in a CD-ROM drive. Optical scanning techniques, using lasers, are employed with CD-ROMs, which allow massive amounts of data to be stored in a compact area. A CD-ROM drive is about the same size as a floppy disk drive. CD-ROM is currently more reliable and durable than magnetic media (disks and tapes). In terms of capacity, a single CD-ROM may store up to600 megabytes. In terms of text this is equivalent to about 200 books of 1000 pages each.
A disadvantage of CD-ROM is that it takes longer to access information compared to a hard disk. However, clever software tailored for particular applications often means that this is not a serious problem. **Video Disks** are similar to CD-ROM (but have a larger capacity) and are used for similar applications.

Rewritable CD storage (CD-R) is now becoming more widely used. This storage combines there liability and storage capacity of CD-ROM with the flexibility of magnetic disks in that user can store their own information on them. They are still slower to access than conventional hard disks. **Magneto-optical (MO)** disks combine the use of magnetic and optical principles to store information. MO disks have a smaller capacity than CD-ROMs (e.g. a 3.5 inch MO disk stores128Mb) and are quite expensive in comparison to conventional hard disks.

CD-ROMs are now being replaced by **DVD-ROMs** (Digital Versatile Disks). DVD-ROM capacity ranges from 4.7 GB upward (4.7x 2 or 4.7 x 4 GB). These are also used for distributing films as a rival to video tapes. A current PC will typically have (DVD) drives which are also capable of reading conventional CDs. DVD-ROMs are also available which allow users to store files.



CD-ROM = 640 Mb
DVD-ROM = up to 8.5 GB

Computer Hierarchy

The traditional way of comparing classes of computers is by their processing power. This section presents each class of computers beginning with most powerful and ending with least powerful. We describe the computers and their respective roles in modern organizations.

1-Super computers

• Are the computer with the most processing power.
• The primary application of it has been in scientific and military work.
• Are used for image creation and processing.
• Are used to model the weather for better weather prediction
• To make many sequences in motion pictures (e.g. star wars and Jurassicpark)
• Super computers generally operate 4 to 10 times faster than the next most powerful computer class.

2-Mainframe Computers

•Are less powerful and generally less expensive than super computers.
• Are used for centralized data processing and maintain large databases.
• Examples of mainframe applications include airline Reservation systems, student's grade calculations and reporting.
• A mainframe system may have anywhere from 50MB several gigabytes of primary storage, secondary storage may use high capacity magnetic and optical storage media with capacities in the gigabyte rang.

3- Minicomputers

• Are called midrange computers
• Are relatively small inexpensive that performs the same functions as mainframe computers but to a limited extent.
• Are used to accomplish specific tasks, such as process control, scientific research, and engineering applications.

4- Workstations
• Are also called desktop engineering workstations
• Are developed to provide the high levels of performance demanded by engineers
• Provide very high-speed calculations and high-resolution graphic
• These computers have found a widespread acceptance within the scientific community and with the business community.
5-Microcomputers
• Also called micros or personal computers (PCs) Are smallest and least expensive category of general purpose computers
• They can be subdivided into three classifications based on their size
a) Desktops **b**) laptops and notebooks **c**) palmtops

3-Input Technologies
a) Keyboards
Is the most common input device the keyboard is designed like a type writer but with additional function keys.

Keyboard

b) Mouse and track balls:
A mouse is hand held device used to point a cursor at a desired place onscreen, such as an icon, cell in a table. A variant of the mouse is the trackball, which is often used in graphic design, the user holds an object much like a mouse but rather than moving the entire device to move the cursor.

Stander Mouse optical Mouse

Trackball

c) Touch screen
Is a technology that divides the computer screen into different areas. Users simply touch the desired area (often buttons or squares) to trigger an action.

d)a stylus
Is a pen style device that allows the user either to touch parts of predetermined menu of options

e) Joysticks
Is used primarily at workstations that can display dynamic graphics, they can also use to play video games.
Many games require a joystick for the proper playing of the game. There are many different types, the more sophisticated respond to movement in 3 axis directions, as well as having a number of configurable buttons. Like most things in life you get what you pay for with joysticks and it is worth investing in a good, strongly constructed model, especially bearing in mind that children will hammer these devices whilst playing games.

f) Voice input for PCs (microphones)
Early voice recognition systems offered very poor results, due to the limitations of the software combined with hardware limitations. It takes an awful lot of CPU processing power to convert the spoken word into text which appears on the screen. Things are changing rapidly however and recent systems allow you

to talk to a PC and see text appear on the screen. Most of these systems require an initial training period, where you train the software to respond to your particular voice. Whilst still not perfect this is a key technology of the future.

g) Web Cams
Ever since it was invented, the Web has become increasingly interactive. You can now use a small digital movie camera (a Web cam) mounted on the PC monitor to allow two-way communication involving not just text Communication but sound and video communication as well.

Web Camera

h) Scanners
A scanner allows you to scan printed material and convert it into a file format which may be used within the PC. You can scan pictures and then manipulate these inside the PC using a graphics application of your choice. In addition, you can scan printed text and convert this not just to a picture of the text but also to, actual text which can be manipulated and edited as text within your word-processor. There are a number of specialist programs, generally called OCR (Optical Character Recognition) programs which are specifically designed for converting printed text into editable text within your applications.

Scanner

i) Light Pens
A light pen is used to allow users to point to areas on a screen and is often used to select menu choices.

j) Digital Cameras
A digital camera can be used in the same way a traditional camera can, but instead of storing images on rolls of film which require developing, the images are stored digitally in memory housed within the camera. These pictures can

easily be transferred to your computer and then manipulated within any graphics programs which you have installed on your computer.
Currently they are limited by the quality of the image recorded and the number of pictures which you may store within the camera.

Digital Camera

4- Output Technologies

A) Monitors

Are the video screens used with most computers that display input as well as output like television sets, monitors come in a variety of sizes and color/resolution quality, and like television sets, the common desktop monitor uses cathode ray tube (CRT) technology to shoot beams of electrons to the screen. The points on the screen known as pixels, the more pixels on the screen mean the better resolution. Here are some other useful facts about monitors:
1-portable computers use a flat screen that uses liquid crystal display (LCD) technology not (CRT)
2-LCDs use less power than CRT monitors but cost six to eight times what an equivalent CRT

Monitor

B) Printers

There are Two types of printers:

1-impact printers:
Work like typewriters, using some kind of striking action, raised metal character strikes an inked ribbon that makes a printed impression of the character on the paper, these devices cannot produce high-resolution graphics, and they are relatively slow, noisy, and subject to mechanical failure, although inexpensive, they are becoming less popular.

2- Non-impact printers
Come in two styles

a) **laser printers:** Are higher speed, high _quality devices that uses laser beams to write information on photo sensitive drums, laser printers produce very high quality resolution text and graphics.

Printer

b) **Inkjet printers**: work differently, by shooting fine streams of colored ink onto the paper. These are less expensive than laser printers, but offer less resolution quality.

3- Plotters: are printing devices that use computer-directed pens for creating high-quality images. They are used in complex, low-volume situations, for example, creating maps and architectural drawings. Some plotters are quite large, suited for producing correspondingly large graphics.

Machine language:-

The native language of the computer, In order for a program to run, it must be presented to the computer as binary-coded machine instructions that are specific to that CPU model or family. Although programmers are sometimes able to modify machine language in order to fix a running program, they do not create it. Machine language is created by programs called "assemblers," "compilers" and "interpreters," which convert the lines of programming code a human writes into the machine language the computer understands. Machine language tells the computer what to do and where to do it.

Assembly language:-

Assembly language program using instruction abbreviation called mnemonics, such as LD(load), ST(store) and ADD(Add to Accumulator).This is converted to machine language program with a translator called "assembler".

High-level-Language (HLL):-

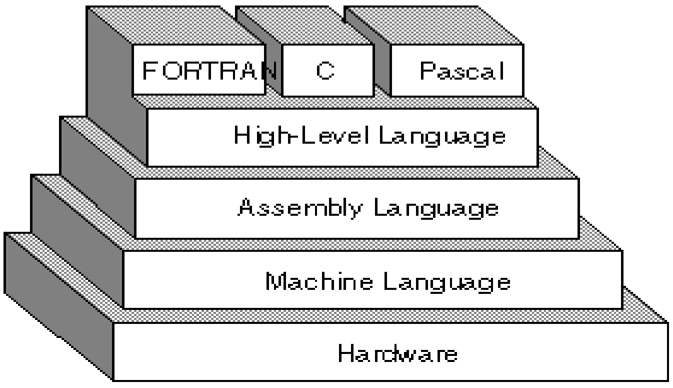
High-Level-programming language is one type of programming Language available. The other type of programming language is known as low-level-language or assembly language. High level-language is easier to learn and understood than the assembly language, because high level languages uses names and commands the resemble English, while the assembly language uses mnemonic codes.

- Some of the common high-level-languages are:
-Fortran (Formula Translation) for engineers.
-COBOL (Common Business Oriented Language) for business Programmers.
-Basic (Beginner's All-purpose symbolic Instruction Code) for Engineers and scientists.
- Pascal.

Unlike assembly programs, high-level-languages programs may be used with different makes of computers, while the assembly languages are machine oriented.

Other advantages of high-level-languages are:

1) They are easier to learn than assembly languages.
2) They are easier to use for problem solving, than assembly.
3) They require less time to write, than assembly.
4) They provide better documentation.
5) They are easier to monition.



<u>A compiler</u> is a program that translates a source language text into an quivalent target language text.

What is a compiler?

In order to reduce the complexity of designing and building computers, nearly all of these are made to execute relatively simple commands (but do so very quickly).

A compiler translates (or compiles) a program written in a high-level programming language that is suitable for human programmers into the low-level machine language that is required by computers. During this process, the compiler will also attempt to spot and report obvious programmer mistakes using a high-level language for programming has a large impact on how fast

-**<u>Interpreter</u>** is another way of implementing a programming language. Interpretation shares many aspects with compiling. Lexing, parsing and type-checking are in an interpreter done just as in a compiler. But instead of generating code from the syntax tree, the syntax tree is processed directly to evaluate expressions and execute statements, and so on. An interpreter may need to process the same piece of the syntax tree (for example, the body of a loop) many times and, hence; interpretation is typically slower than executing a compiled program.

In computer science, an **interpreter** is a computer program that executes, i.e. *performs*, instructions written in a programming language. An interpreter generally uses one of the following strategies for program execution:

1. Execute the source code directly.
2. translate source code into some efficient intermediate representation and immediately execute this
3. explicitly execute stored precompiled code made by a compiler which is part of the interpreter system

Introduction to Microcomputer Programming

Instruction: is a unit of information that used to indicate to the computer what operation it is to performs.

Program: is a set of instructions that are used together to accomplish a complete computational task.

Instruction types:

Because of their limited size, microcomputers do not usually have a sufficient instructions repertoire to be able to perform each of these operations with just one instruction. In fact a measure of the sophistication of a MP is the number of these operations that can be done with only one instruction and the flexibility these instructions have in performing their operations.

1- **Transfer:** reg-reg, mem-reg and transfer onto and off of the stack
2- **Arithmetic:** +,-,*,/, and negation, the types of data can be used to represent the numbers (int, float,BCD)
3- **Logical:** OR, AND, exclusive OR and complement.
4- **Shift and Rotate:** SR,SL,RR,RL, and multiple bytes shifts and rotates.
5- **Indexing and Counting**: incrementing & decrementing.
6- **Bit Manipulation:** selectively setting, clearing, and testing bits within a byte or word.
7- **Looping:** A combination of incrementing or decrementing, comparing, testing, and branching; the purpose being to repetitively execute a program segment.
8- **Branching:** Ordinary instructions are taken form consecutive memory locations, however, it some necessary to jump out of the ordinary sequence of code. The branching instructions perform this function. They fall into three main categories
    i- Unconditional jump
    ii- Conditional jump.
    iii- Subroutines jump and Returns.
9- **I/O communication & Transfer:** for initiating and performing I/O data transfers, testing the I/O status, giving I/O commands, and so on.

Addressing modes:

In order to communicate with their various components, all computers must have some means of identifying the individual external memory locations, stack, CPU register, and I/O interface register. Usually a computer will have separate address spaces for its memory and its CPU registers set, and will sometimes have separate space for it I/O interface registers and stack. This mean that there would be two to four storage unites with address 0; one memory location, one CPU register, and perhaps one I/O device register and one stack or internal memory location. Which of the two to four units is intended would be determined by the instruction.

Because memory is used to store instructions and data, all computers include a variety of methods, called addressing modes for accessing it.

1- **Direct Addressing Mode**: The instruction contains the exact memory address of the data item.
2- **Register Addressing Mode**: the instruction specifies a register or a register pair which the data is located.
3- **Register Indirect Addressing Mode**: The instruction specifies a register pair which contains the memory address where the data is locate.
4- **Immediate Addressing Mode**: The instruction contains the data itself.
5- **Base Addressing Mode**: The address is formed by adding the contents of a memory location or register to a specified number called a *displacement*.
6- **Indexing**: Is the process of increment or decrement on address as the computer sequences through a set of consecutive address.
7- **Auto-incrementing / Auto-decrementing**: A form of indexing in which the index is automatically incremented (or decremented) by the instruction.
8- **Relative Addressing Mode**: The address is the sum of a number and the current contents of the Program Counter.
9- **Page Addressing Mode**: A form of indirect addressing mode in which the specified address is formed from a page address, which determines the high-order bits of the address and a displacement which determines the low-order bits (the page address normally comes from a special register called the "page address register").

The 8085 have four addressing mode for addressing data stored in memory or in registers: (direct, register, register indirect, immediate).

Instruction Format:

All instructions are made up of a sequence of bytes, each byte being a combination of 1's and 0's. The portion of an instruction that specifies what the instruction does is called the ***Operation Code*** **(OP Code)**. Any address or a piece of data that is needed by an instruction in order to complete its execution is called ***Operand***.

There are four type of address instruction

1- Three address instruction: **ADD A,B,C** this mean A=B+C

2- Two address instruction (Double operand instruction): **ADD A,B** this mean A=A+B

3- One address instruction: (Single operand instruction): **ADD B** this mean Acc=Acc+B

4- Zero Address instruction: (Zero operand instruction): Ex: **ADD** this type of instruction is use the stack to get the operands and then store the result again in the stack.

The instruction size must be small as possible to reduce fetch time and the space required to store the instruction. To reduce the number of bits needed to specify the operands is obtained by:

1- Permit only branch instruction to include the address of the next instruction.
2- Using the location of one of the data item being operated on to store the result.

3- Having either one or both operands contained in the CPU registers (register address needs few bits to designate a register address).

OP-Code	Destination	Source

Source: is the operand from which the information is taken.

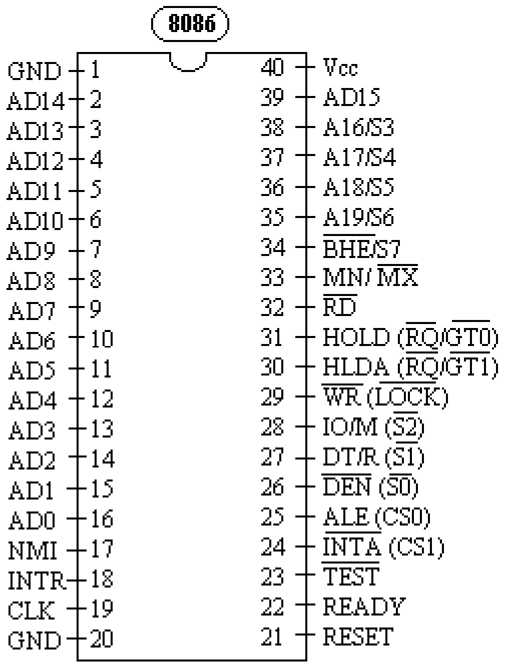
Destination: is the location that is changed.

A typical Assembly language instruction has the following format:

LABLE: OPERATION-MNEMONIC OPERANDS; COMMENTS

An Introduction to 8086 Microprocessor Architecture

The 8086 is a 16-bit microprocessor chip designed by Intel in 1978, which gave rise to the x86 architecture. It has 20-bit address bus and 16-bit data bus.



Address / Data Bus:

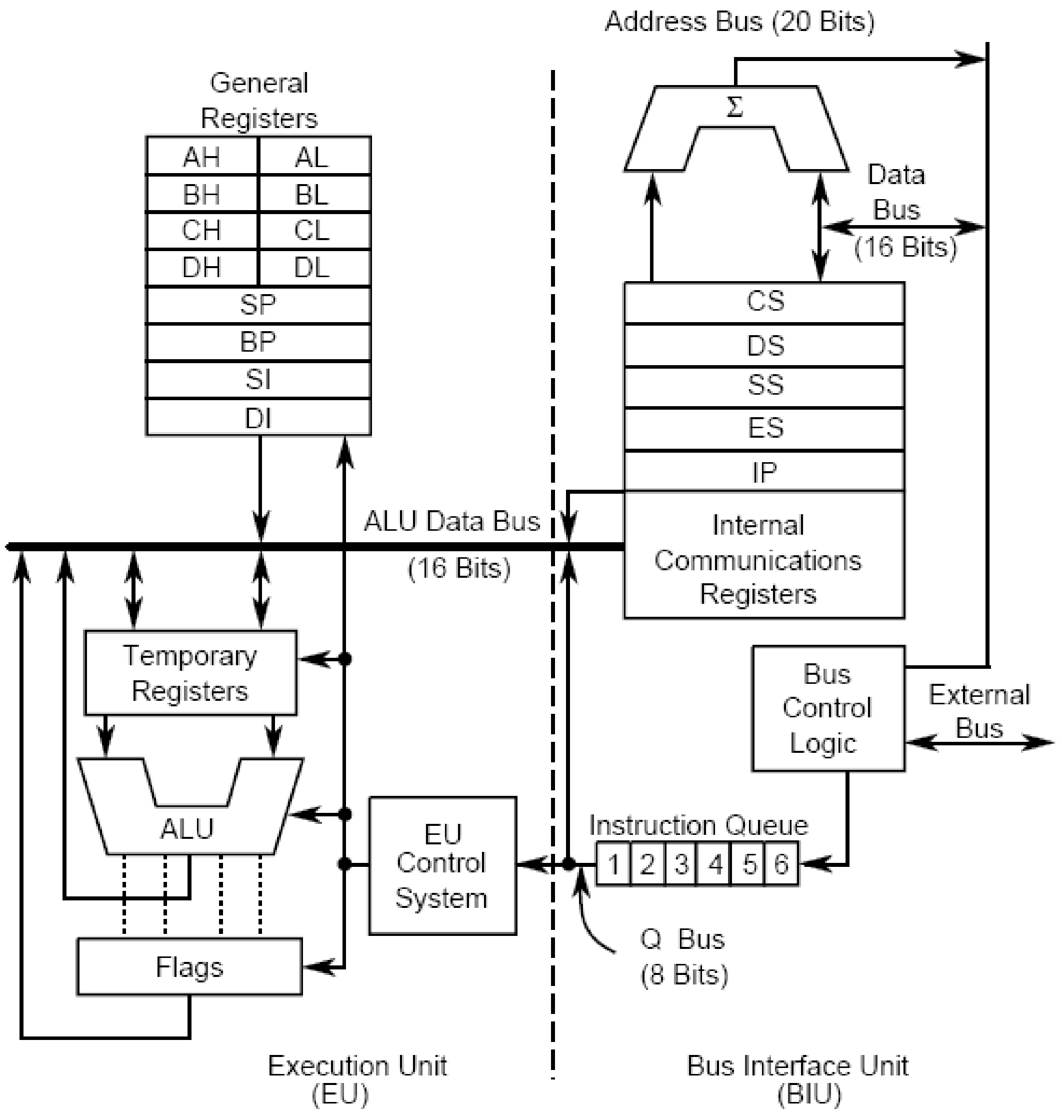
The address bus is 20-bit (A0-A19). A 20-bit address gives the 8086 a one Mega Byte memory address space. Moreover, it has an independent I/O address space, which is 64KB in length.

The 16-bit data bus lines D0-D15 are actually multiplexed with address lines A0-A15.

Internal Architecture of the 8086 MP:

The 8086 Microprocessor Core incorporates two separate processing units: an **Execution Unit (EU)** and a **Bus Interface Unit (BIU)**. The Execution Unit is functionally identical among all family members.

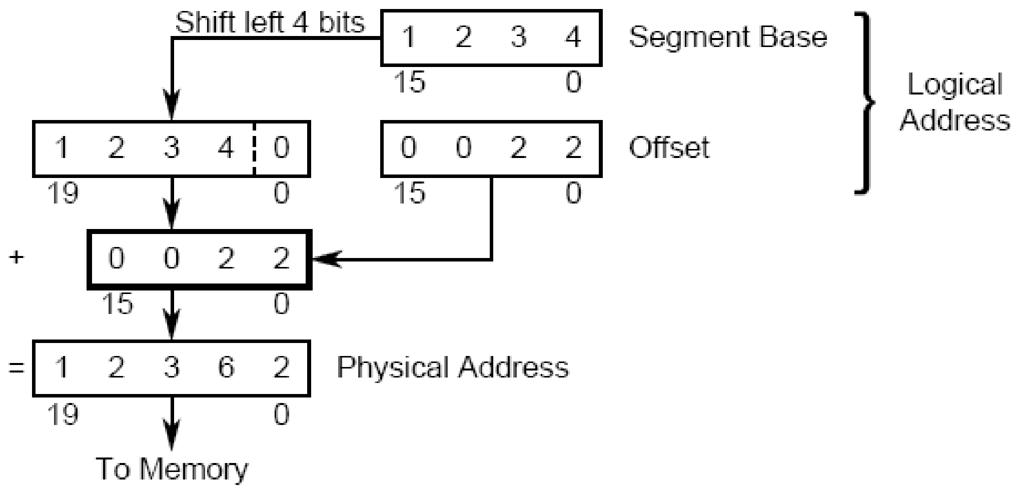
The Bus Interface Unit is configured for a 16-bit external data bus for the 8086 core. The two units interface via an instruction prefetch queue.



Bus Interface Unit

The Bus Interface Unit executes all external bus cycles. This unit consists of the segment registers (CS, DS, SS and ES) allow simple memory partitioning to aid modular programming, the Instruction Pointer (IP), the instruction code queue and several miscellaneous registers. The Bus Interface Unit transfers data to and from the Execution Unit on the ALU data bus.

The Bus Interface Unit generates a 20-bit physical address in a dedicated adder. The adder shifts a 16-bit segment value left 4 bits and then adds a 16-bit offset. This offset is derived from combinations of the pointer registers, the Instruction Pointer and immediate values. Any carry from this addition is ignored.



Physical Address Generation

During periods when the Execution Unit is busy executing instructions, the Bus Interface Unit sequentially prefetches instructions from memory. As long as the prefetch queue is partially full, the Execution Unit fetches instructions. BIU is also responsible for generating bus control signal such as those for memory Read/Write and I/O Read/Write.

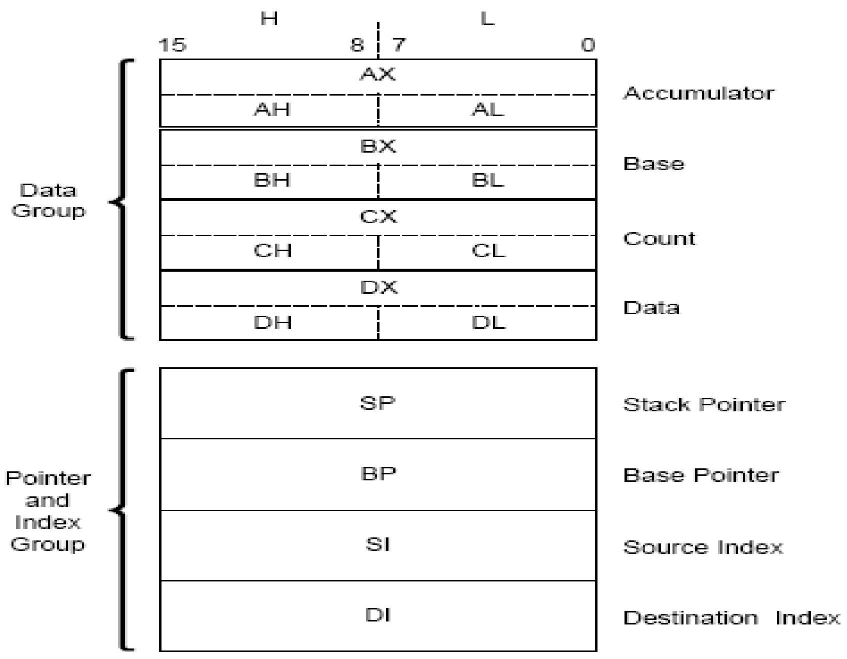
Execution Unit

The Execution Unit executes all instructions, provides data, addresses to the Bus Interface Unit, and manipulates the general registers and the Processor Status Word. The 16-bit ALU within the Execution Unit maintains the CPU status and control flags and manipulates the general registers and instruction operands. All registers and data paths in the Execution Unit are 16 bits wide for fast internal transfers. The Execution Unit does not connect directly to the system bus. It obtains instructions from a queue maintained by the Bus Interface Unit. When an instruction requires access to memory or a peripheral device, the Execution Unit requests the Bus Interface Unit to read and write data. Addresses manipulated by the Execution Unit are 16 bits wide. The Bus Interface Unit, however, performs an address calculation that allows the Execution Unit to access the full megabyte of memory space. To execute an instruction, the Execution Unit must first fetch the object code byte from the instruction queue and then execute the instruction. If the queue is empty when the Execution Unit is ready to fetch an instruction byte, the Execution Unit waits for the Bus Interface Unit to fetch the instruction byte.

- *Internal Registers of the 8086*

1- General Registers

The 8086 CPU has eight 16-bit general registers. The general registers are subdivided into two sets of four registers. These sets are the data registers (also called the H & L group for high and low) and the pointer and index registers (also called the P & I group).



General Registers

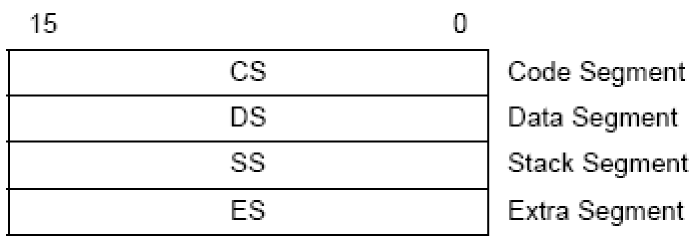
The data registers can be addressed by their upper or lower halves. Each data register can be used interchangeably as a 16-bit register or two 8-bit registers. The pointer registers are always accessed as 16-bit values. The CPU can use data registers without constraint in most arithmetic and logic operations. Arithmetic and logic operations can also use the pointer and index registers.

Implicit Use of General Registers

Register	**Operations**
AX	Word Multiply, Word Divide, Word I/O
AL	Byte Multiply, Byte Divide, Byte I/O, Translate, Decimal Arithmetic
AH	Byte Multiply, Byte Divide
BX	Translate
CX	String Operations, Loops
CL	Variable Shift and Rotate
DX	Word Multiply, Word Divide, Indirect I/O
SP	Stack Operations
SI	String Operations
DI	String Operations

2-Segment Registers

The 8086 memory space is 1 Mbyte in size and divided into logical segments of up to 64 Kbytes each. The CPU has direct access to four segments at a time. The segment registers contain the base addresses (starting locations) of these memory segments. The CS registers points to the current code segment, which contains instructions to be fetched. The SS register points to the current stack segment, which is used for all stack operations. The DS register points to the current data segment, which generally contains program variables. The ES register points to the current extra segment, which is typically used for data storage. The CS register initializes to 0FFFFH, and the SS, DS and ES registers initialize to 0000H. Programs can access and manipulate the segment registers with several instructions.



3-Instruction Pointer

The Bus Interface Unit updates the 16-bit Instruction Pointer (IP) register so it contains the offset of the next instruction to be fetched. Programs do not have direct access to the Instruction Pointer, but it can change, be saved or be restored as a result of program execution. For example, if the Instruction Pointer is saved on the stack, it is first automatically adjusted to point to the next instruction to be executed.

Reset initializes the Instruction Pointer to 0000H. The CS and IP values comprise a starting execution address of 0FFFF0H.

4-Flags

The 8086 MP has six status flags that the Execution Unit posts as the result of arithmetic or logical operations. Program branch instructions allow a program to alter its execution depending on conditions flagged by a prior operation. Different instructions affect the status flags differently, generally reflecting the following states:

• If the Auxiliary Flag (AF) is set, there has been a carry out from the low nibble into the high nibble or a borrow from the high nibble into the low nibble of an 8-bit quantity (low-order byte of a 16-bit quantity). This flag is used by decimal arithmetic instructions.

• If the Carry Flag (CF) is set, there has been a carry out of or a borrow into the high-order bit of the instruction result (8- or 16-bit). This flag is used by instructions that add or subtract multibyte numbers. Rotate instructions can also isolate a bit in memory or a register by placing it in the Carry Flag.

• If the Overflow Flag (OF) is set, an arithmetic overflow has occurred. A significant digit has been lost because the size of the result exceeded the capacity of its destination location. An Interrupt On Overflow instruction is available that will generate an interrupt in this situation.

• If the Sign Flag (SF) is set, the high-order bit of the result is a 1. Since negative binary numbers are represented in standard two’s complement notation, SF indicates the sign of the result (0 = positive, 1 = negative).

• If the Parity Flag (PF) is set, the result has even parity, an even number of 1 bits. This flag can be used to check for data transmission errors.

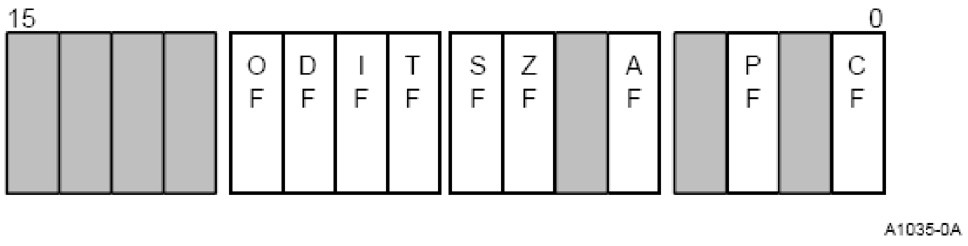
• If the Zero Flag (ZF) is set, the result of the operation is zero. Additional control flags can be set or cleared by programs to alter processor operations:

• Setting the Direction Flag (DF) causes string operations to auto-decrement. Strings are processed from high address to low address. Clearing DF causes string operations to auto-increment. Strings are processed from low address to high address.

• Setting the Interrupt Enable Flag (IF) allows the CPU to recognize maskable external or internal interrupt requests. Clearing IF disables these interrupts. The Interrupt Enable Flag has no effect on software interrupts or non-maskable interrupts.

• Setting the Trap Flag (TF) bit puts the processor into single-step mode for debugging. In this mode, the CPU automatically generates an interrupt after each instruction. This allows a program to be inspected instruction by instruction during execution.

The status and control flags are contained in a 16-bit Processor Status Word. Reset initializes the Processor Status Word to 0F000H.



Bit Mnemonic	Bit Name	Reset State	Function
OF	Overflow Flag	0	If OF is set, an arithmetic overflow has occurred.
DF	Direction Flag	0	If DF is set, string instructions are processed high address to low address. If DF is clear, strings are processed low address to high address.
IF	Interrupt Enable Flag	0	If IF is set, the CPU recognizes maskable interrupt requests. If IF is clear, maskable interrupts are ignored.
TF	Trap Flag	0	If TF is set, the processor enters single-step mode.
SF	Sign Flag	0	If SF is set, the high-order bit of the result of an operation is 1, indicating it is negative.
ZF	Zero Flag	0	If ZF is set, the result of an operation is zero.
AF	Auxiliary Flag	0	If AF is set, there has been a carry from the low nibble to the high or a borrow from the high nibble to the low nibble of an 8-bit quantity. Used in BCD operations.
PF	Parity Flag	0	If PF is set, the result of an operation has even parity.
CF	Carry Flag	0	If CF is set, there has been a carry out of, or a borrow into, the high-order bit of the result of an instruction.

The 8086 Addressing Mode

1- Register Addressing Mode

```
MOV AX,BX
MOV DS,SI
MOV AL,CH
MOV SP,SI
```

2- Immediate Addressing Mode

The immediate data can be byte or word and it stored in program storage memory in the byte location immediately following the op-code of the instruction. This value is also fetched into the instruction queue in the BIU.

```
MOV AX,0AF10H
MOV SI,0FFFFH
MOV AH,255
MOV AL,0A0H
```

3- Direct Addressing Mode

Effective Address (EA) is a 16-bit offset of the storage location of the operand.

Physical address= Offset + (shift DS 4-bit to the left)

```
MOV CX,BETA
MOV DS,TEST
MOV TEMP,ES
MOV TEMP,5550H
MOV BYTE PTR TEMP,50
MOV SUM+2,AX
```

4- Register Indirect Addressing Mode

It is similar to direct addressing mode in that the EA is combined with the contents of DS. This time EA resides in either a pointer register or index register (BX, BP, SI, DI).

Physical address= Reg.+ (shift DS 4-bit to the left)

```
MOV AL,[BX]
MOV [BP],CX ; this will use SS instead of DS
MOV [SI],DI
MOV [DI],CS
MOV [BX],0AFAFH
```

5- Based Addressing Mode

In this mode EA= displacement + Base register (BX or BP)

PA= displacement + Base register (BX or BP) + (shift DS 4-bit to the left)

```
MOV BETA.[BX],AL
MOV [BETA+BP],AL
MOV [BX+10],BX
```

6- Indexed Addressing Mode

This mode works identically to the based addressing however it use the contents of one of the index registers (SI, DI) instead of BX or BP.

EA= displacement + Index register (SI or DI)

PA= displacement + Index register (SI or DI) + (shift DS 4-bit to the left)

```
MOV ARRAY[SI],AX
```

7- Based Indexed Addressing Mode

Combining the based addressing mode and the indexed addressing mode together results new and more powerful mode.

EA= displacement + Base register (BX or BP)+ Index register (SI or DI)

PA= displacement + Base register (BX or BP) + Index register (SI or DI) + (shift DS 4-bit to the left)

```
MOV AH,[BX].BETA[SI]
```

String addressing mode

The string instructions automatically use the source and destination index registers to specify the EA.

Port Addressing Mode

Port addressing mode is used in conjunction with IN and OUT instructions to access input output ports. For ports in the I/O address space, only the direct addressing mode and indirect addressing mode using DX are available.

```
IN AL,DX
OUT 0A0H,AX
```

The 8086 Instruction format:

Byte 1			Byte 2			Byte 3	Byte 4	Byte 5	Byte 6
6	1	1	2	3	3				
OP.Code	D	W	MOD	Reg	R/M	Low Disp/Data	High Disp/Data	Low Data	High Data

D: Register Direction

D=0 :- Register in Byte 2 is a source operand.

D=1 :- Register in Byte 2 is a destination operand.

W: Data Word size

W=0 :- Data is 8-bit.

W=1 :- Data is 16-bit.

Ex1:

MOV AL,BH
Op code=100010
D=0
W=0
MOD=11
Reg=111
r/m=000

```
10001000 11111000
    88               F8
```

Then the machine code for MOV AL,BH is 88 F8 (2 Bytes).
Or the machine code is 8A C7 (How?).

Ex2:

MOV [12FAH],CX
Op code=100010
D=0
W=1
MOD=00
Reg=001
r/m=110
Byte 3=FAH
Byte 4=12H

```
10001001 00001110 11111010 00010010
     89       0E            FA             12
```

Then the machine code for MOV [12FAH],CX is 89 0E FA 12 (4 Bytes).

Note: Some assembly instructions have more than one machine code and may be different in size, especially the instruction that used accumulator (AL,AX).

H.W:

What is the size of the following program (in Bytes)?

```
MOV BL,AL
XOR CL,[1234H]
ADD [BX].DISP[SI],CX
MOV [BP].[DI+55],0ABCDH
MOV [BX].[SI+0FFAAH],0ABCDH
```

The 8086 Instruction Set

The microprocessor's instruction set defines the basic operations that a programmer can use. The instruction organized into groups of functionally related instructions. These groups consist of the:

1- Data transfer instructions.
2- Arithmetic instructions.
3- Logical instructions.
4- Control transfer instructions.
5- String manipulation instructions.
6- Processor control instructions.

1. Data Transfer Instructions

1.1 The MOV Instruction

The move (**MOV**) instruction is one of the instructions in the data transfer group of the 8086/8808 instruction set. The format of this instruction is

MOV dest, src (dest) ← (src)

The source and destination can be register or memory address, ….. etc (as described in the addressing modes).

1.2 The XCHG Instruction

The exchange (**XCHG**) instruction is used to swap data between two registers or register with memory.

XCHG dest, src (dest) ←→ (src)

Instruction Operands:

XCHG accum, reg
XCHG mem, reg
XCHG reg, reg

Ex:

```
XCHG  TEMP,AX
XCHG AX,BX
XCHG AL,CL
XCHG SI,DI
XCHG [BX+SI],AL
```

1.3 LEA Instruction

The Load effective address (**LEA**) instruction is used to load a specified register with 16-bit offset address.

LEA dest, src ; (dest) ← EA

Instruction Operands:

LEA reg16, mem16

Ex:

LEA SI,DATA ;SI ← DATA

LEA BX,TEAMP[SI+BP] ; BX ← TEMP+SI+BP

1.4 LDS and LES Instructions

LDS and **LES** instructions load the specified register as well as the DS and ES segment register, respectively. That is they are able to load complete address pointer that is stored in memory. In this way, executing a single instruction can activate a new data segment.

LDS dest, src ; (dest) ← (EA)
(DS) ← (EA + 2)

LES dest, src ; (dest) ← (EA)
(ES) ← (EA + 2)

Instruction Operands:

LDS reg16, mem32

Ex:

OS DW 0AFFAH,05566H

LDS BX,DWORD PTR OS; BX ← 0AFFAH, DS ← 05566H

2. Arithmetic instructions

2.1 INC, ADD, and ADC Instructions

Increment instruction (**INC**) adds one to the destination operand. The operand may be byte or a word and is treated as an unsigned binary number.

INC dest ; (dest) ← (dest) + 1

Flags Affected:

OF,SF,ZF,AF,PF

Instruction Operands:

INC reg

INC mem

Ex:

```
INC AX
INC AL
INC [SI+BP]
INC TEMP
```

Addition (**ADD**) instruction sums two operands, which may be bytes or words, replaces the destination operand. Both operands may be signed or unsigned binary numbers.

ADD dest, src ;(dest) $\leftarrow$ (dest) + (src)

Flags Affected:

CF,OF,SF,ZF,AF,PF

Instruction Operands:

```
ADD reg, reg
ADD reg, mem
ADD mem, reg
ADD reg, immed
ADD mem, immed
ADD accum, imme
```

Ex:

```
ADD AX,SI
ADD TEMP,10
ADD AL,DH
ADD AX,[SI+BP]
```

Add with Carry (**ADC**) instruction sums the operands, which may be bytes or words, adds one if CF is set and replaces the destination operand with the result. Both operands may be signed or unsigned binary umbers. Since ADC incorporates a carry from a previous operation, it can be used to write routines to add numbers longer than 16 bits.

```
ADC dest, src; if (CF) = 1
                    then
                        (dest) ← (dest) + (src) + 1
                    else
                        (dest) ← (dest) + (src)
```

Flags Affected:

CF,OF,SF,ZF,AF,PF

Instruction Operands:

ADC reg, reg
ADC reg, mem
ADC mem, reg
ADC reg, immed
ADC mem, immed
ADC accum, immed

Ex:

ADC AX,SI
ADC TEMP,10
ADC AL,DH
ADC AX,[SI+BP]

2.2 DEC, SUB, SBB and NEG Instructions

Decrement (**DEC**) instruction works same as INC but subtract one instead of adding one. Subtract (**SUB)** and subtract with borrow (**SBB)** works same as ADD and ADC but subtract the source from the destination.

DEC dest; (dest)← (dest)-1
SUB dest,src; (dest)←(dest)-(src)
SBB dest,src; (dest)←(dest)-(src)-(CF)

Negate (**NEG**) instruction subtracts the destination operand, which may be a byte or a word, from 0 and returns the result to the destination. This forms the two's complement of the number, effectively reversing the sign of an integer. If the operand is zero, its sign is not changed. Attempting to negate a byte containing –128 or a word containing –32,768 causes no change to the operand and sets OF.

NEG dest;

When Source Operand is a Byte:
(dest) ← FFH – (dest)
(dest) ← (dest) + 1 (affecting flags)

When Source Operand is a Word:
(dest)← FFFFH – (dest)
(dest) ← (dest) + 1 (affecting flags)

Flags Affected:

CF,OF,SF,ZF,AF,PF

Instruction Operands:

NEG reg

NEG mem

Ex:

NEG AX

NEG temp

3. Logical instructions (AND, OR, XOR and NOT Instruction)

Logical And (**AND**) instruction performs the logical "and" of the two operands (byte or word) and returns the result to the destination operand. A bit in the result is set if both corresponding bits of the original operands are set; otherwise the bit is cleared.

```
AND dest, src;      (dest) ← (dest) and (src)
                    (CF) ← 0
                    (OF) ← 0
```

Logical Or (**OR**) instruction performs the logical "inclusive or" of the two operands (bytes or words) and returns the result to the destination operand. A bit in the result is set if either or both corresponding bits in the
original operands are set; otherwise the result bit is cleared.

```
OR dest, src;       (dest) ← (dest) or (src)
                    (CF) ← 0
                    (OF) ← 0
```

Exclusive Or (**XOR**) instruction performs the logical "exclusive or" of the two operands and returns the result to the destination operand. A bit in the result is set if the corresponding bits of the original operands contain opposite values (one is set, the other is cleared); otherwise the result bit is cleared.

```
XOR dest, src;      (dest) ← (dest) xor (src)
                    (CF) ← 0
                    (OF) ← 0
```

Flags Affected:

CF,OF,SF,ZF,PF

AF undefined

Instructions Operands:

reg, reg
reg, mem
mem, reg
accum, immed
reg, immed
mem, immed

Logical Not (**NOT**) instruction Inverts the bits (forms the one's complement) of the byte or word operand.

NOT dest

Instruction Operands:

NOT reg
NOT mem

Flags Affected:

None

Ex:

```
AND AX,TEMP
OR AX,BX
NOT AX
XOR SI,AX
AND [SI],AL
```

4. The Shift Instructions: SHL/SAL, SHR and SAR

The 8086 supports three different shift instructions (SHL and SAL are the same instruction): SHL (shift left), SAL (shift arithmetic left), SHR (shift right), and SAR (shift arithmetic right). The shift instructions move bits around in a register or memory location. The general format for a shift instruction is

SHL dest, count
SAL dest, count
SHR dest, count
SAR dest, count

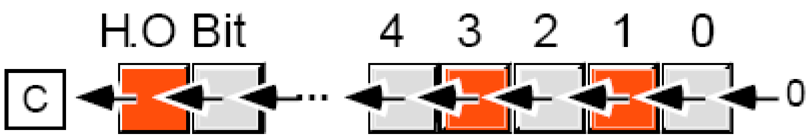
Flags Affected:

CF,OF,SF,ZF,PF
AF undefined

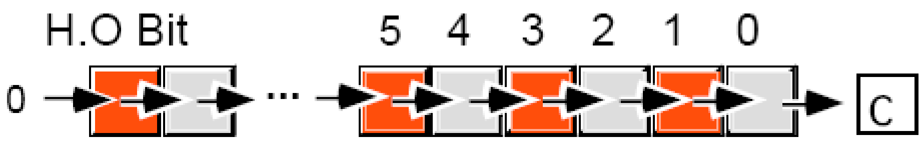
Instructions Operands:

reg, 1
reg, CL
mem, 1
mem, CL

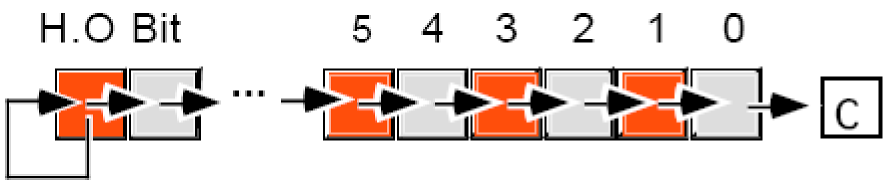
Note:TASM accept immediate "count" more than one, but it is really repeat the instruction.



Shift Left Operation (SHL and SAL)



Shift Right Operation (SHR)



Arithmetic Shift Right Operation (SAR)

Ex:

SHL AL,1
SAR BX,CL
SHR [SI],1
SHL TEMP,CL
SAL [SI+BX],CL

5. The Rotate Instructions: RCL, RCR, ROL, and ROR

The rotate instructions shift the bits around, just like the shift instructions, except the bits shifted out of the operand by the rotate instructions recirculate through the operand. They include RCL (rotate through carry left), RCR (rotate through carry right), ROL (rotate left), and ROR (rotate right). These instructions all take the forms:

ROL dest, count
ROR dest, count
RCL dest, count
RCR dest, count

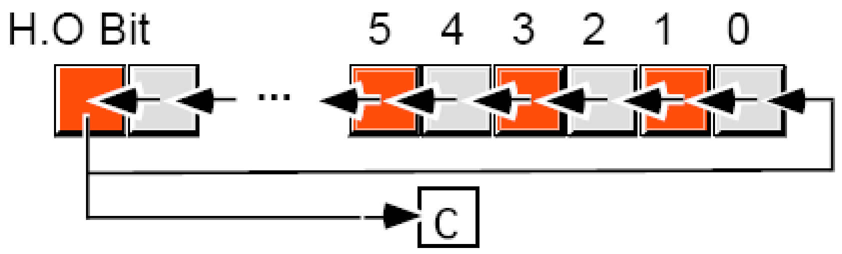
Flags Affected:

CF,OF

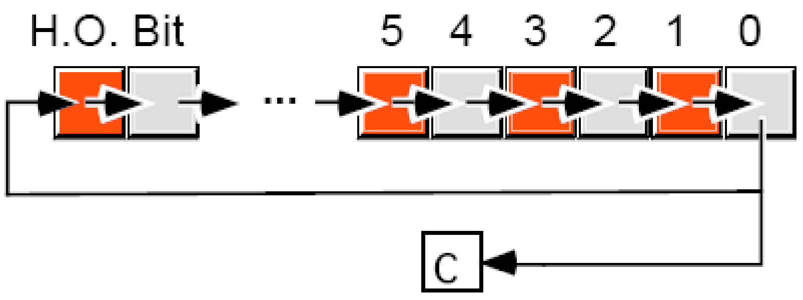
Instructions Operands:

reg, 1
reg, CL
mem, 1
mem, CL

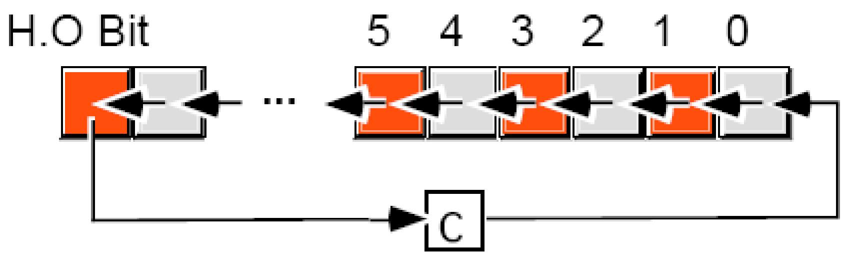
Note:TASM accept immediate "count" more than one, but it is really repeat the instruction.



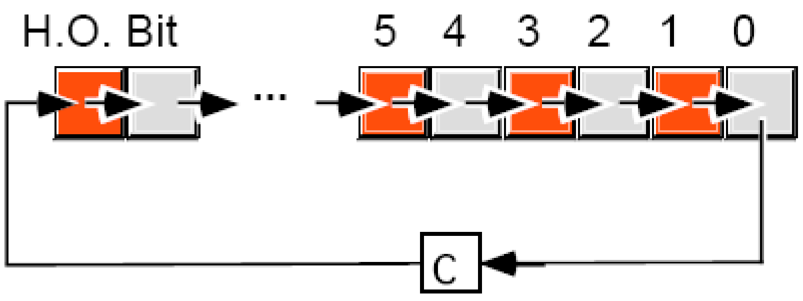
Rotate Left Operation (ROL)



Rotate Right Operation (ROR)



Rotate Through Carry Left Operation (RCL)



Rotate Through Carry Right Operation (RCR)

Ex:

```
ROL AL,1
ROR BX,CL
RCR [SI],1
RCL TEMP,CL
RCL [SI+BX],CL
```

The CMP Instruction

The cmp (compare) instruction is identical to the sub instruction with one crucial difference – it does not store the difference back into the destination operand. The syntax for the cmp instruction is very similar to sub, the generic form is

```
CMP dest, src
```

The CMP instruction updates the 8086's flags according to the result of the subtraction operation (dest - src). The result of the comparison can be tested by checking the appropriate flags in the flags register.

The TEST Instruction

The test instruction logically and its two operands and sets the flags but does not save the result. Test and AND share the same relationship as CMP and SUB. Typically, this instruction can be used to see if a bit contains one. the generic form is

```
TEST dest, src
```

Consider the following instruction:

```
TEST AL, 1
```

This instruction logically ANDs AL with the value one. If bit zero of AL contains a one, the result is non-zero and the 8086 clears the zero flag. If bit zero of al contains zero, then the result is zero and the test operation sets the zero flag.