University of Technology- Computer Science

Microprocessor

Third Class

2023-2024

Asst. Prof Dr. Raheem Abdul Sahib

Micro-processors – 1’st course

- Introduction to Microprocessor and Microcomputer system.
  - Microprocessor Architecture and Register Set.
  - System Buses
  - Memory types and physical addressing.
  - I/O devices
- Instruction Set and Format
- Addressing Modes
- Introduction to Assembly Programming Language.
  - Arithmetic and logical Instructions (Shift and Rotate).
  - Program Control (interrupt and subroutine call).

References:

1. Abel P., "IBM PC Assembly Language and Programming", 4th Edition, Prentice Hall, 1998..
2. Thorne M., "Computer Organization and Assembly Language Programming", 2nd Edition, Benjamin/Cummings, 1990.

1. Introduction to Microprocessor and Microcomputer system.

1.1 What Is an Integrated Circuit?

Integrated circuit or IC or microchip or chip is a microscopic electronic circuit array formed by the fabrication of various electrical and electronic components (resistors, capacitors, transistors, and so on) on a semiconductor material (silicon) wafer, which can perform operations similar to the large discrete electronic circuits made of discrete electronic components. See figure 1

Figure 1: Integrated Circuit

1.2 What is a Microprocessor?

Computer's Central Processing Unit (CPU) built on a single Integrated Circuit (IC) is called a microprocessor. A digital computer with one microprocessor which acts as a CPU is called microcomputer (See figure 2). It is a programmable, multipurpose, clock -driven, register-based electronic device that reads binary instructions from a storage device called memory, accepts binary data as input and processes data according to those instructions and provides results as output.

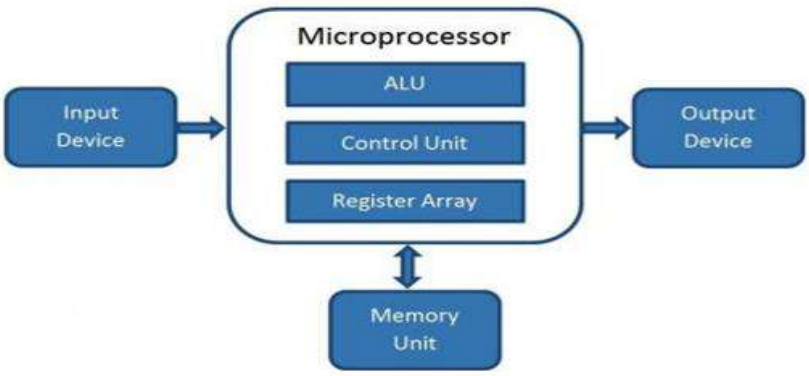


Figure 2: Block Diagram of a Microcomputer

The microprocessor contains millions of tiny components like transistors, registers, and diodes that work together. A microprocessor consists of an ALU, Control Unit and Register array:

1. **ALU**: performs arithmetic and logical operations on the data received from an input device or memory.
2. **Control unit: controls** the instructions and flow of data within the computer.
3. **Register array:**consists of registers identified by letters like B, C, D, E, H, L, and accumulator.

1.3 Evolution of Microprocessors (التطور التاريخي للمعالجات الماكروية)

We can categorize the microprocessor according to the generations or according to the size of the microprocessor:

1. **First Generation (4 - bit Microprocessors)**
   The first generation microprocessors were introduced in the year 1971-1972 by Intel Corporation. It was named Intel 4004 since it was a **4-bit processor.**
   It was a processor on a single chip. It could perform simple arithmetic and logical operations such as addition, subtraction, Boolean OR and Boolean AND.
   I had a control unit capable of performing control functions like fetching an instruction from storage memory, decoding it, and then generating control pulses to execute it.
2. **Second Generation (8 - bit Microprocessor)**
   The second generation microprocessors were introduced in 1973 again by Intel. It was a first
   8-bit microprocessorwhich could perform arithmetic and logic operations on 8-bit words. It was Intel 8008, and another improved version was Intel 8088.
3. **Third Generation (16 - bit Microprocessor)**
   The third generation microprocessors, introduced in 1978 were represented by Intel's 8086, Zilog Z800 and 80286, which **were 16 - bit processors**with a performance like inicomputers.
4. **Fourth Generation (32 - bit Microprocessors)**
   Several different companies introduced the **32-bit microprocessors**, but the most popular one is the Intel 80386.
5. Fifth Generation (64 - bit Microprocessors)
   From 1995 to now we are in the fifth generation. After 80856, Intel came out with a new processor namely Pentium processor followed by **Pentium Pro CPU**, which allows multiple CPUs in a single system to achieve multiprocessing.
   Note: Other improved 64-bit processors are **Celeron, Dual, Quad, Octa Core processors**.

Table 1: Important Intel Microprocessors

Microprocessor	Year of Invention	Word Length	Memory addressing Capacity	Pins	Clock	Remarks
4004	1971	4-bit	1 KB	16	750 KHz	First Microprocessor
8085	1976	8-bit	64 KB	40	3-6 MHz	Popular 8-bit Microprocessor
8086	1978	16-bit	1MB	40	5-8 MHz	Widely used in PC/XT
80286	1982	16-bit	16MB real, 4 GB virtual	68	6-12.5 MHz	Widely used in PC/AT
80386	1985	32-bit	4GB real, 64TB virtual	132 14X14 PGA	20-33 MHz	Contains MMU on chip
80486	1989	32-bit	4GB real, 64TB virtual	168 17X17 PGA	25-100 MHz	Contains MMU, cache and FPU, 1.2 million transistors
Pentium	1993	32-bit	4GB real,32-bit address,64-bit data bus	237 PGA	60-200	Contains 2 ALUs,2 Caches, FPU, 3.3 Million transistors, 3.3 V, 7.5 million transistors
Pentium Pro	1995	32-bit	64GB real, 36-bit address bus	387 PGA	150-200 MHz	It is a data flow processor. It contains second level cache also,3.3 V
Pentium II	1997	32-bit	-	-	233-400 MHz	All features Pentium pro plus MMX technology,3.3 V, 7.5 million transistors
Pentium III	1999	32-bit	64GB	370 PGA	600-1.3 MHz	Improved version of Pentium II; 70 new SIMD
Pentium 4	2000	32-bit	64GB	423 PGA	600-1.3 GHz	Improved version of Pentium III
Itanium	2001	64-bit	64 address lines	423 PGA	733 MHz-1.3 GHz	64-bit EPIC Processor

1.4 List of Terms Used in a Microprocessor

Here is a list of some of the frequently used terms in a microprocessor−

- **Instruction Set**− It is the set of instructions that the microprocessor can understand and given to the microprocessor. The instruction set acts as an interface between the software and hardware.
- **Bus**- The bus is used for the transmission of data, address and control information. This transmission occurs in different elements of the microprocessor. The bus in this is basically of three types which are data bus, a control bus, and address bus.
- **Bandwidth**− It is the number of bits processed in a single instruction.
- **Clock Speed**− It determines the number of operations per second the processor can perform. It is expressed in megahertz (MHz) or gigahertz (GHz). It is also known as Clock Rate.
- **Word Length**− It depends upon the (number of bits) of internal data bus, registers, ALU, etc. An 8-bit microprocessor can process 8-bit data at a time. The word length ranges from 4 bits to 64 bits depending upon the type of the microcomputer. A processor with longer word length is more powerful and can process data at a faster speed as compared to processor with shorter word length
- **IPC (Instructions Per Cycle)** - It is a measure of how many instructions a CPU is capable of executing in a single clock.
- **Data Types**– The microprocessor has multiple data type formats like binary, BCD, ASCII, signed and unsigned numbers.

1.5 Working of Microprocessor

The microprocessor follows a sequence to execute the instruction:**Fetch**, **Decode**, and then **Execute**.

Initially, the instructions are stored in the storage memory of the computer in sequential order. The microprocessor fetches those instructions from the stored area (memory), then decodes it and executes those instructions till STOP instruction is met. Then, it sends the result in binary form to the output port. Between these processes, the register stores the temporary data and ALU (Arithmetic and Logic Unit) performs the computing functions.

1.5.1 How does a Microprocessor Work?

- A computer can only do what the programmer has told it to do, in the form of a program.
- A program is just a sequence of very simple commands that lead the computer to solve some problem.
- Once the program is written and debugged, the computer can execute the instructions very fast, and always do it the same, every time, without a mistake.
- The microprocessor itself is usually a single integrated circuit (IC).
- Most microprocessors (MPU), or very small computers, have much the same commands or instructions that they can perform.

- In a typical MPU, there are commands to move data around, do simple math (add, subtract, multiply, and divide), bring data into the micro from the outside world, and send data out of the micro to the outside world.
- A microprocessor executes a program by repeatedly cycling through following three basic steps: Fetch, Decode, and then Execute, see figure 3
  - Initially, the instructions are stored in the memory in a sequential order. The microprocessor fetches those instructions from the memory,
  - then decodes it and executes those instructions till STOP instruction is reached.
  - **Later**, it sends the result in binary to the output port. Between these processes, the register stores the temporarily data and ALU performs the computing functions.
    NOT: The power of the given microprocessor is measured in terms of bits.
  - The processor executes the instruction using the following steps: Figure 3: Workflow of instructions execution
  - This sequence is continued until all instructions are performed

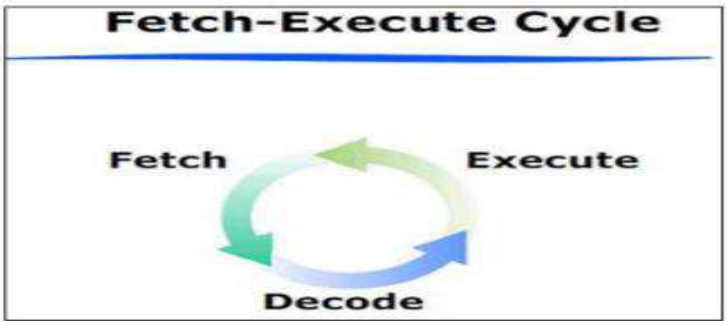


Figure 3: Workflow of instructions execution

1.6 Features of Microprocessor

- **Low Cost** - Due to integrated circuit technology microprocessors are available at very low cost. It will reduce the cost of a computer system.
- **High Speed** - Due to the technology involved in it, the microprocessor can work at very high speed. It can execute millions of instructions per second.
- **Small Size** - A microprocessor is fabricated in a very less footprint due to very large scale and ultra large scale integration technology. Because of this, the size of t he computer system is reduced.
- **Versatile** - The same chip can be used for several applications, therefore, microprocessors are versatile.
- **Low Power Consumption** - Microprocessors are using metal oxide semiconductor technology, which consumes less power.
- **Less Heat Generation** - Microprocessors uses semiconductor technology which will not emit much heat as compared to vacuum tube devices.
- **Reliable** - Since microprocessors use semiconductor technology, therefore, the failure rate is very less. Hence it is very reliable.
- **Portable** - Due to the small size and low power consumption microprocessors are portable.

1.7 Instructions Execution

A microprocessor executes a collection of machine instructions that tell the processor what to do. Based on the instructions, a microprocessor does three basic things:

- Using its ALU (Arithmetic/Logic Unit), a microprocessor can perform mathematical operations like addition, subtraction, multiplication and division. Modern microprocessors contain complete floating-point processors that can perform extremely sophisticated operations on large floating-point numbers.
- A microprocessor can move data from one memory location to another.
- A microprocessor can make decisions and jump to a new set of instructions based on those decisions.

This is about as simple as a microprocessor gets. This microprocessor has:

- ✓ An address bus (that may be 8, 16, 32 or 64 bits wide) that sends an address to memory
- ✓ A data bus (that may be 8, 16, 32 or 64 bits wide) that can send data to memory or receive data from memory
- ✓ An RD (read) and WR (write) line to tell the memory whether it should set or get the addressed location
- ✓ A clock line that lets a clock pulse sequence the processor
- ✓ A reset line that resets the program counter to zero (or whatever) and restarts execution

2. Types of Microprocessors

There are many types of microprocessor:

1. **Vector Processors: A vector processor is designed for vector computations. A vector is an array of operands of the same type. Consider the following vectors:**

   Vector A (a_1, a_2, a_3, ..., a_n), Vector B (b_1, b_2, b_3,....., b_n)

   Examples of vector processors are:

   DEC's VAX 9000, IBM 390/VF, CRAY Research Y-MP family, Hitachi's S-810/20, etc

2. **Array Processors or SIMD Processors: Array processors** are also designed for vector computations. **The difference between** an array processor and a vector processor is that a vector processor uses multiple vector pipelines whereas an array processor employs a number of processing elements to operate in parallel. An array processor contains multiple numbers of ALUs. Each ALU is provided with the local memory. The ALU together with the local memory is called a Processing Element (PE). An array processor is a **SIMD (Single Instruction Multiple Data)** processor. Thus using a single instruction, the same operation can be performed on an array of data which makes it suitable for vector computations.

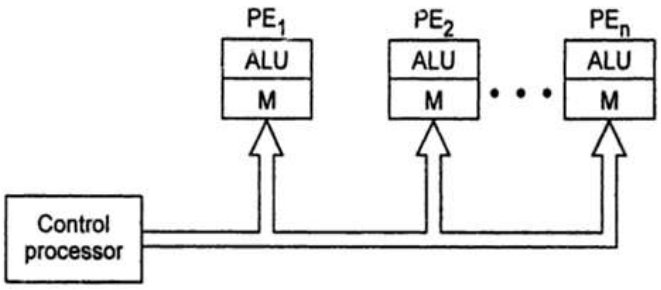


Figure 4: Schematic Diagram of an Array Processor or SIMD

3. Scalar and Superscalar Processors

A processor that executes scalar data is called scalar processor. The simplest scalar processor makes processing of only integer instruction using fixed-points operands. A powerful scalar processor makes processing of both integer as well floating- point numbers. It contains an integer ALU and a **Floating Point Unit (FPU)** on the same CPU chip.

A scalar processor may be RISC processor or CISC processor.

Examples of CISC processors are:

- Intel 386, 486; Motorola's 68030, 68040; etc.

Examples of RISC scalar processors are:

- Intel i860, Motorola MC8810, SUN's SPARC CY7C601, etc.

A **superscalar processor** has multiple pipelines and executes more than one instruction per clock cycle.

- **Examples of superscalar processors are:** Pentium, Pentium Pro, Pentium II, Pentium III,

4. RISC and CISC Processors

RISC stands for **Reduced Instruction Set Computer and**
CISC stands for **Complex Instruction Set Computer.**
There are two approaches of the design of the control unit of a microprocessor i.e.-

- Hardware approach and Software approach.

Q:What are the main differences between RISC and CISC?

S.No.	RISC	CISC
1.	Simple instruction set	Complex instruction set
2.	Consists of Large number of registers.	Less number of registers
3.	Larger Program	Smaller program
4.	Simple processor circuitry (small number of transistors)	Complex processor circuitry (more number of transistors)
5.	More RAM usage	Little Ram usage
6.	Simple addressing modes	Variety of addressing modes
7.	Fixed length instructions	Variable length instructions
8.	Fixed number of clock cycles for executing one instruction	Variable number of clock cycles for each instructions

What is an 8085 Microprocessor?
Answer: It is an 8-bit microprocessor that can process, accept and send 8-bit data simultaneously. It operates on a 3MHz clock frequency and has 16 address lines so it has 64-kilo bytes of memory
What is the 8086 microprocessor?
Answer: It is a 16-bit microprocessor having 20 address lines and 16 data lines along with 1MB of memory storage. The processing is faster than that of the 8085 microprocessor.

Computer Abstraction Hierarchy

Hardware or software component	How this component affects performance
Algorithm	Determines both the number of source-level statements and the number of 1/O operations executed
Programming language, compiler, and architecture	Determines the number of computer instructions for each source-level statement
Processor and memory system	Determines how fast instructions can be executed
I/O system (hardware and operating system)	Determines how fast 1/O operations may be executed

The performance of a program depends on a combination of the effectiveness of the algorithms used in the program, the software systems used to create and translate the program into machine instructions, and the effectiveness of the computer in executing those instructions, which may include input/output (I/O) operations.

If we open the box containing the computer, we see a fascinating board of thin plastic, covered with dozens of small gray or black rectangles.

Figure 5 shows the contents of the Desktop computer and Figure 6 shows the inside of laptop computer

In figure 6 , The motherboard is shown in the upper part of the photo. Two disk drives are in front—the hard drive on the left and a DVD drive on the right. The hole in the middle is for the laptop battery. The small rectangles on the motherboard contain the devices that drive our advancing technology, called integrated circuits and nicknamed chips.

The board is composed of three pieces: the piece connecting to the I/O devices mentioned earlier, the memory, and the processor. The memory is where the programs are kept when they are running; it also contains the data needed by the running programs. Figure 7 shows that memory is found on the two small boards, and each small memory board contains eight integrated circuits.

.

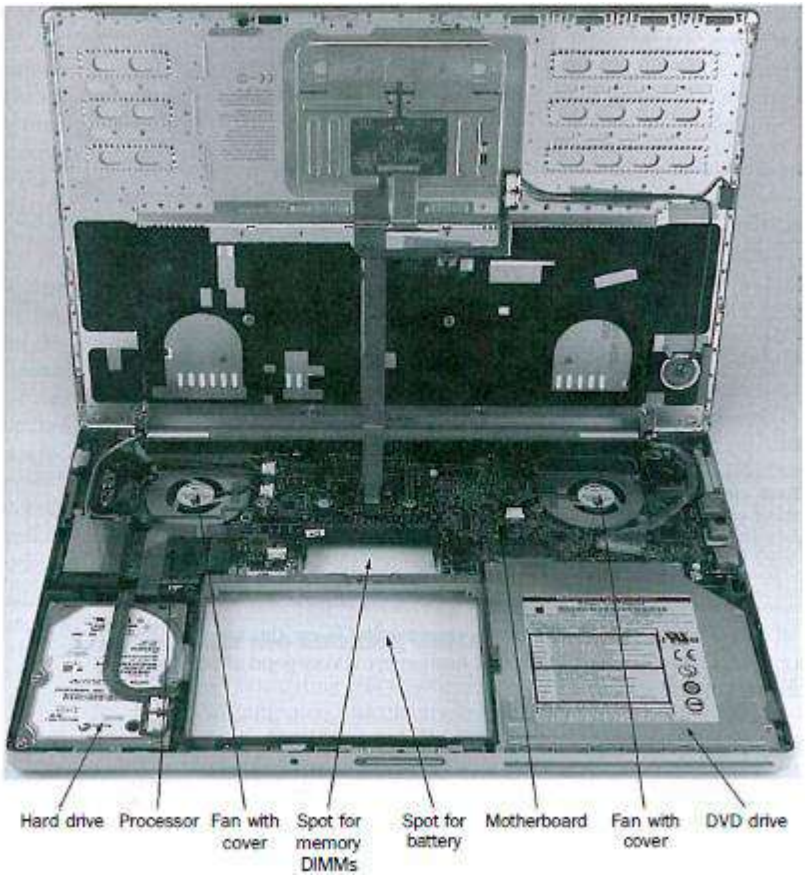


Figure 5: Desktop computer

DRAM stands for Dynamic Random Access Memory. Several DRAMs are used together to contain the instructions and data of a program. In contrast to sequential access memories, such as magnetic tapes, the *RAM* portion of the term DRAM means that memory accesses take basically the same amount of time no matter what portion of the memory is read.

5. Motherboard

A plastic board containing packages of integrated circuits or chips, including processor, cache, memory, and connectors for I/O devices such as networks and disks. integrated circuit Also called a chip. A device combining dozens to millions of transistors. Memory is the storage area in which programs are kept when they are running and that contains the data needed by the running programs

5.1 Dynamic Random Access memory (DRAM)

Memory built as an integrated circuit; it provides random access to any location.

The *processor:* ***it*** is the active part of the board, following the instructions of a program to the letter. It adds numbers, tests numbers, signals I/O devices to activate, and so on. The processor is under the fan and covered by a heat sink on the left side of Figure 1.7. Occasionally, people call the processor the CPU or central processor unit. Descending even lower into the hardware, Figure 1.9 reveals details of a microprocessor. The processor logically comprises two main components**: data path** and **control**, the respective brawn and brain of the processor. The data path performs the arithmetic operations, and control tells the

data path, memory, and I/O devices what to do according to the wishes of the instructions of the program.

5.2 Central Processor unit (CPU)

Also called processor. The active part of the computer, which contains the data path and control and which adds numbers, tests numbers, signals I/O devices to activate, and so on. Data path The component of the processor that performs arithmetic operations, control is the component of the processor that commands the data path, memory, and I/O devices.

One of the most important abstractions is the interface between the hardware and the lowest-level software abstraction A mode l that renders lower-level details of computer systems temporarily invisible to facilitate design of sophisticated systems.

name: the **instruction set architecture,** or simply **architecture,** of a computer. The instruction set architecture includes anything programmers need to know to make a binary machine language program work correctly, including instructions, I/O devices, and so on. Typically, the operating system will encapsulate the details of doing I/O, allocating memory, and other low-level system functions so that application programmers do not need to worry about such details. The combination of the basic instruction set and the operating system interface provided for application programmers is called the **application binary interface (ABI).**

instruction set architecture Also called architecture. An abstract interface between the hardware and the lowest-level software that encompasses all the information necessary to write a machine language program that will run correctly, including instructions, registers, memory access, I/O.

An instruction set architecture allows computer designers to talk about functions independently from the hardware that performs them. For example, we can talk about the functions of a digital clock (keeping time, displaying the time, setting the alarm) independently from the clock hardware (quartz crystal, LED displays, plastic buttons).

Computer **implementation** of an architecture along the same lines: an implementation is hardware that obeys the architecture abstraction.

Both hardware and software consist of hierarchical layers, with each lower layer hiding details from the level above. This principle of *abstraction* is the way both hardware designers and software designers cope with the complexity of computer systems. One key interface between the levels of abstraction is the *instruction set architecture*—the interface between the hardware and low-level software. This abstract interface enables many *implementations* of varying cost and performance to run identical software.

5.3 Technologies for Processors (PERFORMANCE MEASURES)

Processors and memory have improved at an incredible rate, because computer designers have long embraced the latest in electronic technology to try to win the race to design a better computer. Figure 7 shows the technologies that have been used over time

5.3.1 Performance

we need to determine the time taken by a computer to execute a given job. We define the clock cycle time as the time between two consecutive rising (trailing) edges of a periodic clock signal (Fig. 7).

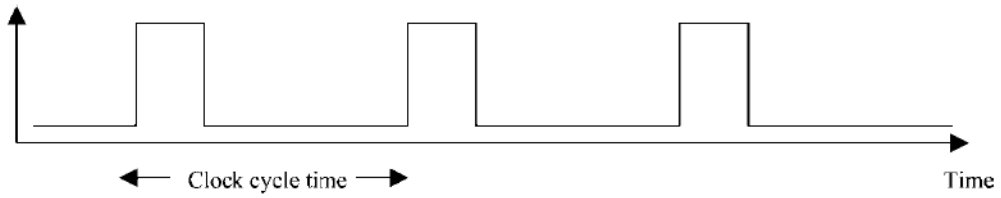


Figure 7: clock signal

Clock cycles allow counting unit computations, because the storage of computation results is synchronized with rising (trailing) clock edges. The time required to execute a job by a computer is often expressed in terms of **clock cycles**. We denote the number of CPU clock cycles for executing a job to be the cycle count (CC), the cycle time by CT, and the clock frequency by f =1/CT. The time taken by the CPU to execute a job can be expressed as
CPU time = CC × CT = CC/f
It may be easier to count the number of instructions executed in a given program as compared to counting the number of CPU clock cycles needed for executing that program.

5.3.2 Defining Performance

we can define computer performance in several different ways.
If you were running a program on two different desktop computers, you'd say that the faster one is the desktop computer that gets the job done first. If you were running datacenter that had several servers running jobs submitted by many users, you'd say that the faster computer was the one that completed the most jobs during a day.
As an individual computer user, you are interested in reducing response time—the time between the start and completion of a task—also referred to as execution time. Datacenter managers are often interested in increasing throughput or bandwidth—the total amount of work done in a given time. Hence, in most cases, we will need **different performance metrics** as well as different sets of applications to benchmark embedded and desktop computers, which are more focused on response time, versus servers, which are more focused on throughput.
Therefore, the average number of clock cycles per instruction (CPI) has been used as an alternate performance measure. The following equation shows how to compute the CPI.

$$CPI = \frac{CPU \text{ clock cycles for the program}}{\text{Instruction count}}$$

$$CPU \text{ time} = \text{Instruction count} \times \text{CPI} \times \text{Clock cycle time}$$

$$= \frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}}$$

It is known that the instruction set of a given machine consists of a number of instruction categories: ALU (simple assignment and arithmetic and logic instructions), load, store, branch, and so on. In the case that the CPI for each instruction category is known, the overall CPI can be computed as:

$$CPI = \frac{\sum_{i=1}^{n} CPI_i \times I_i}{\text{Instruction count}}$$

where Ii is the number of times an instruction of type i is executed in the program and CPIi is the average number of clock cycles needed to execute such instruction.

Example: Consider computing the overall CPI for a machine A for which the following performance measures were recorded when executing a set of benchmark programs. Assume that the clock rate of the CPU is 200 MHz.

Instruction category	Percentage of occurrence	No. of cycles per instruction
ALU	38	1
Load & store	15	3
Branch	42	4
Others	5	5

Assuming the execution of 100 instructions, the overall CPI can be computed as

$$CPI_a = \frac{\sum_{i=1}^{n} CPI_i \times I_i}{Instruction\ count} = \frac{38 \times 1 + 15 \times 3 + 42 \times 4 + 5 \times 5}{100} = 2.76$$

It should be noted that the CPI reflects the organization and the instruction set architecture of the processor while the instruction count reflects the instruction set architecture and compiler technology used. This shows the degree of interdependence between the two performance parameters. Therefore, it is imperative that both the CPI and the instruction count are considered in assessing the merits of a given computer or equivalently in comparing the performance of two machines.

A different performance measure that has been given a lot of attention in recent years is MIPS (million instructions-per-second (the rate of instruction execution per unit time)), which is defined as

$$MIPS = \frac{\text{Instruction count}}{\text{Execution time} \times 10^6} = \frac{\text{Clock rate}}{CPI \times 10^6}$$

Example: Suppose that the same set of benchmark programs considered above were executed on another machine, call it machine B, for which the following measures were recorded.

Instruction category	Percentage of occurrence	No. of cycles per instruction
ALU	35	1
Load & store	30	2
Branch	15	3
Others	20	5

What is the MIPS rating for the machine considered in the previous example (machine A) and machine B assuming a clock rate of 200 MHz, and instruction Account is 100;

$$CPI_a = \frac{\sum_{i=1}^{n} CPI_i \times I_i}{\text{Instruction count}} = \frac{38 \times 1 + 15 \times 3 + 42 \times 4 + 5 \times 5}{100} = 2.76$$

$$\text{MIPS}_{\ a} = \frac{\text{Clock rate}}{CPI_a \times 10^6} = \frac{200 \times 10^6}{2.76 \times 10^6} = 70.24$$

$$CPI_b = \frac{\sum_{i=1}^{n} CPI_i \times I_i}{\text{Instruction count}} = \frac{35 \times 1 + 30 \times 2 + 20 \times 5 + 15 \times 3}{100} = 2.4$$

$$\text{MIPS}_{\ b} = \frac{\text{Clock rate}}{CPI_a \times 10^6} = \frac{200 \times 10^6}{2.4 \times 10^6} = 83.67$$

Thus $MIPS_b > MIPS_a$.

It is interesting to note here that although MIPS has been used as a performance measure for machines, one has to be careful in using it to compare machines having different instruction sets. This is because MIPS does not track execution time. Consider, for example, the following measurement made on two different machines running a given set of benchmark programs.

Suppose that you have a set of benchmark programs of two machines (A and B) that were executed on both machines for which the following measures were recorded.

	Machine A		Machine B	
Instruction Category	**Percentage of Occurrence**	**No. of Cycles Per Instruction**	**Percentage of Occurrence**	**No. of Cycles Per Instruction**
CPU	**38**	**1**	**35**	**1**
Load and Store	**15**	**3**	**30**	**2**
Brach	**42**	**4**	**15**	**3**
Others	**5**	**5**	**20**	**5**

What is the MIPS rating for the machines considered in the table above for machines (A, B) assuming a clock rate of 200 MHz?

Computer Microprocessor (Part Two)

Lecturers:
Ass. Prof. Shaimaa Hameed Shaker
And Asst. Prof. Dr. Raheem Abdul Sahib Olga

2022-2023

Computer Science Dept.

References

1. "**fundamentals of computer organization and architecture**", by Mostafa Abd-El-Barr Hesham El-Rewini, 1PstP edition/2005.
2. David A. Patterson and John L. Hennessy, **"Computer Organization and Design"**,1998.
3. M. M. Mano, **"computer system architecture"** third edition, prentice Hall, 1993.
4. Walter A. Triebel, **"The 80386, 80486, and Pentium® Processors Hardware,Software, and Interfaceing",** 1998.

2.Microprocessor - 8086 Overview

8086 Microprocessor is an enhanced version of 8085 Microprocessor that was designed by Intel in 1976. It is a 16-bit Microprocessor having 20 address lines and16 data lines that provides up to 1MB storage. It consists of powerful instruction set, which provides operations like multiplication and division easily.
It supports two modes of operation, i.e. **Maximum mode and Minimum mode**. Maximum mode is suitable for system having multiple processors and Minimum mode is suitable for system having a single processor.

2.1 Features of 8086

The most prominent features of an 8086 microprocessor are as follows −

- It has an instruction queue, which is capable of storing six instruction bytes from the memory resulting in faster processing.
- It was the first 16-bit processor having 16-bit ALU, 16-bit registers, internal data bus, and 16-bit external data bus resulting in faster processing.
- It is available in 3 versions based on the frequency of operation −
  - 8086 → 5MHz
  - 8086-2 → 8MHz
  - (c)8086-1 → 10 MHz
- It uses two stages of pipelining, i.e. Fetch Stage and Execute Stage, which improves performance.
- Fetch stage can prefetch up to 6 bytes of instructions and stores them in the queue.
- Execute stage executes these instructions.
- It has 256 vectored interrupts.
- It consists of 29,000 transistors.

2.2 Comparison between 8085 & 8086 Microprocessor

- **Size** − 8085 is 8-bit microprocessor, whereas 8086 is 16-bit microprocessor.
- **Address Bus** − 8085 has 16-bit address bus while 8086 has 20-bit address bus.
- **Memory** − 8085 can access up to 64Kb, whereas 8086 can access up to 1 Mb of memory.
- **Instruction** − 8085 doesn’t have an instruction queue, whereas 8086 has an instruction queue.
- **Pipelining** − 8085 doesn’t support a pipelined architecture while 8086 supports a pipelined architecture.
- **I/O** − 8085 can address 2^8 = 256 I/O's, whereas 8086 can access 2^16 = 65,536 I/O's.
- **Cost** − The cost of 8085 is low whereas that of 8086 is high.

2.3 Architecture of 8086

The following diagram depicts the architecture of an 8086 Microprocessor (Figure 2.1)

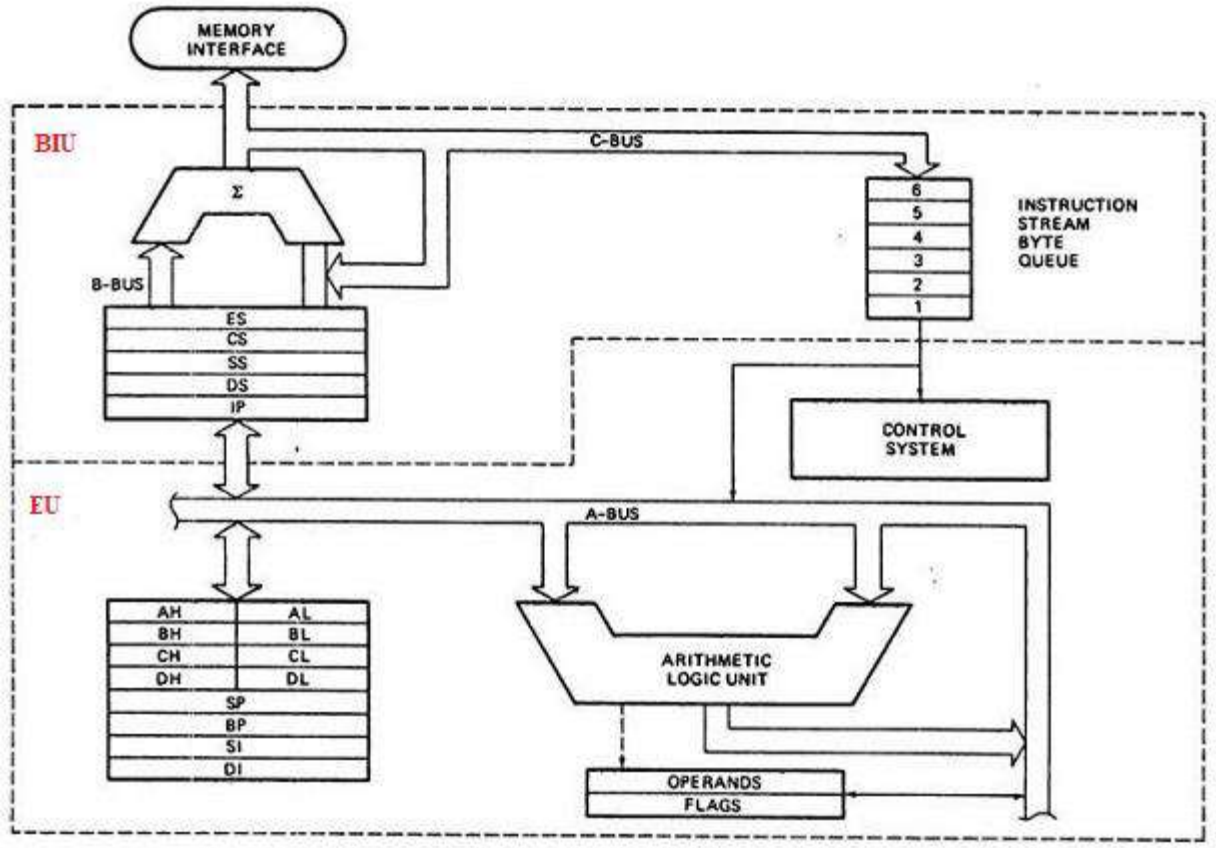


Figure (2.1) Architecture of 8086

2.3.1 Microprocessor - 8086 Functional Units

8086 Microprocessor is divided into two functional units, i.e., **EU** (Execution Unit) and **BIU** (Bus Interface Unit).

A. EU (Execution Unit)

Execution unit gives instructions to BIU stating from where to fetch the data and then decode and execute those instructions. Its function is to control operations on data using the instruction decoder & ALU. EU has no direct connection with system buses as shown in the above figure, it performs operations over data through BIU.

Let us now discuss the functional parts of 8086 microprocessors.

1. ALU

It handles all arithmetic and logical operations, like +, −, ×, /, OR, AND, NOT operations.

2. Flag Register

It is a 16-bit register that behaves like a flip-flop, i.e. it changes its status according to the result stored in the accumulator. It has 9 flags and they are divided into 2 groups − **Conditional Flags** and Control Flags.

1. Conditional Flags

It represents the result of the last arithmetic or logical instruction executed. Following is the list of conditional flags :

- **Carry flag** − This flag indicates an overflow condition for arithmetic operations.
- **Auxiliary flag** − When an operation is performed at ALU, it results in a carry/barrow from lower nibble (i.e. D0 – D3) to upper nibble (i.e. D4 – D7), then this flag is set, i.e. carry given by D3 bit to D4 is AF flag. The processor uses this flag to perform binary to BCD conversion.
- **Parity flag** − This flag is used to indicate the parity of the result, i.e. when the lower order 8-bits of the result contains even number of 1's, then the Parity Flag is set. For odd number of 1's, the Parity Flag is reset.
- **Zero flag** − This flag is set to 1 when the result of arithmetic or logical operation is zero else it is set to 0.
- **Sign flag** − This flag holds the sign of the result, i.e. when the result of the operation is negative, then the sign flag is set to 1 else set to 0.
- **Overflow flag** − This flag represents the result when the system capacity is exceeded.

2. **Control Flags**

Control flags controls the operations of the execution unit. Following is the list of control flags

- **Trap flag** − It is used for single step control and allows the user to execute one instruction at a time for debugging. If it is set, then the program can be run in a single step mode.
- **Interrupt flag** − It is an interrupt enable/disable flag, i.e. used to allow/prohibit the interruption of a program. It is set to 1 for interrupt enabled condition and set to 0 for interrupt disabled condition.
- **Direction flag** − It is used in string operation. As the name suggests when it is set then string bytes are accessed from the higher memory address to the lower memory address and vice-a-versa.

3. **General Purpose Register**

There are 8 general purpose registers, i.e., AH, AL, BH, BL, CH, CL, DH, and DL. These registers can be used individually to store 8-bit data and can be used in pair s to store 16bit data. The valid register pairs are AH and AL, BH and BL, CH and CL, and DH and DL. It is referred to the AX, BX, CX, and DX respectively.

- **AX register** − It is also known as Accumulator registers. It is used to store operands for arithmetic operations.
- **BX register** − It is used as a base register. It is used to store the starting base address of the memory area within the data segment.
- **CX register** − It is referred to as counter. It is used in loop instruction to store the loop counter.
- **DX register** − This register is used to hold I/O port address for I/O instruction.

4. **pointer register**

It is a 16-bit register, which holds the address from the start of the segment to the memory location, where a word was most recently stored on the stack.

B. BIU (Bus Interface Unit)

BIU takes care of all data and addresses transfers on the buses for the EU like sending addresses, fetching instructions from the memory, reading data from the ports and the memory as well as writing data to the ports and the memory. EU has no direction connection with System Buses so this is possible with the BIU. EU and BIU are connected with the Internal Bus.

It has the following functional parts (BIU)

- **Instruction queue** − BIU contains the instruction queue. BIU gets up to 6 bytes of next instructions and stores them in the instruction queue. When EU executes instructions and is ready for its next instruction, then it simply reads the instruction from this instruction queue resulting in increased execution speed.
- Fetching the next instruction while the current instruction executes is called **pipelining**.
- **Segment register** − BIU has 4 segment buses, i.e. CS, DS, SS& ES. It holds the addresses of instructions and data in memory, which are used by the processor to access memory locations. It also contains 1 pointer register IP, which holds the address of the next instruction to executed by the EU.
  - **CS** − It stands for Code Segment. It is used for addressing a memory location in the code segment of the memory, where the executable program is stored.
  - **DS** − It stands for Data Segment. It consists of data used by the program and is accessed in the data segment by an offset address or the content of other register that holds the offset address.
  - **SS** − It stands for Stack Segment. It handles memory to store data and addresses during execution.
  - **ES** − It stands for Extra Segment. ES is additional data segment, which is used by the string to hold the extra destination data.
- **Instruction pointer** − It is a 16-bit register used to hold the address of the next instruction to be executed.

2.4 Microprocessor - 8086 Pin Configuration

8086 was the first 16-bit microprocessor available in 40-pin DIP (Dual Inline Package) chip. Let us now discuss in detail the pin configuration of a 8086 Microprocessor.

2.4.1 8086 Pin Diagram

Here is the pin diagram of 8086 microprocessors (Figure 2.2)

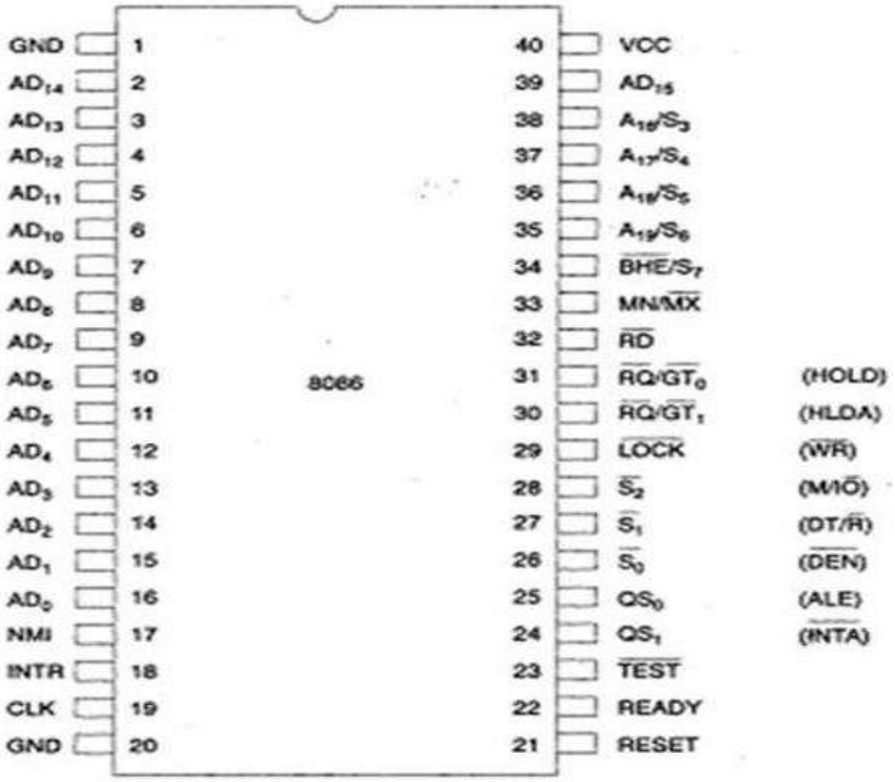


Figure (2.2)8086 Pin Diagram

Let us now discuss the signals in detail −

1. **Power supply and frequency signals**: It uses 5V DC supply at V_{CC} pin 40, and uses ground at V_{SS} pin 1 and 20 for its operation.
2. **Clock signal: Clock** signal is provided through Pin-19. It provides timing to the processor for operations. Its frequency is different for different versions, i.e. 5MHz, 8MHz and 10MHz.
3. **Address/data bus**: AD0-AD15. These are 16 address/data bus. AD0-AD7 carries low order byte data and AD8AD15 carries higher order byte data. During the first clock cycle, it carries 16-bit address and after that it carries 16-bit data.
4. **Address/status bus :**A16-A19/S3-S6. These are the 4 address/status buses. During the first clock cycle, it carries 4-bit address and later it carries status signals.
5. **S7/BHE: BHE** stands for Bus High Enable. It is available at pin 34 and used to indicate the transfer of data using data bus D8-D15. This signal is low during the first clock cycle, thereafter it is active.
6. **Read(*$overline{RD}$*)**It is available at pin 32 and is used to read signal for Read operation.
7. **Ready**: It is available at pin 22. It is an acknowledgement signal from I/O devices that data is transferred. It is an active high signal. When it is high, it indicates that the device is ready to transfer data. When it is low, it indicates wait state.
8. **RESET:** It is available at pin 21 and is used to restart the execution. It causes the processor to immediately terminate its present activity. This signal is active high for the first 4 clock cycles to RESET the microprocessor.
9. **INTR:** It is available at pin 18. It is an interrupt request signal, which is sampled during the last clock cycle of each instruction to determine if the processor considered this as an interrupt or not.
10. **NMI**: It stands for non-mask able interrupt and is available at pin 17. It is an edge triggered input, which causes an interrupt request to the microprocessor.
11. $\overline{TEST}$: This signal is like wait state and is available at pin 23. When this signal is high, then the processor has to wait for IDLE state, else the execution continues.
12. **MN/*$\overline{MX}$:***It stands for Minimum/Maximum and is available at pin 33. It indicates what mode the processor is to operate in; when it is high, it works in the minimum mode and vice-aversa.

13. **INTA:** It is an interrupt acknowledgement signal and id available at pin 24. When the microprocessor receives this signal, it acknowledges the interrupt.
14. **ALE**: It stands for address enable latch and is available at pin 25. A positive pulse is generated each time the processor begins any operation. This signal indicates the availability of a valid address on the address/data lines.
15. **DEN**: It stands for Data Enable and is available at pin 26. It is used to enable Transreceiver 8286. The trans receiver is a device used to separate data from the address/data bus.
16. **DT/R**: It stands for Data Transmit/Receive signal and is available at pin 27. It decides the direction of data flow through the trans receiver. When it is high, data is transmitted out and vice-a-versa.
17. **M/IO**: This signal is used to distinguish between memory and I/O operations. When it is high, it indicates I/O operation and when it is low indicating the memory operation. It is available at pin 28.
18. **WR: It** stands for write signal and is available at pin 29. It is used to write the data into the memory or the output device depending on the status of M/IO signal.
19. **HLDA: It** stands for Hold Acknowledgement signal and is available at pin 30. This signal acknowledges the HOLD signal.
20. **HOLD**: This signal indicates to the processor that external devices are requesting to access the address/data buses. It is available at pin 31.
21. **QS_1 and QS_0** : These are queue status signals and are available at pin 24 and 25. These signals provide the status of instruction queue. Their conditions are shown in the following table

Q S_0	Q S_1	Status
0	0	No operation
0	1	First byte of opcode from the queue
1	0	Empty the queue
1	1	Subsequent byte from the queue

22. **S_0, S_1, S_2**: These are the status signals that provide the status of operation, which is used by the Bus Controller 8288 to generate memory & I/O control signals. These are available at pin 26, 27, and 28. Following is the table showing their status:

S_2	S_1	S_0	Status
0	0	0	Interrupt acknowledgement
0	0	1	I/O Read
0	1	0	I/O Write
0	1	1	Halt
1	0	0	Opcode fetch
1	0	1	Memory read
1	1	0	Memory write
1	1	1	Passive

23. **LOCK:** When this signal is active, it indicates to the other processors not to ask the CPU to leave the system bus. It is activated using the LOCK prefix on any instruction and is available at pin 29.
24. **RQ/GT_1 and RQ/GT_0**These are the Request/Grant signals used by the other processors requesting the CPU to release the system bus. When the signal is received by CPU, then it sends acknowledgment. RQ/GT_0 has a higher priority than RQ/GT_1.

2.4.2 Microprocessor - 8086 Instruction Sets

The 8086 microprocessor supports 8 types of instructions −

1. **Data Transfer Instructions**
2. **Arithmetic Instructions**
3. **Bit Manipulation Instructions**
4. **String Instructions**
5. **Program Execution Transfer Instructions (Branch & Loop Instructions)**
6. **Processor Control Instructions**
7. **Iteration Control Instructions**
8. **Interrupt Instructions**

Let us now discuss these instruction sets in detail.

1. Transfer Instructions

These instructions are used to transfer the data from the source operand to the destination operand. Following are the list of instructions under this group −

A: Instruction to transfer a word

- **MOV** − Used to copy the byte or word from the provided source to the provided destination.
- **PPUSH** − Used to put a word at the top of the stack.
- **POP** − Used to get a word from the top of the stack to the provided location.
- **PUSHA** − Used to put all the registers into the stack.
- **POPA** − Used to get words from the stack to all registers.
- **XCHG** − Used to exchange the data from two locations.
- **XLAT** − Used to translate a byte in AL using a table in the memory.

B. Instructions for input and output port transfer

- **IN** − Used to read a byte or word from the provided port to the accumulator.
- **OUT** − Used to send out a byte or word from the accumulator to the provided port.

C: Instructions to transfer the address

- **LEA** − Used to load the address of operand into the provided register.
- **LDS** − Used to load DS register and other provided register from the memory
- **LES** − Used to load ES register and other provided register from the memory.

D: Instructions to transfer flag registers

- **LAHF** − Used to load AH with the low byte of the flag register.
- **SAHF** − Used to store AH register to low byte of the flag register.
- **PUSHF** − Used to copy the flag register at the top of the stack.
- **POPF** − Used to copy a word at the top of the stack to the flag register.

2. Arithmetic Instructions

These instructions are used to perform arithmetic operations like addition, subtraction, multiplication, division, etc. Following is the list of instructions under this group:

A: Instructions to perform addition

- **ADD** − Used to add the provided byte to byte/word to word.
- **ADC** − Used to add with carry.
- **INC** − Used to increment the provided byte/word by 1.

- **AAA** − Used to adjust ASCII after addition.
- **DAA** − Used to adjust the decimal after the addition/subtraction operation.

B: Instructions to perform subtraction

- **SUB** − Used to subtract the byte from byte/word from word.
- **SBB** − Used to perform subtraction with borrow.
- **DEC** − Used to decrement the provided byte/word by 1.
- **NPG** − Used to negate each bit of the provided byte/word and add 1/2's complement.
- **CMP** − Used to compare 2 provided byte/word.
- **AAS** − Used to adjust ASCII codes after subtraction.
- **DAS** − Used to adjust decimal after subtraction.

C: Instruction to perform multiplication

- **MUL** − Used to multiply unsigned byte by byte/word by word.
- **IMUL** − Used to multiply signed byte by byte/word by word.
- **AAM** − Used to adjust ASCII codes after multiplication.

D: Instructions to perform division

- **DIV** − Used to divide the unsigned word by byte or unsigned double word by word.
- **IDIV** − Used to divide the signed word by byte or signed double word by word.
- **AAD** − Used to adjust ASCII codes after division.
- **CBW** − Used to fill the upper byte of the word with the copies of sign bit of the lower byte.
- **CWD** − Used to fill the upper word of the double word with the sign bit of the lower word.

3. Bit Manipulation Instructions

These instructions are used to perform operations where data bits are involved, i.e. operations like logical, shift. Following is the list of instructions under this group −

A: Instructions to perform logical operation

- **NOT** − Used to invert each bit of a byte or word.
- **AND** − Used for adding each bit in a byte/word with the corresponding bit in another byte/word.
- **OR** − Used to multiply each bit in a byte/word with the corresponding bit in another byte/word.
- **XOR** − Used to perform Exclusive-OR operation over each bit in a byte/word with the corresponding bit in another byte/word.
- **TEST** − Used to add operands to update flags, without affecting operands.

B: Instructions to perform shift operations

- **SHL/SAL** − Used to shift bits of a byte/word towards left and put zero(S) in LSBs.
- **SHR** − Used to shift bits of a byte/word towards the right and put zero(S) in MSBs.
- **SAR** − Used to shift bits of a byte/word towards the right and copy the old MSB into the new MSB.

C: Instructions to perform rotate operations

- **ROL** − Used to rotate bits of byte/word towards the left, i.e. MSB to LSB and to Carry Flag [CF].
- **ROR** − Used to rotate bits of byte/word towards the right, i.e. LSB to MSB and to Carry Flag [CF].
- **RCR** − Used to rotate bits of byte/word towards the right, i.e. LSB to CF and CF to MSB.
- **RCL** − Used to rotate bits of byte/word towards the left, i.e. MSB to CF and CF to LSB.

4. String Instructions

String is a group of bytes/words and their memory is always allocated in a sequential order. Following is the list of instructions under this group –

- **REP** − Used to repeat the given instruction till CX ≠ 0.
- **REPE/REPZ** − Used to repeat the given instruction until CX = 0 or zero flag ZF = 1.
- **REPNE/REPNZ** − Used to repeat the given instruction until CX = 0 or zero flag ZF = 1.
- **MOVS/MOVSB/MOVSW** − Used to move the byte/word from one string to another.
- **COMS/COMPSB/COMPSW** − Used to compare two string bytes/words.
- **INS/INSB/INSW** − Used as an input string/byte/word from the I/O port to the provided memory location.
- **OUTS/OUTSB/OUTSW** − Used as an output string/byte/word from the provided memory location to the I/O port.
- **SCAS/SCASB/SCASW** − Used to scan a string and compare its byte with a byte in AL or string word with a word in AX.
- **LODS/LODSB/LODSW** − Used to store the string byte into AL or string word into AX.

5. Program Execution Transfer Instructions (Branch and Loop Instructions)

These instructions are used to transfer/branch the instructions during an execution. It includes the following instructions −

A: Instructions to transfer the instruction during an execution without any condition −

- **CALL** − Used to call a procedure and save their return address to the stack.
- **RET** − Used to return from the procedure to the main program.
- **JMP** − Used to jump to the provided address to proceed to the next instruction.

B: Instructions to transfer the instruction during an execution with some conditions −

- **JA/JNBE** − Used to jump if above/not below/equal instruction satisfies.
- **JAE/JNB** − Used to jump if above/not below instruction satisfies.
- **JBE/JNA** − Used to jump if below/equal/ not above instruction satisfies.
- **JC** − Used to jump if carry flag CF = 1
- **JE/JZ** − Used to jump if equal/zero flag ZF = 1
- **JG/JNLE** − Used to jump if greater/not less than/equal instruction satisfies.
- **JGE/JNL** − Used to jump if greater than/equal/not less than instruction satisfies.
- **JL/JNGE** − Used to jump if less than/not greater than/equal instruction satisfies.
- **JLE/JNG** − Used to jump if less than/equal/if not greater than instruction satisfies.
- **JNC** − Used to jump if no carry flag (CF = 0)
- **JNE/JNZ** − Used to jump if not equal/zero flag ZF = 0
- **JNO** − Used to jump if no overflow flag OF = 0
- **JNP/JPO** − Used to jump if not parity/parity odd PF = 0
- **JNS** − Used to jump if not sign SF = 0
- **JO** − Used to jump if overflow flag OF = 1
- **JP/JPE** − Used to jump if parity/parity even PF = 1
- **JS** − Used to jump if sign flag SF = 1

6. Processor Control Instructions

These instructions are used to control the processor action by setting/resetting the flag values. Following are the instructions under this group −

- **STC** − Used to set carry flag CF to 1
- **CLC** − Used to clear/reset carry flag CF to 0
- **CMC** − Used to put complement at the state of carry flag CF.
- **STD** − Used to set the direction flag DF to 1
- **CLD** − Used to clear/reset the direction flag DF to 0
- **STI** − Used to set the interrupt enable flag to 1, i.e., enable INTR input.
- **CLI** − Used to clear the interrupt enable flag to 0, i.e., disable INTR input.

7. Iteration Control Instructions

These instructions are used to execute the given instructions for number of times. Following is the list of instructions under this group −

- **LOOP** − Used to loop a group of instructions until the condition satisfies, i.e., CX = 0
- **LOOPE/LOOPZ** − Used to loop a group of instructions till it satisfies ZF = 1 & CX = 0
- **LOOPNE/LOOPNZ** − Used to loop a group of instructions till it satisfies ZF = 0 & CX = 0
- **JCXZ** − Used to jump to the provided address if CX = 0

8. Interrupt Instructions

These instructions are used to call the interrupt during program execution.

- **INT** − Used to interrupt the program during execution and calling service specified.
- **INTO** − Used to interrupt the program during execution if OF = 1
- **IRET** − Used to return from interrupt service to the main program.

Computer Microprocessor

Part 3:

Lecturer: Asst. Prof. Dr. Raheem Abdul Sahib Ogla

References

1. **"fundamentals of computer organization and architecture**", by Mostafa Abd-El-Barr Hesham El-Rewini, 1PstP edition/2005.
2. David A. Patterson and John L. Hennessy, **"Computer Organization and Design"**,1998.
3. M. M. Mano, **"computer system architecture"** third edition, prentice Hall, 1993.
4. Walter A. Triebel, **"The 80386, 80486, and Pentium® Processors Hardware,Software, and Interfaceing",** 1998.

Microprocessor: Memory system and Addressing Mode

Micro-processors – 1’st course

- Introduction to Microprocessor and Microcomputer system.
  - Microprocessor Architecture and Register Set.
  - System Buses
  - Memory types and physical addressing.
  - I/O devices
- Instruction Set and Format
- Addressing Modes
- Introduction to Assembly Programming Language.
  - Arithmetic and logical Instructions (Shift and Rotate).
  - Program Control (interrupt and subroutine call).

References:

1. Abel P., "IBM PC Assembly Language and Programming", 4th Edition, Prentice Hall, 1998..
2. Thorne M., "Computer Organization and Assembly Language Programming", 2nd Edition, Benjamin/Cummings, 1990.

1. Memory system

The memory of a computer system consists 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, is shown in Figure 3. This is referred to as memory address of the byte. Such memory is called byte addressable memory.

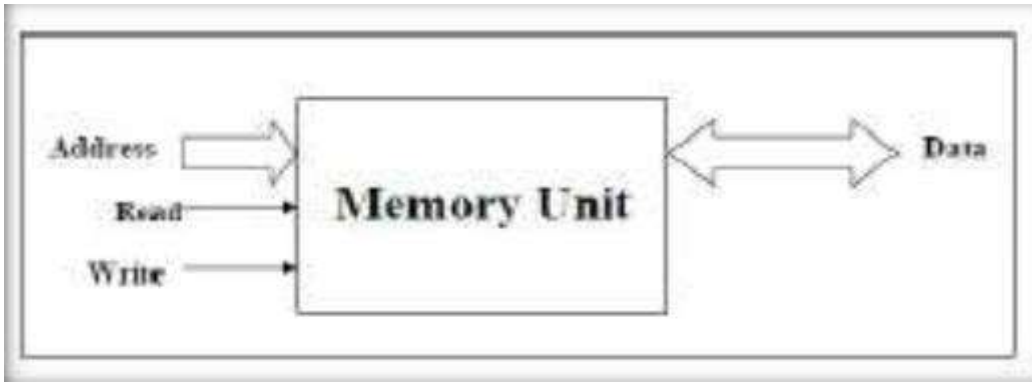


Figure3 logical view of the memory

8086 can address up to 1 MB (2^{20} bytes) of main memory this magic number comes from the fact that the address bud of the 8086 has 20 address lines. This number is referred to as the Memory Address Space (MAS). The memory address space of a system is determined by the address bus width of the CPU used in the system. The actual memory in a system is always less than or equal to the MAS.

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, is shown in Figure 4.

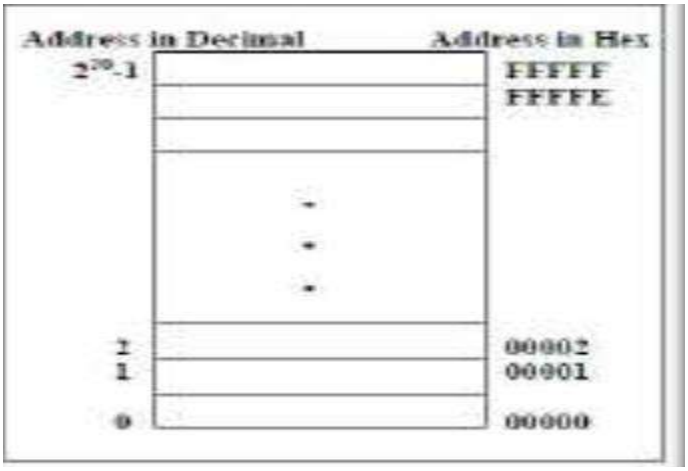


Figure 4: Block Diagram of Memory system

Steps in a typical read cycle:

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 write cycle:

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.

Q1: Define Offset address?

Offset is the displacement of the memory location from the starting location of the segment. The 8086 addresses a segmented memory. The complete physical address which is 20-bits long is generated using segment and offset registers each of the size 16-bit. In Other word, **offset address is the number of address locations added to a base address in order to get to a specific absolute address**. See figure (5)

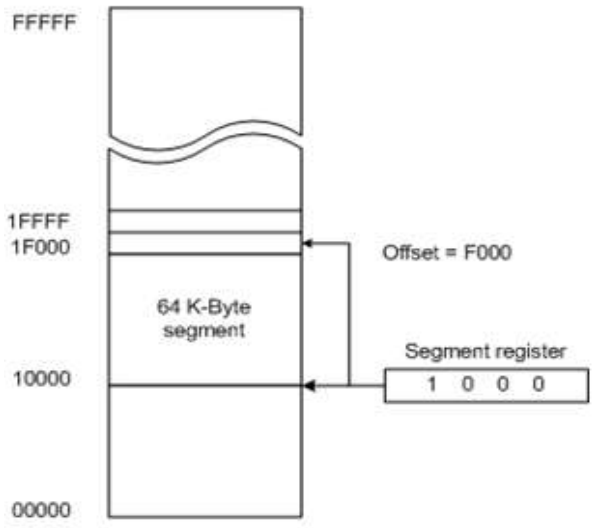


Figure (5) Shows the implementation o f offset Address

*** addresses type **

1. Logical address= CS: IP (Instruction Pointer (IP))
2. Offset address= CS0+IP
3. Physical address= CS0+offset address
4. Lower range= CS+0000 or 00000
5. Upper rang= CS+FFFF

Example:
If CS=24F6H and IP 634AH
The Logical address= CS: IP then 24F6:634A
Offset address= CS0+IP then 634A
Physical address= CS0+offset address then
2B2AA(24F60+634A)
Lower range= CS+0000 then 24F60(24F60+0000)
Upper rang= CS+FFFF then 34F5F(24F60+FFFF)

7- INPUT/OUTPUT

Input / Output (I/O) devices provide the means by which the computer system can interact with the outside world. Computers use I/O devices (also called peripheral devices) for two major purposes:

1. To communicate with the outside world and,
2. Store data.

Devices that are used to communicate like, printer, keyboard, modem, Devices that are used to store data like disk drive. I/O devices are connected to the system bus through I/O controller (interface) – which acts as interface between the system bus and I/O devices.

There are two main reasons for using I/O controllers

1. I/O devices exhibit different characteristics and if these devices are connected directly, the CPU would have to understand and respond appropriately to each I/O device. This would cause the CPU to spend a lot of time interacting with I/Devices and spend less time executing user programs.
2. The amount of electrical power used to send signals on the system bus is very low. This means that the cable connecting the I/O device has to be very short (a few centimeters at most). I/O controllers typically contain driver hardware to send current over long cable that connects I/O devices, is shown in Figure 6.

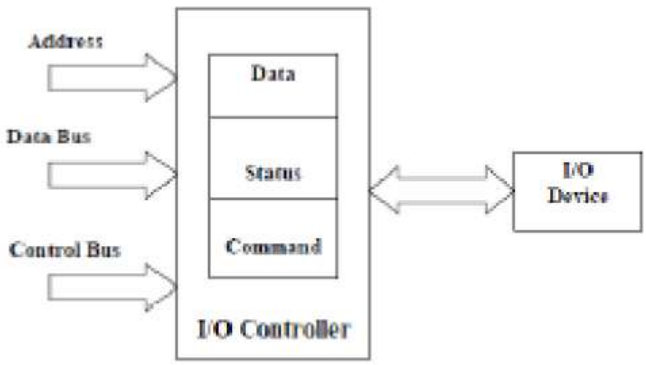
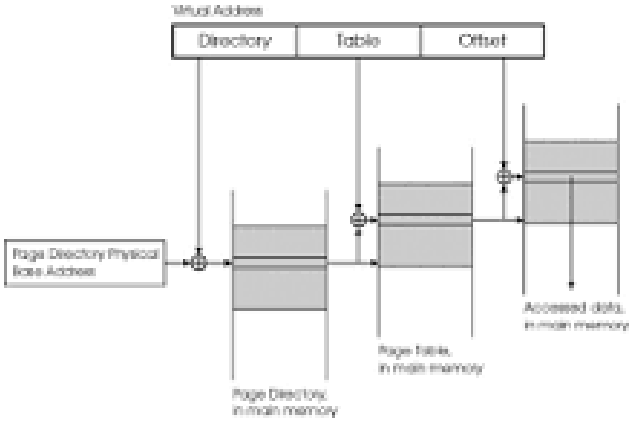


Fig6: Input / Output device interface

Q1;\ What is the difference between base address and offset address?

Solution

The base address register is a pointer to a byte in memory, and the offset specifies a number of bytes. Immediate means the address is calculated using the base address register and a 12-bit offset encoded in the instruction.



2.Addressing mode of 8086:

What Means by Addressing Modes?

The different ways in which a source operand is denoted in an instruction is known as **addressing modes**. There are 8 different addressing modes in 8086 programming –

1. Immediate addressing mode

The addressing mode in which the data operand is a part of the instruction itself is known as immediate addressing mode.

```
Example
MOV CX, 4929 H,
ADD AX, 2387 H,
 MOV AL, FFH
```

2. Register addressing mode

It means that the register is the source of an operand for an instruction.

```
Example
MOV CX, AX   ; copies the contents of the 16-bit AX register into  ; the 16-bit CX
register),
ADD BX, AX
```

3. Direct addressing mode

The addressing mode in which the effective address of the memory location is written directly in the instruction.

```
Example
MOV AX, [1592H], MOV AL, [0300H]
```

4. Register indirect addressing mode

This addressing mode allows data to be addressed at any memory location through an offset address held in any of the following registers: BP, BX, DI & SI.

```
Example
MOV AX, [BX]; Suppose the register BX contains 4895H, then the contents
          ; 4895H are moved to AX
ADD CX, {BX}
```

5. Based addressing mode

In this addressing mode, the offset address of the operand is given by the sum of contents of the BX/BP registers and 8-bit/16-bit displacement.

```
Example
MOV DX, [BX+04], ADD CL, [BX+08]
```

6. Indexed addressing mode

In this addressing mode, the operands offset address is found by adding the contents of SI or DI register and 8-bit/16-bit displacements.

Example
MOV BX, [SI+16], ADD AL, [DI+16]

7. Based-index addressing mode

In this addressing mode, the offset address of the operand is computed by summing the base register to the contents of an Index register.

Example
ADD CX, [AX+SI], MOV AX, [AX+DI]

8. Based indexed with displacement mode

In this addressing mode, the operands offset is computed by adding the base register contents. An Index registers contents and 8 or 16-bit displacement.

Example
MOV AX, [BX+DI+08], ADD CX, [BX+SI+16]

Q2: \What is the difference between logical address and physical address 8086?
Solution
The basic difference between Logical and physical address is that **Logical address is generated by CPU in perspective of a program whereas the physical address is a location that exists in the memory unit**.

Q3: \ What is the main difference between base and index register addressing modes?
In the 8086 through the 80286, this type of addressing uses one base register (BP or BX), and one index register (DI or SI) to indirectly address memory. **The base register often holds the beginning location of a memory array, while the index register holds the relative position of an element in the array**.

Q4: \ What are the 3 basic addressing modes?
Lesson Summary

Here are the addressing modes discussed: **Immediate: The operand is included in the instruction. Direct: The effective address of the operand in memory is part of the instruction. Indirect: The instruction contains a memory address, which contains the effective address of the operand in memory.**

5.What are the addressing modes?
The most common types of addressing modes are **immediate, indirect, direct, indexed, and register** addressing modes.

طرق العنونة في المعالج الدقيق 8086 Microprocessor

ما المقصود بطرق العنونة Addressing Mode؟

تعد طرق وأوضاع العنونة طرقًا مختلفة يمكن لوحدة المعالجة المركزية من خلالها الوصول إلى البيانات أو المعاملات، وتحدد كيفية الوصول إلى عنوان ذاكرة محدد، لتحميل أي بيانات من وإلى الذاكرة /السجلات(memory/registers) ، يتم استخدام تعليمات (MOV)، صيغة تعليمات (MOV)هي:

MOV Destination, Source

يقوم بنسخ بيانات المعامل الثاني "المصدر (source) "في المعامل الأول "الوجهة" (destination)، للوصول إلى الذاكرة، يتم استخدام سجلات المقاطع (segment registers)جنبًا إلى جنب مع سجلات الأغراض العامة .(general-purpose registers)

ما هي طرق العنونة في المعالج الدقيق 8086 Microprocessor؟

هناك ثمانية أوضاع عنونة في المعالج الدقيق (8086)، الآن، سنناقشهم جميعًا بالتفصيل مع أمثلة على ايعازات التجميع :(assembly instructions)

1. وضع عنونة السجل:Register addressing mode

يتضمن هذا الوضع استخدام السجلات، هذه السجلات تحمل العمليات (operands)، هذا الوضع سريع جدًا مقارنة بالأوضاع الأخرى، لأنّ وحدة المعالجة المركزية (CPU) لا تحتاج إلى الوصول إلى الذاكرة، يمكن لوحدة المعالجة المركزية إجراء عملية مباشرة من خلال السجلات، كمثال على ذلك:

```
MOV AX, BL
MOV AL, BL
```

الايعازات المذكورة أعلاه تنسخ بيانات سجل (BL)إلى (AX)و.(AL)

مثال :

(ORG 100h) هو توجيه يخبر المترجم بكيفية التعامل مع كود المصدر .(source code)

```
ORG  100h
MOV AX, CX
ret
```

في هذا المثال،محتوى كل من سجلات (AX)و (CX)متماثل، لأنّ المثال أعلاه ينقل محتوى (CX) إلى سجل .(AX)

2- وضع العنونة الفوري :Immediate Addressing Mode

في هذا الوضع، هناك نوعان من العمليات، أحدهما سجل (register)والآخر قيمة ثابتة (constant value)، يأتي السجل بسرعة بعد رمز العملية، كمثال على ذلك:

- **الايعاز (MOV AX، (30Hتنسخ القيمة السداسية العشرية (30H)للسجل .(AX)**

الايعاز (MOV BX 255)تنسخ القيمة العشرية (255)للسجل .(BX)

لا يمكنك استخدام وضع العنونة الفوري لتحميل القيمة الفورية في سجلات المقطع، لنقل أي قيمة إلى سجلات المقطع، قم أولاً بتحميل هذه القيمة في سجل للأغراض العامة ثمّ أضف هذه القيمة إلى سجل المقطع.

مثال :

لنفترض أنّك تريد تخزين (320H)في سجل .(DS)

```
ORG  100h
MOV AX,320H; copies hexadecimal value 320h to AX
MOV DS, AX; copies value of AX into DS
ret; stops the program
```

3- وضع العنونة المباشر :Direct Addressing Mode

يقوم بتحميل أو تخزين البيانات من الذاكرة للسجل والعكس صحيح، تتكون الايعازات من سجل وعنوان تعويض (offset address)، لحساب العنوان الفعلي، انتقل إلى اليسار من سجل (DS) وأضف عنوان الإزاحة إليه.

MOV CX, [481]

القيمة الست عشرية (hexadecimal)لـ (481)هي (1E1)، افترض أنّ (DS = 2162H)، فإنّ العنوان المنطقي سيكون (2162: 01E1)، لحساب العنوان الفعلي، انتقل إلى اليسار من سجل (DS) وأضفه إلى عنوان الإزاحة، سيكون العنوان الفعلي (26120H + 1E1H = 26301H)، ومن ثمّ، بعد تنفيذ تعليمات (MOV)، سيتم تحميل محتويات موقع الذاكرة (26301H)في السجل (CX)، ستعمل الايعازات (MOV [2481]، (CXعلى تخزين محتوى السجل (CX)في موقع الذاكرة (26301H).

مثال :

```
ORG 100h
MOV AX, 2162H; copies hexadecimal value 2162h to AX
MOV DS, AX; copies value of AX into DS
MOV CX, 24; copies decimal value 24 into CX
MOV [481], CX; stores the data of CX to memory address 2162:01E1
MOV BX, [481]; load data from memory address 2162:01E1 into BX
RET; stops the program
```

في هذا المثال (18) (hexadecimal)، ويتم تخزين القيمة العشرية (24)في عنوان الذاكرة (21801h).

4- وضع العنونة غير المباشر للسجل: Register Indirect Addressing Mode

يستخدم وضع العنونة غير المباشر للسجل عنوان الإزاحة الموجود في أحد هذه السجلات الثلاثة، أي BX)و SIو(DI، ينتج عن مجموع عنوان الإزاحة وقيمة (DS)التي تمّ إزاحتها بواسطة موضع واحد عنوانًا فعليًا، فمثلًا:

MOV AL, [SI]

ستحسب هذه الايعازات العنوان الفعلي عن طريق تحويل (DS)إلى اليسار بمقدار موضع واحد وإضافته إلى عنوان الإزاحة الموجود في (SI)، تشير الأقواس حول (SI)إلا أنّ (SI)تحتوي على عنوان الإزاحة لموقع الذاكرة الذي يجب الوصول إلى بياناته، إذا كانت الأقواس غائبة، فستقوم الايعازات بنسخ محتويات سجل (SI)إلى (AL)، لذلك، تعتبر الأقواس ضرورية.

مثال :

```
ORG  100h
MOV AX, 0708h; set AX to hexadecimal value of 0708h
MOV DS, AX; copy value of AX to DS
MOV CX, 0154h; set CX to hexadecimal value of 0154h
MOV SI, 42Ah; set SI to 42Ah
MOV [SI], CX; copy contents of CX to memory at 0708:042Ah
RET ;returns to operating system
```

في المثال أعلاه، العنوان الفعلي (physical address)هو (07080 + 042A = 74AA)، وبالتالي، فإنّ (074AA)سيخزن أقل بايت من البيانات، أي (54)و (074 AB)سيخزن أهم أجزاء البيانات، أي .(01)

5- وضع العنونة على أساس النسبية: Based Relative Addressing Mode

يستخدم وضع العنونة هذا سجلًا أساسيًا (base register)، إمّا (BX)أو (BP)وقيمة إزاحة لحساب العنوان الفعلي.

Physical Address= Segment Register (Shifted to left by 1) + Effective address

العنوان الحقيقي (effective address)هو مجموع سجل الإزاحة (offset register)وقيمة الإزاحة، المقاطع الافتراضية لـ (BX)و (BP)هي (DS)و(SS)، فمثلًا:

MOV [BX+5], DX

في هذا المثال ، العنوان الحقيقي هو (BX + 5)والعنوان الفعلي هو + (DS (shifted left) (BX+5، ستقوم الايعازات الخاصة بالتنفيذ بنسخ قيمة (DX)إلى موقع الذاكرة الخاص بالعنوان الفعلي .(physical address= DS (shifted left) +BX+5)

مثال :

```
ORG  100h
MOV AX, 0708h; set AX to hexadecimal value of 0708h
MOV DS, AX; copy value of AX to DS
MOV CX, 0154h; set CX to hexadecimal value of 0154h
MOV BX, 42Ah; set BX to 42Ah
MOV [BX+5],DX ;copy contents of DX to memory at 0708:042Fh
RET ;returns to operating system
```

بعض الكودات البديلة الأخرى لـ (MOV DX, [BX + 5])هي:

```
MOV  DX, 5[BX]
MOV  DX, [BX]+5
```

6- وضع العنونة النسبية المفهرسة: Indexed Relative Addressing Mode

وضع العنونة هذا هو نفسه وضع العنونة النسبي المستند (based relative addressing mode)، الاختلاف الوحيد هو أنّه يستخدم سجلات (DI)و (SI)بدلاً من سجلات (BX)و(BP)، مثال على ذلك كالتالي:

إذا كان (DS = 704، SI = 2B2، DI = 145).

MOV [DI]+12, AL

ستعمل هذه الايعازات الخاصة بالتنفيذ على نسخ محتوى (AL)على عنوان الذاكرة (7197 "7040" + 145 + 12").

MOV BX, [SI]+10

ستقوم هذه الايعازات بتحميل المحتويات من عنوان الذاكرة ("7040 + 2B2 +10" 7302)للسجل (BX).

7- وضع العنونة المفهرس المعتمد: Based Indexed Addressing Mode

إن وضع العنونة المفهرس المعتمد هو في الواقع مزيج من وضع العنونة النسبي المستند إلى وضع العنونة النسبي المفهرس، يستخدم سجلًا أساسيًا (base register)واحدًا (BX، BP)وسجل فهرس (index register)واحد (SI، DI)، فمثلًا:

```
MOV AX, [BX+SI+20]
```

يمكن أيضًا كتابة الايعازات المذكورة أعلاه على النحو التالي:

```
MOV AX, [SI+BX+20]
OR
MOV AX, [SI][BX]+20
```

في هذه الحالة، سيكون العنوان الفعلي هو (DS (Shifted left) + SI + BX + 20)، الآن، إذا استبدلنا (BX)بـ (BP)، فسيكون العنوان الفعلي يساوي (SS (Shifted left) + SI + BX + 20)، الآن، دعونا نلقي نظرة على هاتين الايعازين:

MOV AX, [BX][BP]+20
MOV AX, [SI][DI]+20

كلا التعبيرين غير مسموح بهما، لأنّ هذا الوضع يدعم سجل أساسي واحد وسجل مقطع واحد.

8- نمط العنونة Based Indexed with Displacement mode

في هذا النوع من نمط العنونة ، يتم حساب ال offset بجمع امحتويات ال base محتويات مسجلات ال index مع 8 او 16 بت-displacement register.

مثال :

MOV AX, [BX+DI+08]
ADD CX, [BX+SI+16]

Q6: \ What are different addressing modes in 8086? Give Examples for each one

- Immediate addressing mode.
- Register addressing mode.
- Direct addressing mode.
- Register indirect addressing mode.
- Based addressing mode.
- Indexed addressing mode.
- Based-index addressing mode.
- Based indexed with displacement mode.

Q7: \ What are displacement base and index in 8086?

In 8086 we have base registers and index registers. So EU calculates EA by summing a displacement, the content of base register and the content of index register.

EA = {Base Register} + {Index Register} + {8 0r 16-bit displacement}

The displacement is an 8 or 16 bit number that is contained in the instruction.

Q8: \ How physical address is calculated in 8086 explain with example?

The 8086 addresses a segmented memory. The complete physical address which is 20-bits long is generated **using segment and offset registers each of the size 16-bit**. The content of a segment register also called as segment address, and content of an offset register also called as offset address.