

**قــســــــــــم الامــــــــــــــــــــــــــن الــــــــــــــــــــــــــــــــــــسيبرانــــــــــــــــــــــــــــــــــــي**

**Department of Cyber Security**

**MICROPROCESSORS**

**Lecture: (6)**

**microprocessor**

**Class: Second**

**Lecturer: M.Sc.Muntather AL-mussawee**

# Microprocessor Architecture

The microprocessor has become more essential part of many gadgets. The evolution of microprocessors was divided into five generations such as first, second, third, fourth and fifth generation.



The 8086 microprocessor has a segmented memory architecture, which means that memory is divided into segments that are addressed using both a segment register and an offset. The segment register points to the start of a segment, while the offset specifies the location of a specific byte within the segment. This allows the 8086 microprocessor to access large amounts of memory, while still using a 16-bit data bus

**the data registers of the 8086**

1. **AX (Accumulator Register)**
	* Function: Used for arithmetic operations and serves as an input/output for operations
2. BX (Base Register)
	* Function: Used as a base address when accessing memory.
3. **CX (Count Register)**
	* Function: Used as a counter in loops or iterative operations.

1. **DX (Data Register)**
	* Function: Used to store data and works with AX in operations requiring higher precision
2. **SI (Source Index)**

Function: Used in operations that require accessing data in memory

1. DI (Destination Index)
	* Function: Used as the destination for data being transferred from the source.
2. **BP (Base Pointer)**
	* Function: Used to point to the location of data in the stack
3. **SP (Stack Pointer)**
	* Function: Used to track the top address of the stack

**some basic instructions (operations) used in the 8086 microprocessor**

1. **Arithmetic Instructions**
	* **ADD**: Adds two values.
	* **SUB**: Subtracts one value from another.
	* **MUL**: Multiplies two values (16-bit result, using AX and DX).
	* **DIV**: Divides one value by another (uses DX and AX).
2. **Logical Instructions**
	* **AND**: Performs a logical AND operation between two values.
	* **OR**: Performs a logical OR operation between two values.
	* **XOR**: Performs a logical XOR operation between two values. NOT:

Inverts a given value.

1. **Data Transfer Instructions**
	* **MOV**: Transfers data from one register to another or from memory to a register.
	* **PUSH**: Pushes a value onto the stack.
	* **POP**: Pops a value off the stack.
	* **XCHG:** Exchanges values between registers
2. **Control Instructions**
	* **JMP**: Jumps to a specified address (unconditional).
	* **CALL**: Calls a procedure (function).
	* **RET**: Returns from a procedure.
	* **CMP**: Compares two values

**Types of Instructions:**

To simplify learning the Intel 8086 MP instruction set, instructions are divided into seven categories these groups are:

1. **Data Transfer**: the basic data transfer instruction is MOV this instruction can be used in several ways to copy a byte, a word (16 bit) or a double word between various sources and destinations such as registers, memory.

NOTES:

* the MOV instruction cannot transfer data directly between two memories. To solve this problem, the data to be transferred must first be transferred to an internal register using a MOV instruction, and then the contents of this register should be transferred to a location in memory using another MOV instruction.·

**MOV [], [] MOV AX,[]**

**MOV [],AX**

* cannot move immediate value into a segment register directly. That is, the following instruction is not allowed MOV DS,1000. To solve this problem, we use the following two instructions:

**MOV AX,1000**

**MOV DS,AX**

It is not possible to transfer the contents of one segment register to another segment register directly, that is, the following instruction is not allowed MOV DS,ES and to solve this problem.

**MOV AX,ES**

**MOV DS,AX**

**Example: MOV AL, [SI]** This instruction means transferring the contents of the memory location pointed to by the SI register to the AL register. The addressing system in this instruction is indirect addressing to the register and the source variable has a physical address of PA = DS x 10h + SI. As for the variable The destination is AL.

**2. Arithmetic:** ADD, SUB/ MUL and DIV instructions. Increment or decrement INC, DEC instruction is also included. These

instructions allow for carry operation and for signed & unsigned arithmetic. The operand located in register or memory location.

**8086 Addressing Modes**

When the 8086 MP execute an instruction, it performs the specified function on data. These data are called operand and may be part of the instruction, reside in one of the internal registers of 8086 or stored at an address in memory. To access these different types of operand, the 8086 provide with various addressing mode. An addressing mode can be defined as the way, in which the operand is specified in an instruction,

i.e. (Addressing mode is a method of specifying an operand), and can be categorized into Three main types:

1. **Register Addressing Mode:**

The register addressing mode involves the use of registers to hold the data to be manipulated. Memory is not accessed when this addressing mode is executed; therefore, it is relatively fast.

Examples:

MOV BX, DX → Copy the contents of DX into BX.

MOV ES, AX → Copy the contents of AX into ES. ADD AI, BL →

AL=AL+BL

1. **Immediate Addressing Mode:**

In this mode, the Source operand is a constant and part of the instruction. Immediate addressing mode can be used to load information into any of the registers except the segment registers and flag register. Also this type of addressing executed quickly because when the instruction is assembled the operand comes immediately after code.

Examples:

MOV BX, 2550h → Move 2550h into BX, where BL=50 and BH=25. MOV CX, 625 → Load the decimal value 625 into CX. CX=0271h

ADD BL, 40h → BL= BL+ 40h

1. **Memory Addressing mode:**

To reference an operand in memory. The 8086 MP must calculate the physical address of the operand and then read or write this storage location.

The physical address (PA) = Segment Address (SA) + Offset Address (EA)

Segment address (SA), which is usually stored at one of the segment register

Offset address or Effective address (EA): is the address of the offset that can be made up from as many as three elements: Base, Index, and Displacement.

Not all effective address elements are always used in the effective address calculation. In fact a number of memory

addressing modes are defined by using various combinations are:

 1direct addressing

2-Register indirect addressing

3-Based addressing

4-Indexed addressing

5-Based indexed addressing

6-Based indexed with displacement addressing

**Example 1**

Write a program to:

1. Transfer 1104 into reg CX and 4224 into reg AX.
2. Put the value of reg AX into memory location 1010 and the value of

CX into memory location 106



**EXAMPL2:**

Write a program to:

1. Transfer 1111 into reg BX and 3224 into reg DX.
2. Put the value of reg BX into memory location 1000 and the value of DX into memory location 1040.