



## Computer Science Dept.



Department	Computer Science	القسم:
Subject Name:	Microprocessors - (1)	أسم المادة :
Year of Study:	2023-2024	السنة الدراسية:
Term:	Second Term	الفصل الدراسي:
Email	<a href="mailto:ali.sadoon@uoa.edu.iq">ali.sadoon@uoa.edu.iq</a>	Email
Instructor Name:	Ali Saadoon Ahmed	أسم التدريسي:



# OUTLINE



- What are logic gates
- Let's do an example
- Types of Logic Gates!
- Number System
- Decimal Number System
- Binary Number System
- Why Binary?
- Octal Number System
- Hexadecimal Number System
- Relationship between Hexadecimal, Octal, Decimal, and Binary
- Number Conversions

# WHAT ARE LOGIC GATES?



Logic gates are the switches that turn ON or OFF depending on what the user is doing!



They are the building blocks for how computers work.

# WHAT ARE LOGIC GATES?

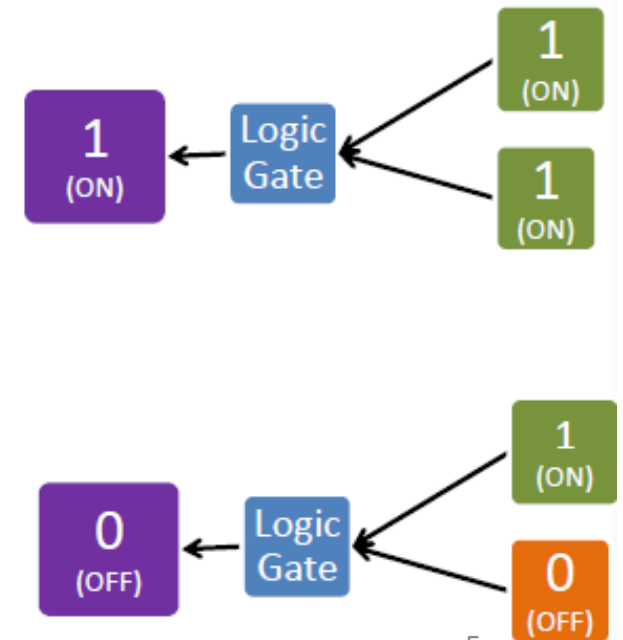
- Logic gates turn **ON** when a certain condition is true, and **OFF** when the condition is false
- They check whether or not the information they get follows a certain rule
- They either spit out the answer true (**ON**) or false (**OFF**)
- Remember:
  - True= ON = 1
  - False = OFF=0



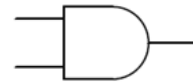


# LET'S DO AN EXAMPLE

- Let's say a certain logic gate needs to determine if two numbers are equal
- We learned before that computers only think of things in terms of **ON** and **OFF**, which to them is 1 and 0
- *Reminder*: Input refers to the information you give the logic gate, and output refers to what it spits out!
- Let's try this example, keeping this rule in mind!



# TYPES OF LOGIC GATES!



**AND**

A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1



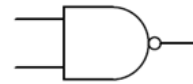
**OR**

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1



**XOR**

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0



**NAND**

A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0



**NOR**

A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0

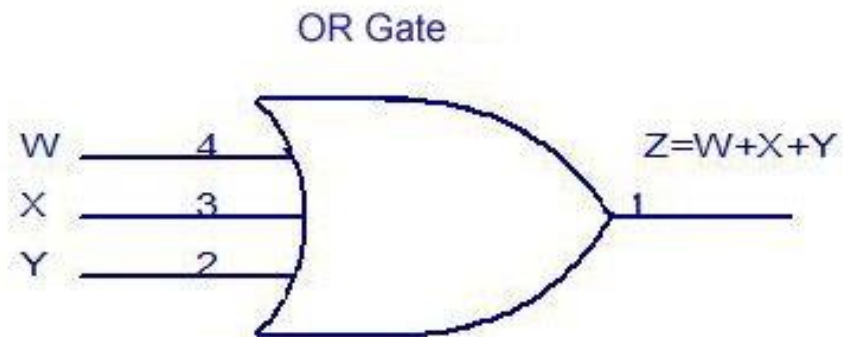


**XNOR**

A	B	Output
0	0	1
0	1	0
1	0	0
1	1	1

# TYPES OF LOGIC GATES!

## 3 Input OR Gate

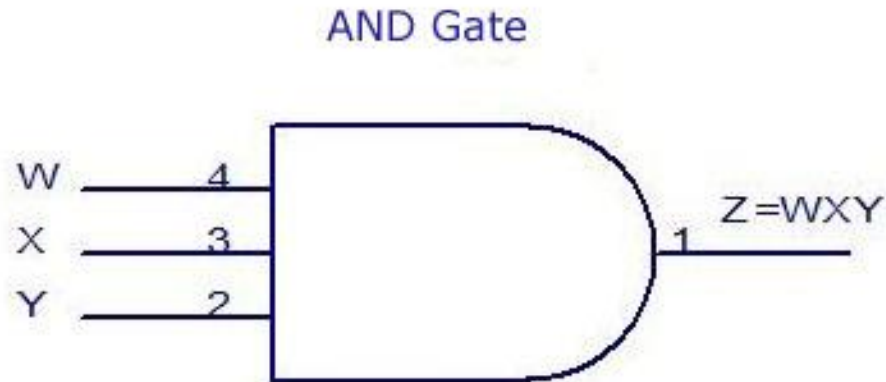


TRUTH TABLE

INPUTS			OUTPUT
W	X	Y	Z
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

# TYPES OF LOGIC GATES!

## 3 Input AND Gate



TRUTH TABLE

INPUTS			OUTPUT
W	X	Y	Z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1





# DECIMAL NUMBER SYSTEM

- It consists of ten digits i.e., 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 with the base 10.
- Each number can be used individually, or they can be grouped to form a numeric value as 85,48,35,456 etc.
- The Binary Number System consists of only two digits— 0 and 1.
- Since this system uses two digits, it has the base 2.
- All digital computers use this number system and convert the data input from the decimal format into its binary equivalent.



# OCTAL NUMBER SYSTEMS

- In the Octal Number System, it consist of 8 digits i.e., 0, 1, 2, 3, 4, 5, 6, 7 with a base 8.
- The sequence of octal number goes as 0, 1, 2,3, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, 16, 17, 20, 21, 22, .....as go on.
- See each successive number after 7 is a combination of two or more unique symbols of octal system.



# HEXADECIMAL NUMBER SYSTEM



- The Hexadecimal system use base 16.
- It has 16 possible digit symbol.
- It use the digit 0 through 9 plus the letters A, B, C, D, E, and F as the 16 digit symbols.



# RELATIONSHIP BETWEEN HEXADECIMAL, OCTAL, DECIMAL, AND BINARY



Hexadecimal	Octal	Decimal	Binary
0	0	0	0000
1	1	1	0001
2	2	2	0010
3	3	3	0011
4	4	4	0100
5	5	5	0101
6	6	6	0110
7	7	7	0111
8	10	8	1000
9	11	9	1001
A	12	10	1010
B	13	11	1011
C	14	12	1100
D	15	13	1101
E	16	14	1110
F	17	15	1111



# DECIMAL-TO-BINARY CONVERSIONS



- The method of converting Decimal to binary is repeated-division method.

For conversion follow the rules:

1. Divide the given decimal number with the base 2.
2. Write down the remainder and divide the quotient by 2.
3. Repeat step 2 till the quotient is zero.

# DECIMAL-TO-BINARY CONVERSIONS

2	200	Remainders	
2	100	0	LSB
2	50	0	
2	25	0	
		Write	
2	12	1	in
2	6	0	this
2	3	0	
		order	
2	1	1	
	0	1	MSB

Reading the remainders from the bottom to top, the result is

$$200_{10} = 11001000_2$$



# BINARY-TO-DECIMAL CONVERSION



- To convert a binary number, follow the steps:
  1. Multiply each binary number with 2 having the power 0 for last position, starting from the right digit.
  2. Increase the power one by one, with base as
  3. Sum up all the products to get decimal number.



# BINARY-TO-DECIMAL CONVERSION

$$\begin{aligned} 110001001_2 &= 1 \times 2^8 + 1 \times 2^7 + 0 \times 2^6 + \\ &\quad 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + \\ &\quad 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\ &= 256 + 128 + 0 + 0 + 0 + 8 + 0 + 0 + 1 \\ &= 393 \end{aligned}$$

Thus,  $110001001_2 = 393_{10}$





# DECIMAL-TO-OCTAL



- The method of converting Decimal to Octal is repeated-division method. For conversion follow the rules:
  1. Divide the given decimal number with the base 8,
  2. Write down the remainder and divide the quotient by 8,
  3. Repeat step 2 till the quotient is zero.



# DECIMAL-TO-OCTAL

$$501 \div 8 = 62.625$$

$$0.625 * 8 = 5$$

$$62 \div 8 = 7.75$$

$$0.75 * 8 = 6$$

$$7 \div 8 = 0.875$$

$$0.875 * 8 = 7$$

Therefore,  $(765)_8$  equals  $(501)_{10}$



# OCTAL-TO-DECIMAL CONVERSION

- To convert a octal number follow the steps:
  1. Multiply each Octal number with 8 having the power 0 for last position, starting from the right digit.
  2. Increase the power one by one, with base as 8.
  3. Sum up all the products to get decimal number.



# OCTAL-TO-DECIMAL CONVERSION

$$\begin{aligned} 372_8 &= 3 \times 8^2 + 7 \times 8^1 + 2 \times 8^0 \\ &= 3 \times 64 + 7 \times 8 + 2 \times 1 \\ &= 192 + 56 + 2 \\ &= 250_{10} \end{aligned}$$

Thus,  $372_8 = 250_{10}$

So, an octal number can be easily converted to its decimal equivalent by multiplying each octal digit by its position weight.

# OCTAL-TO-BINARY CONVERSION

- The conversion from octal to binary is performed by converting each octal digit to its 3-bit binary equivalent.

We convert  $5431_8$  to binary using 3 bits for each octal digit as follows:

5	4	3	1
↓	↓	↓	↓
101	100	011	001

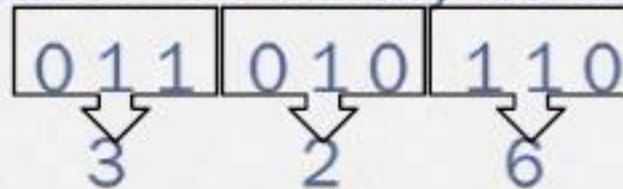
Thus,  $5431_8 = 101100011001_2$

# BINARY-TO-OCTAL CONVERSION

Converting from binary integers to octal integers is simply the reverse of the foregoing process. Firstly you have to do is:

1. Group the binary integer into 3-bits starting at the Least Significant Bit(LSB).
2. If unable to form group then, add one or two 0s.
3. Each group is converted to its octal equivalent.

It illustrated below for binary number 11010110



Thus,  $11010110_2 = 326_8$



# DECIMAL-TO-HEXADECIMAL CONVERSION



- The method of converting Decimal to Hexadecimal is repeated-division method. For conversion follow the rules:
  1. Divide the given decimal number with the base 16.
  2. Write down the remainder and divide the quotient by 16.
  3. Repeat step 2 till the quotient is zero.



# DECIMAL-TO-HEXADECIMAL CONVERSION

Convert the number **921** DECIMAL to HEXADECIMAL

DIVISION	RESULT	REMAINDER (in HEX)
921 / 16	57	9
57 / 16	3	9
3 / 16	0	3
ANSWER		399

Convert the number **188** DECIMAL to HEXADECIMAL

DIVISION	RESULT	REMAINDER (in HEX)
188 / 16	11	C (12 decimal)
11 / 16	0	B (11 decimal)
ANSWER		BC





# HEXADECIMAL-TO-DECIMAL CONVERSION



- To convert a Hexadecimal number, follow the steps:
  1. Multiply each hexadecimal number with 16 having the power 0 for last position, starting from the right digit.
  2. Increase the power one by one, with base as 16.
  3. Sum up all the products to get decimal number.



# HEXADECIMAL-TO-DECIMAL CONVERSION



$$\begin{aligned}2AF_{16} &= 2 \times 16^2 + 10 \times 16^1 + 15 \times 16^0 \\ &= 512 + 160 + 15 \\ &= 687_{10}\end{aligned}$$

$$\text{Thus, } 2AF_{16} = 687_{10}$$

# BINARY-TO-HEXADECIMAL CONVERSION

Converting from binary integers to hexadecimal integers is simple. Firstly you have to do is:

1. Group the binary integer into 4-bits starting at the Least Significant Bit(LSB).
2. If unable to form group then, add one or two 0s.
3. Each group is converted to its Hexadecimal equivalent.

It illustrated below for binary number 1010111010

0010

2

1011

B

1010

A

Thus,  $1010111010_2 = 2BA_{16}$

# HEXADECIMAL-TO-BINARY CONVERSION

The conversion from Hexadecimal to binary is performed by converting each Hexadecimal digit to its 4-bit binary equivalent.

This is illustrated below:

$$9F2_{16} = \begin{array}{ccc} 9 & F & 2 \\ \downarrow & \downarrow & \downarrow \\ 1001 & 1111 & 0010 \end{array}$$

$$\text{Thus, } 9F2_{16} = 100111110010_2$$



**Computer Science Dept.**

**THANK  
YOU**



**By:  
Ali Saadoon Ahmed**



## Computer Science Dept.



Department	Computer Science	القسم:
Subject Name:	Microprocessors - (2)	أسم المادة :
Year of Study:	2023-2024	السنة الدراسية:
Term:	Second Term	الفصل الدراسي:
Email	<a href="mailto:ali.sadoon@uoa.edu.iq">ali.sadoon@uoa.edu.iq</a>	Email
Instructor Name:	Ali Saadoon Ahmed	أسم التدريسي:

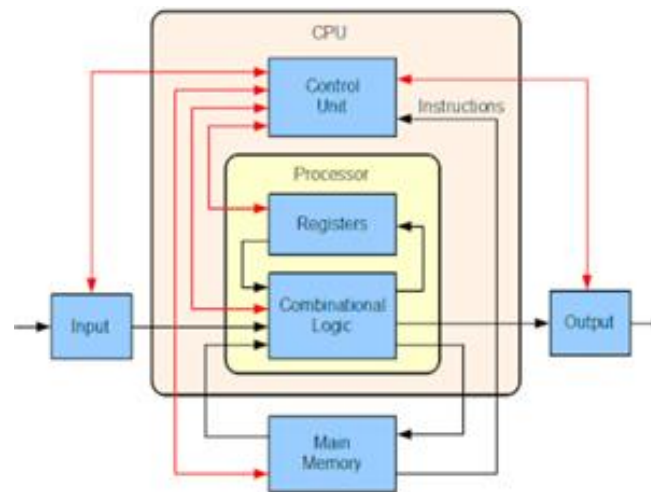


# OUTLINE

- What is a Computer Architecture?
- Important Words:
- Computer element
- Measures of capacity and speed
- Computer level steps

# WHAT IS A COMPUTER ARCHITECTURE?

- ✓ **Computer Architecture:** Computer architecture is a science or a set of rules stating how computer software and hardware are joined together and interact to make a computer work. It not only determines how the computer works but also of which technologies the computer is capable.



- ✓ **Computer Architecture and Organization :** The way of hardware are connected together to form a computer.





# IMPORTANT WORDS:



معناها	الكلمة	معناها	الكلمة
بيانات	Data	تعليمات / أوامر	Instructions
الذاكرة الثانوية	Secondary Memory	الذاكرة الرئيسية	Main Memory (M.M)
وحدة التحكم	Control Unit (CU)	وحدة المعالجة المركزية	Central Processing Unit (CPU)
ملحقات	Peripherals	وحدة الحساب والمنطق	Arithmetic Logic Unit (ALU)
دورة/حلقة	Cycle	أجهزة الإدخال والإخراج	Input/Output device (I/O)
نظام	System	عنونة	addressable
موقع	location	عنوان	address

# THE COMPUTER CONSISTS OF THREE MAIN SUB-SYSTEM:

## 1. *Central Processing Unit (CPU):*

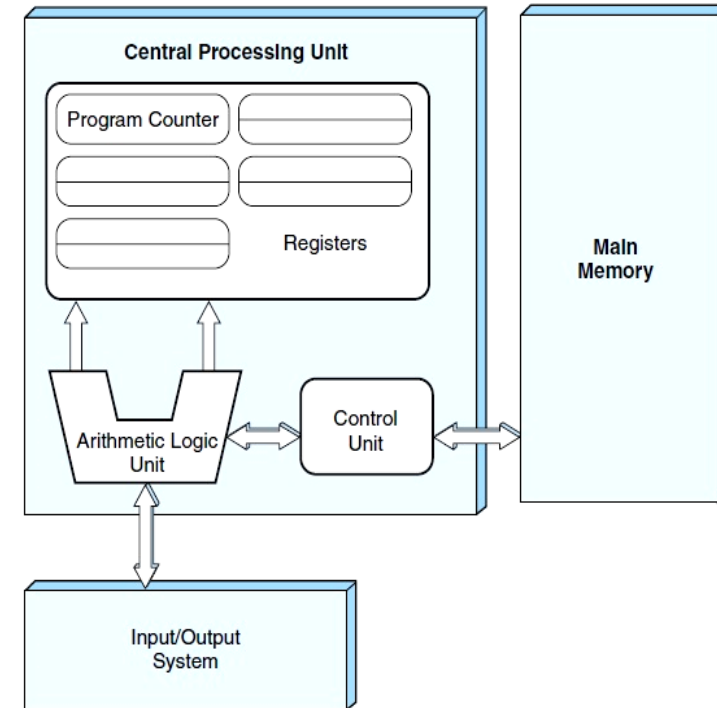
Also simply called as the microprocessor acts as the brain coordinating all activities within a computer.

## 2. *The Memory:*

The program instructions and data are primarily stored.

## 3. *The Input/output (I/O) Devices:*

Allow the computer to input information for processing and then output the results. I/O Devices are also known as computer peripherals.



# COMPUTER ELEMENT

## 1- Memory

- Stores both program and data (Permanent, temporary)

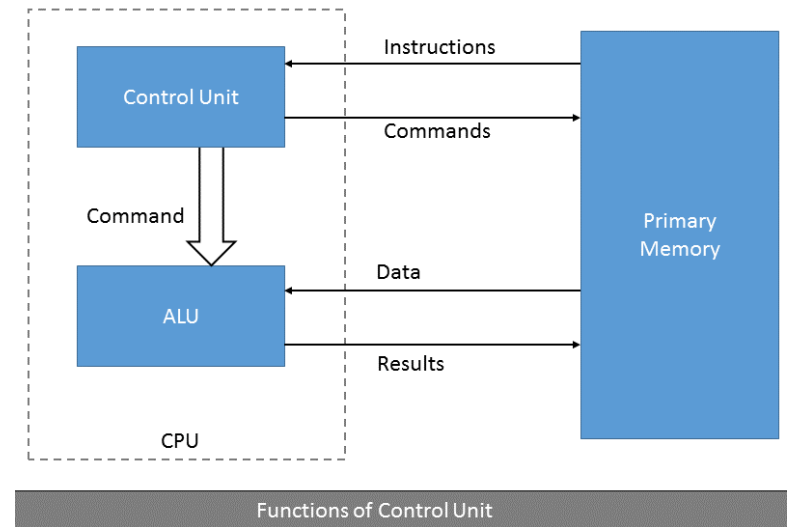


## 2- Control Unit

- Directs the operations of the other units by providing timing and control signals.

## 3- ALU

- Performs arithmetic and logical operations such as addition, subtraction, multiplication and division.



# COMPUTER ELEMENT

## • 4- Input

- An input device gets data from users
- Examples are keyboards, mice, webcams, microphones, and secondary storage devices (hard disks, floppy disks, CD-ROMs etc) .



## 5- Output

- An output device sends data to users.
- Typical output devices are monitors, printers, modems, and secondary storage devices.



# MEMORY SUBSYSTEM

Memory also called RAM(Random Access Memory) consists of many memory cells(Storage Unites) of a fixed size.

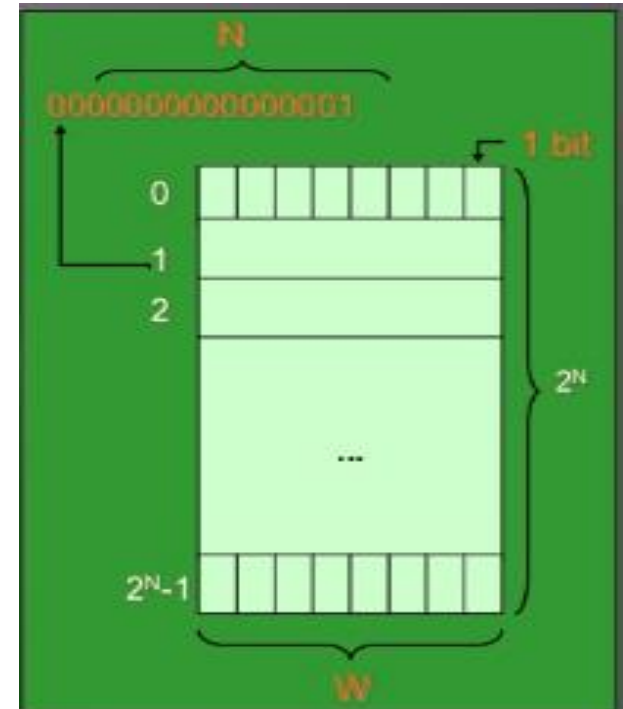
❖ There are two important things in RAM:

1- Memory Width(W)

- How many bits is each memory cell, Typically one byte(=8bits).

2- Address width(N):

- How many bits used to represent each address, determines the maximum memory size=address space.





# MEASURES OF CAPACITY AND SPEED



- Kilo- (K) = 1 thousand =  $10^3$  and  $2^{10}$
- Mega- (M) = 1 million =  $10^6$  and  $2^{20}$
- Giga- (G) = 1 billion =  $10^9$  and  $2^{30}$
- Tera - (T) = 1 trillion =  $10^{12}$  and  $2^{40}$
- Peta - (P) = 1 quadrillion =  $10^{15}$  and  $2^{50}$



# PROCESSOR SPEEDS & MEMORY SIZE



❖ **Hertz** = clock cycles per second (frequency)

$$1\text{MHz} = 1,000,000\text{Hz}$$

$$1\text{GHz} = 1000\text{ MHz}$$

– Processor speeds are measured in MHz or GHz.

❖ **Byte** = a unit of storage

$$1\text{KB} = 2^{10} = 1024\text{ Bytes}$$

$$1\text{MB} = 2^{20} = 1,048,576\text{ Bytes}$$

– Main memory (RAM) is measured in MB

--Disk storage is measured in GB for small systems, TB for large systems.



# MEASURES OF TIME AND SPACE



- Millie- (m) = 1 thousandth =  $10^{-3}$
- Micro- ( $\mu$ ) = 1 millionth =  $10^{-6}$
- Nano- (n) = 1 billionth =  $10^{-9}$
- Pico- (p) = 1 trillionth =  $10^{-12}$
- Femto- (f) = 1 quadrillionth =  $10^{-15}$



# COMPUTER LEVEL STEPS

## Level 6: *The User Level:*

- Program execution and user interface level.
- The level with which we are most familiar.

## Level 5: *High-Level Language Level:*

- The level with which we interact when we write programs in languages such as C, Pascal and Java.

## Level 4: *Assembly Language Level:*

- Acts upon assembly language produced from Level 5, as well as instructions programmed directly at this level.

Level 6	User	Executable Programs
Level 5	High Level Language	C++ , Java
Level 4	Assembly Language	Assembly Code
Level 3	System Software	Operating System
Level 2	Machine	Instruction Set Architecture
Level 1	Control	Microcode
Level 0	Digital Logic	Circuits , Gates

# COMPUTER LEVEL STEPS

## Level 3: **System Software Level:**

- Controls executing processes on the system.
- Protects system resources.
- Assembly language instructions often pass through Level 3 without modification.

## Level 2: **Machine Level:**

- Also known as the Instruction Set Architecture (ISA) Level.
- Consists of instructions that are particular to the architecture of the machine.
- Programs written in machine language need no compilers, interpreters, or assemblers.

Level 6	User	Executable Programs
Level 5	High Level Language	C++ , Java
Level 4	Assembly Language	Assembly Code
Level 3	System Software	Operating System
Level 2	Machine	Instruction Set Architecture
Level 1	Control	Microcode
Level 0	Digital Logic	Circuits , Gates

# COMPUTER LEVEL STEPS

## Level 1: **Control Level:**

- A **control unit** decodes, executes instructions and moves data through the system.
- Control units can be **micro programmed or hardwired**.
- A **micro program** is a program written in a low level language that is implemented by the hardware.
- **Hardwired control units** consist of hardware that directly executes machine instructions.

## Level 0: **Digital Logic Level:**

- This level is where we find digital circuits (the chips).
- Digital circuits consist of **gates and wires**.
- These components implement the mathematical logic of all other levels.

Level 6	User	Executable Programs
Level 5	High Level Language	C++ , Java
Level 4	Assembly Language	Assembly Code
Level 3	System Software	Operating System
Level 2	Machine	Instruction Set Architecture
Level 1	Control	Microcode
Level 0	Digital Logic	Circuits , Gates



**Computer Science Dept.**

**THANK  
YOU**



**By:**

Ali Saadoon Ahmed



## Computer Science Dept.



Department	Computer Science	القسم:
Subject Name:	Microprocessors - (3)	أسم المادة :
Year of Study:	2023-2024	السنة الدراسية:
Term:	Second Term	الفصل الدراسي:
Email	<a href="mailto:ali.sadoon@uoa.edu.iq">ali.sadoon@uoa.edu.iq</a>	Email
Instructor Name:	Asst. Lect. Ali Saadoon Ahmed	أسم التدريسي:



# OUTLINE

- Microcomputers and Microprocessors
- Evolution of Intel 8086 Family Microprocessors
- Pipelining and Registers
- Introduction to Assembly Programming



# MICROCOMPUTERS AND MICROPROCESSORS



There are three major parts of a Computer System.

- 1. *Central Processing Unit (CPU)*:** Also simply called as the microprocessor acts as the brain coordinating all activities within a computer.
- 2. *The Memory*:** The program instructions and data are primarily stored.
  - ❖ 8085 has A Max. memory capacity of 64 KB, while 8086 has Max. memory 1 MB.
- 3. *The Input/output (I/O) Devices*:** Allow the computer to input information for processing and then output the results. I/O Devices are also known as computer peripherals.

# MICROCOMPUTERS AND MICROPROCESSORS

The CPU is connected to memory and I/O devices through a strip of wires called a *bus*. The bus inside a computer carries information from place to place. In every computer there are three types of busses:

1. **Address Bus:** The address bus is used to identify the memory location or I/O device the processor intends to communicate with. The width of the Address Bus ranges from 20 bits (8086) to 36 bits for (Pentium II).

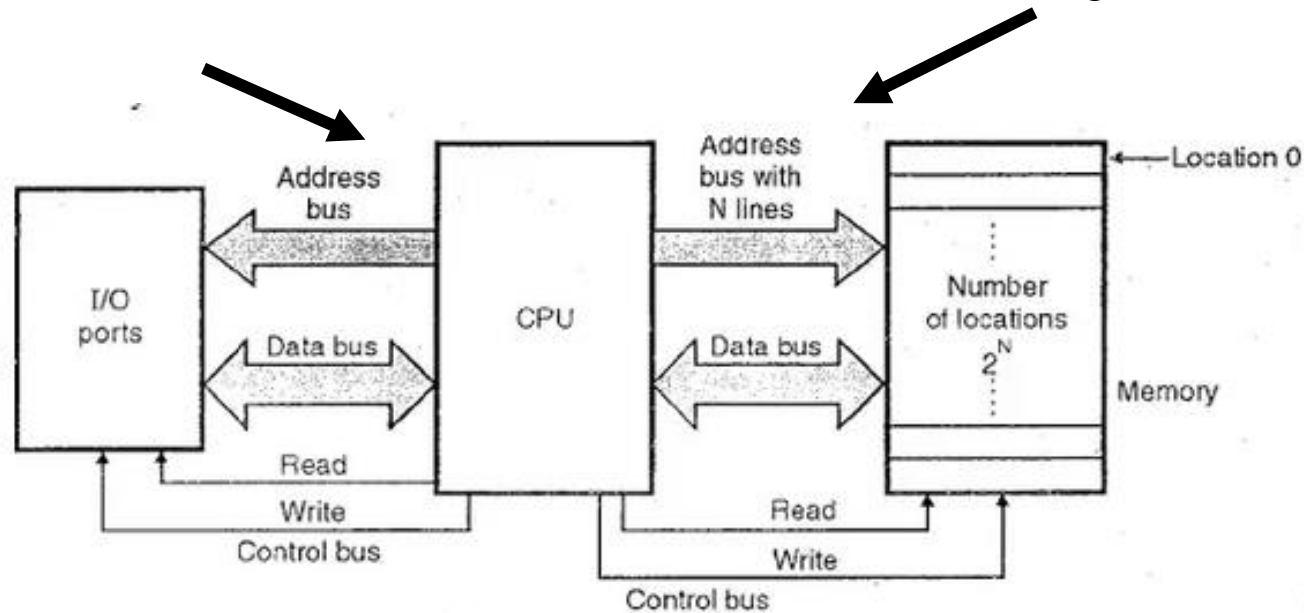


Fig. 1.2.3 : The three types of buses and their utility

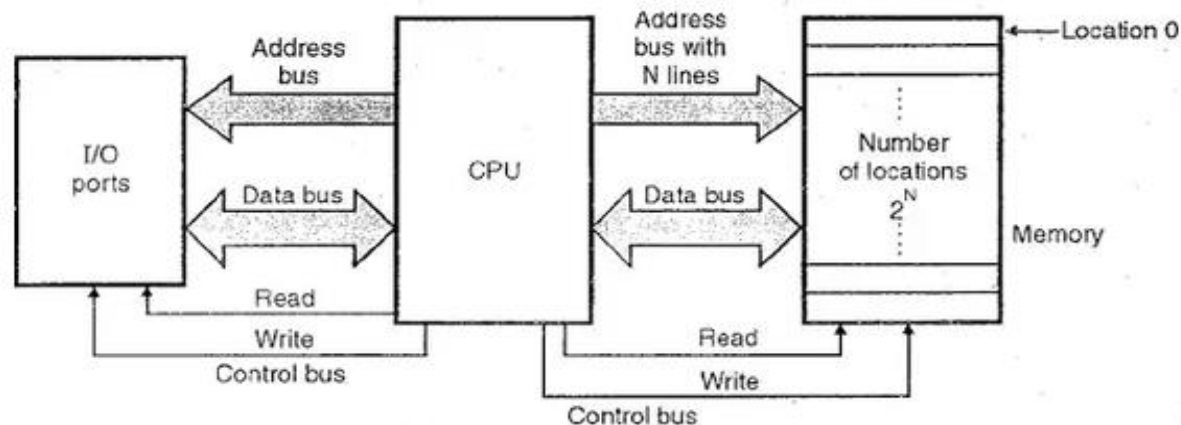


# MICROCOMPUTERS AND MICROPROCESSORS

**2. Data Bus:** Data bus is used by the CPU to get data from User to send data to the memory or the I/O devices. The width of a microprocessor is used to classify the microprocessor. The size of data bus of Intel microprocessors vary between 8-bit (8085) to 64-bit (Pentium).

**3. Control Bus.** How can we tell if the address on the bus is memory address or an I/O device address? This where the control bus comes in. Each time the processor outputs an address it also activates one of the four control bus signals: Memory Read, Memory Write, I/O Read and I/O Write.

❖ The **address** and **control** bus contains output lines only, therefore it is unidirectional, but the **data** bus is bidirectional





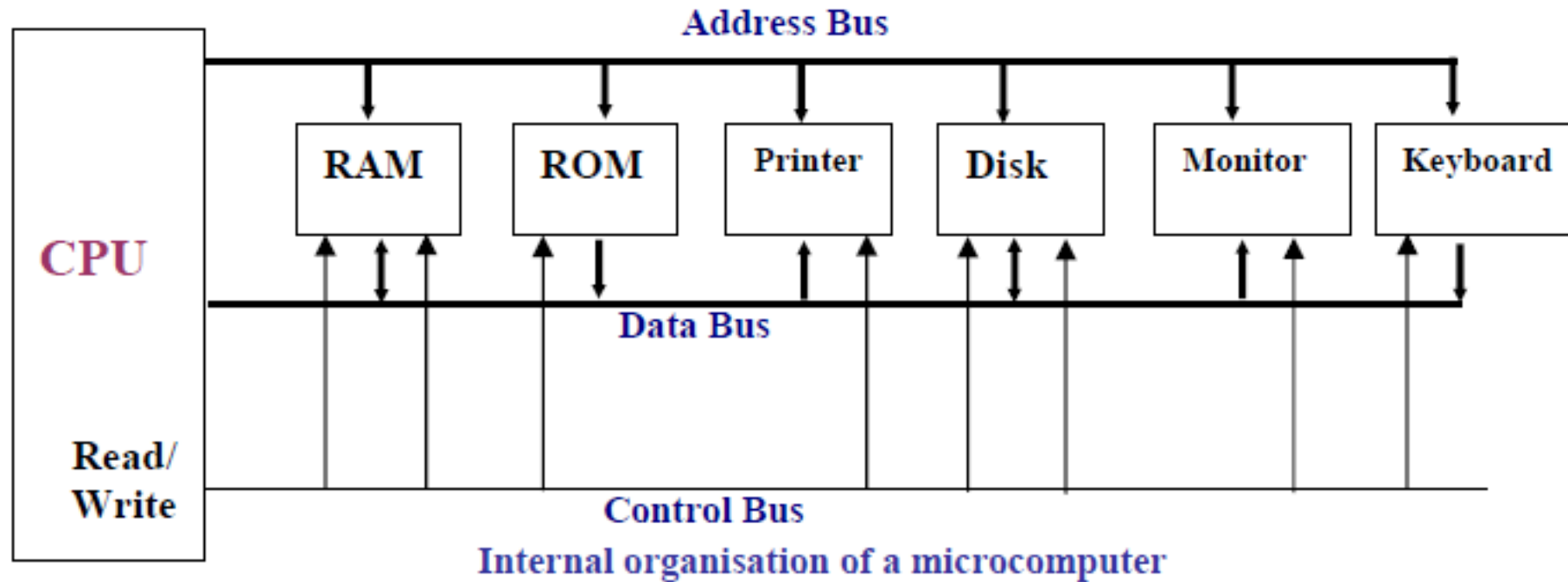
# MICROCOMPUTERS AND MICROPROCESSORS



There two types of memory used in microcomputers:

- **RAM (Random Access Memory/ Read-Write memory):** is used by the computer for the temporary storage of the programs that is running. Data is lost when the computer is turned off. So known as *volatile* memory.
- **ROM (Read Only Memory):** the information in ROM is permanent and not lost when the power is turned off. Therefore, it is called *nonvolatile* memory.
- Note that RAM is sometimes referred as *primary storage*, where magnetic /optical disks are called *secondary storage*.

# MICROCOMPUTERS AND MICROPROCESSORS





# MICROCOMPUTERS AND MICROPROCESSORS



- **Inside the CPU:**
  - A program stored in the memory provides instructions to the CPU to perform a specific action. This action can be a simple addition. It is function of the CPU to *fetch* the program instructions from the memory and *execute* them.
1. The CPU contains a number of *registers* to store information inside the CPU **temporarily**. Registers inside the CPU can be 8-bit, 16-bit, 32-bit or even 64-bit depending on the CPU.
  2. The CPU also contains *Arithmetic and Logic Unit (ALU)*. The ALU performs arithmetic (add, subtract, multiply, divide) and logic (AND, OR, NOT) functions.



# MICROCOMPUTERS AND MICROPROCESSORS

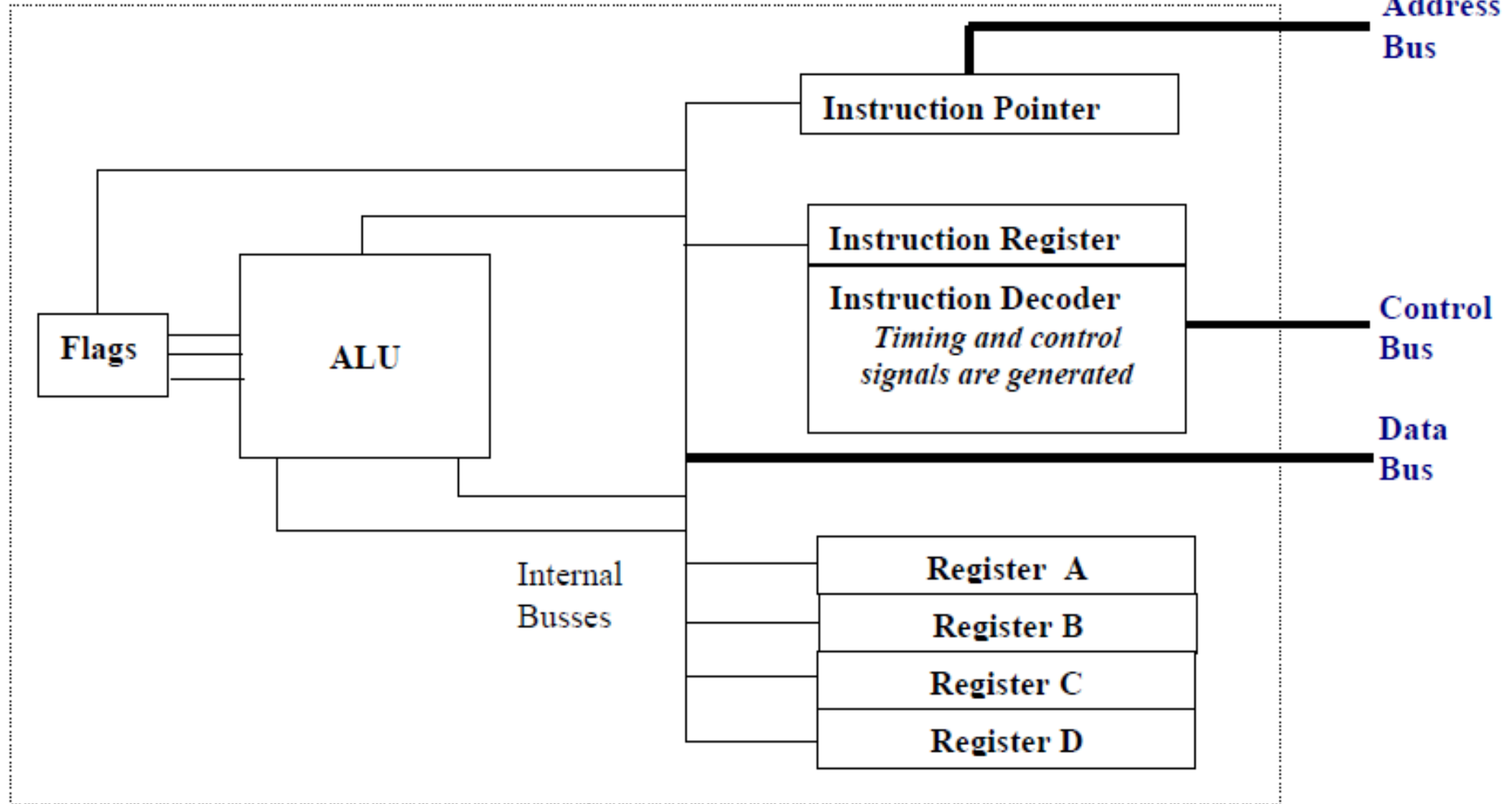


- **Inside the CPU:**

3. The CPU contains a program counter also known as the *Instruction Pointer* to point the address of the next instruction to be executed.

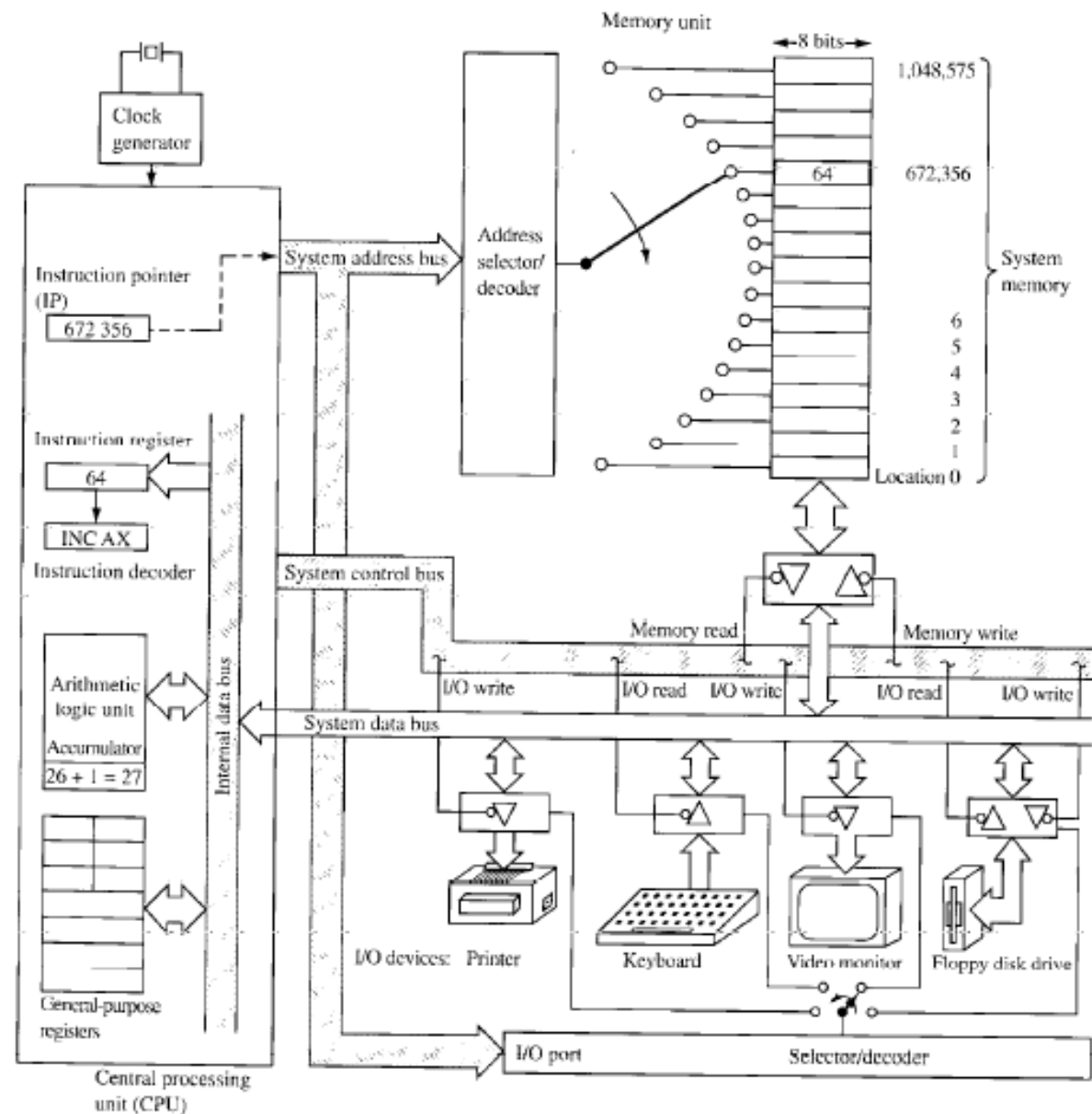
4. *Instruction Decoder* is a kind of dictionary which is used to interpret the meaning of the instruction fetched into the CPU. Appropriate control signals are generated according to the meaning of the instruction.

# MICROCOMPUTERS AND MICROPROCESSORS



Internal block diagram of a CPU

# MICROCOMPUTERS AND MICROPROCESSORS





# EVOLUTION OF INTEL 80X86 FAMILY MICROPROCESSORS

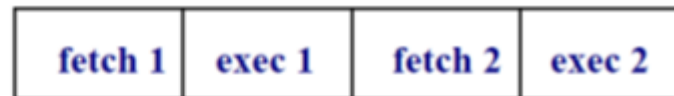
Processor	Year Intro.	Transistors	Clock Rate (MHz.)	External Data Bus	Internal Data Bus	Add. Bus
4004	1971	2,250	0.108	4	8	12
8008	1972	3,500	0.200	8	8	14
8080	1974	6,000	3	8	8	16
8085	1976	6,000	6	8	8	16
8086	1978	29,000	10	16	16	20
8088	1979	29,000	10	8	16	20
80286	1982	134,000	12.5	16	16	25
80386DX	1985	275,000	33	32	32	32
80386SX	1988	275,000	33	16	32	24
Pentium C	1993	3,100,000	66 -200	64	32	32
Pentium MMX	1997	4,500,000	300	64	32	32
Pentium Pro	1995	5,500,000	200	64	32	36
Pentium II	1997	7,500,000	233-450	64	32	36
Pentium III	1999	9,500,000	550-733	64	32	36
Itanium	2001	30,000,000	800-...	128	64	64



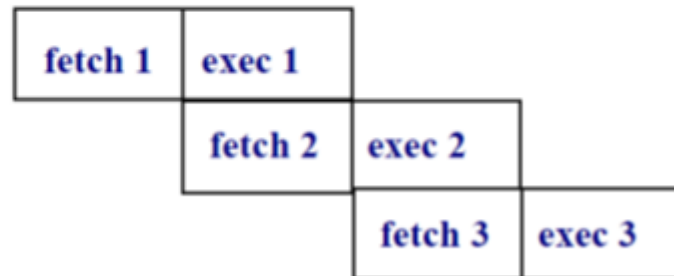
# PIPELINING AND REGISTERS

- **Pipelining**
- In the 8085 microprocessor, the CPU could either fetch or execute at a given time. CPU had to fetch an instruction from the memory, then execute it, then fetch again and execute it and so on..
- Pipelining is the simplest form to allow the CPU to *fetch* and *execute* at the same time. Note that the fetch and execute times can be different.

nonpipelined  
(e.g.. 8085)

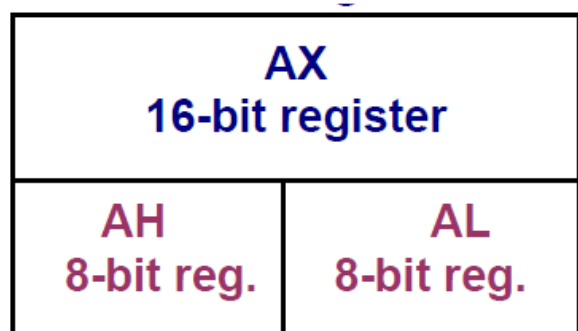


pipelined  
(e.g. 8086)

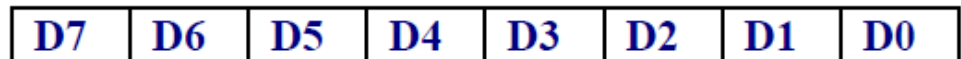


# PIPELINING AND REGISTERS

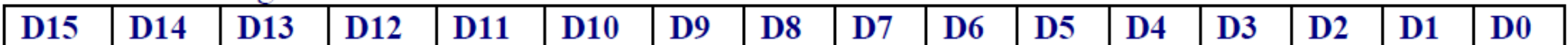
- *Registers of 8086 Microprocessor*
- In the CPU, registers are used store information temporarily. The information can be **one or two bytes** of data, or the address of data.
- In 8088/8086 general-purpose registers can be accessed as either 16-bit or 8-bit registers. All other registers can be accessed as full 16-bit registers.



8-bit register:



16-bit register:





# PIPELINING AND REGISTERS

Different registers are used for different functions. Registers will be explained later within the context of instructions and their applications.

- The first letter of each general register indicates its use.
- **AX** is used for the **Accumulator**.
- **BX** is used for **Base Addressing Register**.
- **CX** is used for **Counter Loop Operations**.
- **DX** is used to point out **data in I/O operations**.

## Registers of 8086

Category	Bits	Register Names
General	16	AX, BX, CX, DX
	8	AH, AL, BH, BL, CH, CL, DH, DL
Pointer	16	SP (stack pointer), BP (base pointer)
Index	16	SI (source index), DI (destination index)
Segment	16	CS (code segment), DS (data segment) SS (stack segment), ES (extra segment)
Instruction	16	IP (instruction pointer)
Flag	16	FR (flag register)



# INTRODUCTION TO ASSEMBLY PROGRAMMING

- **Low / High level languages:**
- Assembly Language is a low-level language. Deals directly with the internal structure of CPU. **Assembler** translates Assembly language program into machine code.
- In high-level languages, Pascal, Basic, C; the programmer does not have to be concerned with internal details of the CPU. **Compilers** translate the program into machine code.

## MOV instruction

MOV destination, source; copy source operand to destination

mnemonic                      operands



# INTRODUCTION TO ASSEMBLY PROGRAMMING

## Example: (16-bit)

```
MOV    CX,468FH    ;move 468FH into CX (now CH =46 , CL=8F)
MOV    AX,CX      ;move/copy the contents of CX into AX (now AX=CX=468FH)
MOV    BX,AX      ;now BX=AX=468FH
MOV    DX,BX      ;now DX=BX=468FH
MOV    DI,AX      ;now DI=AX=468FH
MOV    SI,DI      ;now SI=DI=468FH
MOV    DS,SI      ;now DS=SI=468FH
MOV    BP,DS      ;now BP=DS=468FH
```



# INTRODUCTION TO ASSEMBLY PROGRAMMING

## ➤ *Important points:*

- Values cannot be loaded directly into (CS,DS,SS and ES)

```
MOV AX,1234H ; load 1234H into AX
```

```
MOV SS,AX ;load the value in AX into SS
```

- Sizes of the values:

```
MOV BX,2H ; BX=0002H, BL:02H, BH:00H
```

```
MOV AL,123H ; illegal (larger than 1 byte)
```

```
MOV AX,3AFF21H ; illegal (larger than 2 bytes)
```



# INTRODUCTION TO PROGRAM SEGMENTS

- Assembly Language Program consists of three segments:
- ***Code Segment*** : contains the program code (instructions)
- ***Data Segment*** : used to store data (information) to be processed by the program
- ***Stack Segment***: used to store information temporarily.



**Computer Science Dept.**

**THANK  
YOU**



**By:**

**Asst. Lect. Ali Saadoon Ahmed**





## Computer Science Dept.



Department	Computer Science	القسم:
Subject Name:	Microprocessors - (4)	أسم المادة :
Year of Study:	2023-2024	السنة الدراسية:
Term:	Second Term	الفصل الدراسي:
Email	<a href="mailto:ali.sadoon@uoa.edu.iq">ali.sadoon@uoa.edu.iq</a>	Email
Instructor Name:	Ali Saadoon Ahmed	أسم التدريسي:

Ali Saadoon Ahmed

Al-Maaref University College

Microprocessors



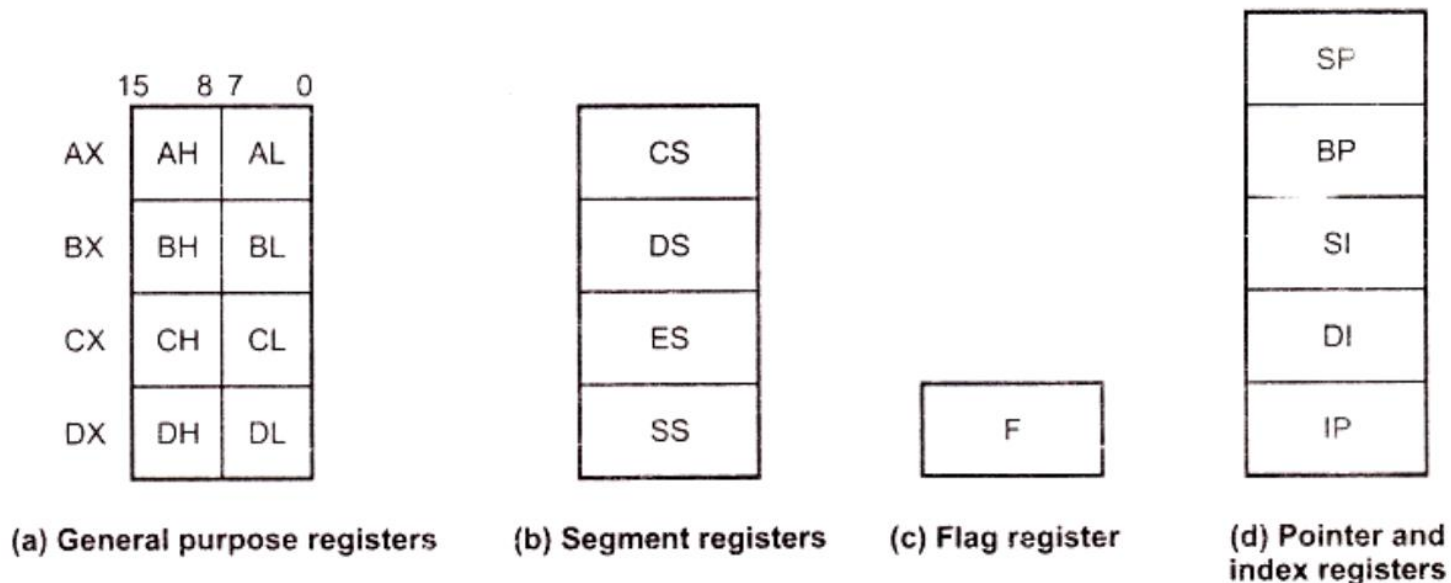
# OUTLINE



- ❖ **Register Organization**
- ❖ **Introduction to Assembly Programming**
- ❖ **Data Transfer Instructions In 8086 Microprocessor**
- ❖ **Arithmetic instruction**
- ❖ **Segment Addressing.**
- ❖ **Logical address and Physical address:**

## ● Register Organization

The 8086 has a powerful set of registers. It includes general purpose registers, segment registers, pointers and index registers, and flag register. The register organization of 8086. The registers shown are accessible to programmer. As shown in the Fig, all the registers of 8086 are 16-bit registers.





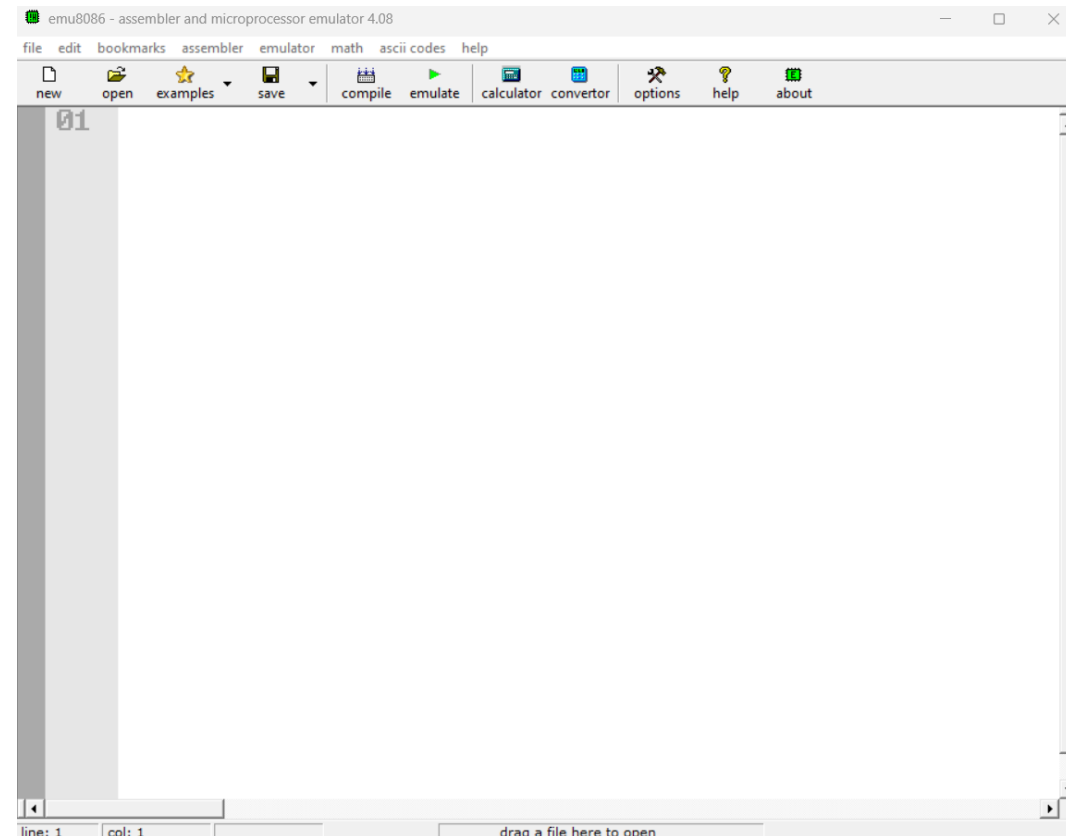
# Assembly Programming



Each personal computer has a microprocessor that manages the computer's arithmetical, logical, and control activities.

## The Environment Used

❖ EMU8086 - MICROPROCESSOR EMULATOR





# Data Transfer Instructions In 8086 Microprocessor

❖ Data transfer instructions are the instructions that transfer data in the microprocessor. They are also called copy instructions.

## Types of Data transfer instructions :

### 1. Move instructions:

These instructions are used to move data from one memory location to another or between a memory location and a register.

### MOV instruction

MOV                      destination, source; copy source operand to destination





# Data Transfer Instructions In 8086 Microprocessor



## MOV instruction

### Example: (8-bit )

```
MOV CL,55H      ;move 55H into register CL
MOV DL,CL       ;move/copy the contents of CL into DL (now DL=CL=55H)
MOV BH,DL       ;move/copy the contents of DL into BH (now DL=BH=55H)
MOV AH,BH       ;move/copy the contents of BH into AH (now AH=BH=55H)
```

### Example: (16-bit)

```
MOV CX,468FH    ;move 468FH into CX (now CH =46 , CL=8F)
MOV AX,CX       ;move/copy the contents of CX into AX (now AX=CX=468FH)
MOV BX,AX       ;now BX=AX=468FH
MOV DX,BX       ;now DX=BX=468FH
MOV DI,AX       ;now DI=AX=468FH
MOV SI,DI       ;now SI=DI=468FH
MOV DS,SI       ;now DS=SI=468FH
MOV BP,DS       ;now BP=DS=468FH
```



## Data Transfer Instructions In 8086 Microprocessor

These types of operands are Supported:

*mov REG, memory*

*mov memory, REG*

*mov REG, REG*

*mov memory, immediate*

*mov REG, immediate*

*mov SREG, memory*

*mov memory, SREG*

*mov REG, SREG*

*mov SREG, REG*

**Note:**

**REG:** AX, BX, CX, DX, AH, AL, BH, BL, CH, CL, DH, DL, DI, SI, BP, SP.

**Immediate:** 5, 24, 3FH, 1000101B, etc

**MEMORY:** [BX], [SI] +BX, [DI] +BX, etc...

The MOV instruction cannot:

- 1- Set the value of the CS and IP registers.
- 2- Copy value of one segment register to another segment register (should copy to general register first).
- 3- Copy immediate value to segment register (should copy to general register first).



# Data Transfer Instructions In 8086 Microprocessor



## Procedure:

Write a program in 8086 trainer to perform the following tasks by using MOV instruction:

```
MOV AX, 1000H  
MOV DS, AX  
MOV AX, 1234H  
MOV BX, 200H  
MOV [200H], AX  
MOV CX, [200H]  
MOV AX, 5566H  
MOV [BX+2], AX  
MOV DX, [BX+2]  
MOV SI, 0030H  
MOV DI, 0040H  
MOV AX, 0ABCDH  
MOV [BX+SI], AH  
MOV [BX+DI], AL  
MOV AL, [BX+SI]  
MOV AH, [BX+DI]  
End the program
```

Run the program and write the result of each register and memory being used.





# Data Transfer Instructions In 8086 Microprocessor



## • XCHG INSTRUCTION

used to exchange (swap) data between two general-purpose registers or between a general-purpose register and a storage location in memory.

The forms of the XCHG instruction and its allowed operands are shown in Figures below:

Mnemonic	Meaning	Format	Operation	Flags affected
XCHG	Exchange	XCHG D,S	(D) ↔ (S)	None

(a)

Destination	Source
Accumulator	Reg16
Memory	Register
Register	Register
Register	Memory

(b)



# Arithmetic instruction

The instruction set of the 8086 microprocessor contains a variety of arithmetic instructions. They include instructions for the addition, subtraction, multiplication, and division operations. These operations can be performed on numbers expressed in a variety of numeric data formats.

## Addition instruction: ADD, INC, DEC

**ADD instruction:** The ADD instruction is used to add the contents of a source operand to the contents of the destination operand. The result is put into the location of the destination operand. In general, the result of executing ADD is expressed as:

**ADD** Destination, Source

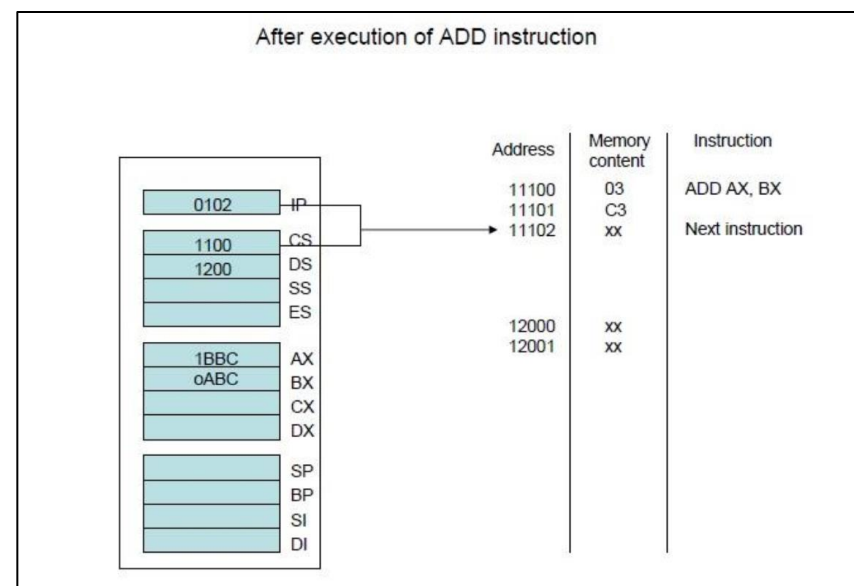
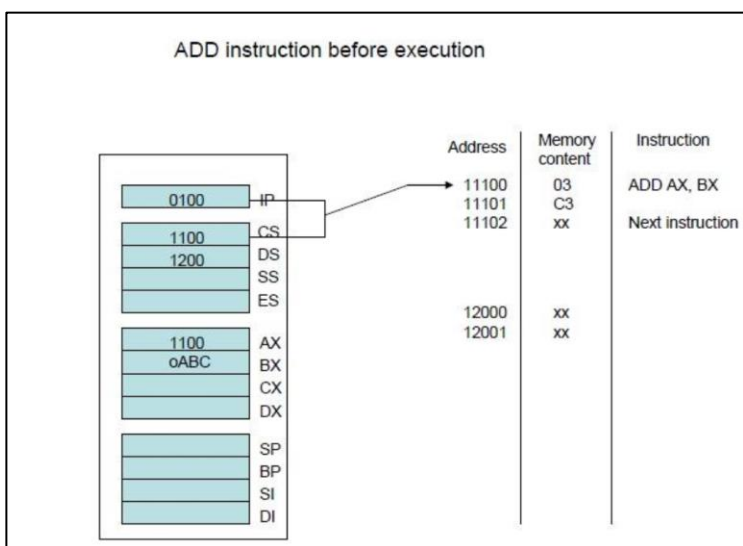
# Arithmetic instruction

## Addition instruction

**Ex:** Assume that AX and BX registers contain 1100H and 0ABCH, respectively. What is the result of executing the instruction?

**ADD AX, BX**

The content of the source (BX) will be added to the content of the destination (AX) to give  $0ABC + 1100 = 1BBC$  in the destination AX. The process of executing this instruction is shown in the following figures.





# Arithmetic instruction

## INC instruction

It increments the byte or word by one.

- ❖ The INC instruction adds 1 to any register or memory location, except a segment register.
- ❖ The operand can be a register or memory location.

Example:

INC **AX**; adds 1 to AX register

INC **DX**; adds 1 to DX register

## DEC instruction

Example:

DEC **AX**; SUB 1 to AX register

DEC **DX**; SUB 1 to DX register



# Arithmetic instruction

## SUB instruction:

The subtract (SUB) instruction is used to subtract the value of a source operand from a destination operand. The result of this operation in general is given as

$$(\text{Destination}) - (\text{Source}) \rightarrow (\text{D})$$

### ❖ Immediate Subtraction

Immediate subtraction is employed whenever constant or known data are subtracted.

Example:

```
MOV CH, 22H
```

```
SUB CH, 44H
```

The subtraction is stored in the CH register.

### ❖ Memory-to-Register Subtraction

Moves memory data to be subtracted to a register.

Example:

```
MOV DI, OFFSET NUMB
```

```
MOV AL, 0
```

```
SUB AL, [DI]
```

```
SUB AL, [DI+1]
```

# Arithmetic instruction

## SUB instruction:

Assembly Language	Operation
SUB CL,BL	CL = CL – BL
SUB AX,SP	AX = AX – SP
SUB ECX,EBP	ECX = ECX – EBP
SUB DH,6FH	DH = DH – 6FH

## Instructions MUL:

The 8086 multiply instructions have the general forms **MUL** **SOURCE**

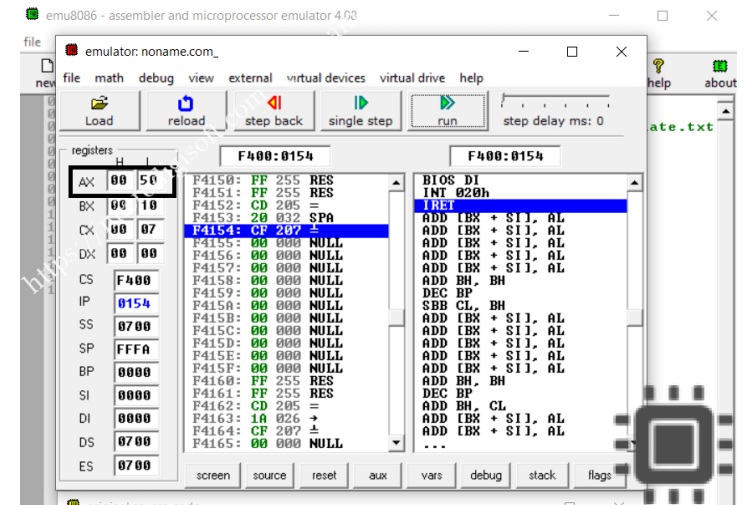
8-Bit Multiplication

Example-Assembly Language Program

MOV AL, 5H ; a byte is moved to AL

MOV BL, 10H ; immediate data must be in BL register

MUL BL



# Arithmetic instruction

## Instructions MUL:

### 16-Bit Multiplication

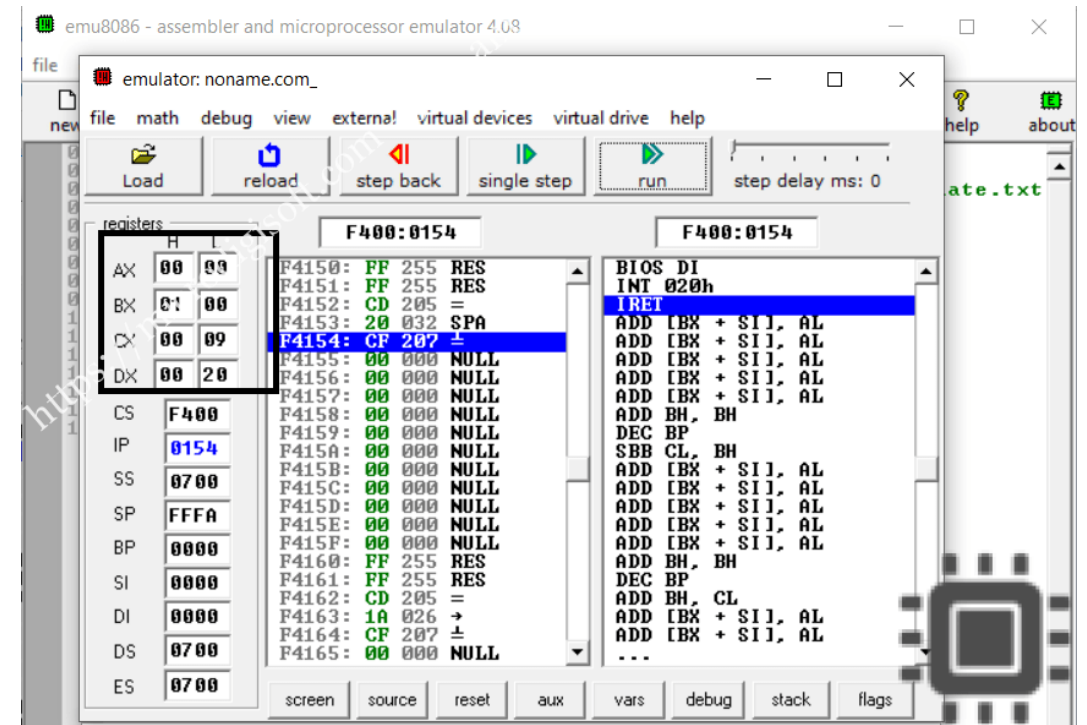
Example-Assembly Language Program

The following statements multiply the 16-bit value 2000H by 0100H. The Carry flag is set because the upper part of the product (located in DX) is not equal to zero:

MOV AX, 2000H ; a word is moved to AX

MOV BX, 0100H ; immediate data must be in BX register

MUL BX





# Arithmetic instruction

## Instructions DIV:

**Q: Write an assembly language program to compute  $C=(5/9)*(F-32)$**

```
MOV AX,05  
MOV BX,09  
DIV BX  
MOV BX,0FH  
MOV CX, 32H  
SUB BX,CX  
MUL BX  
RET
```



# Arithmetic instruction

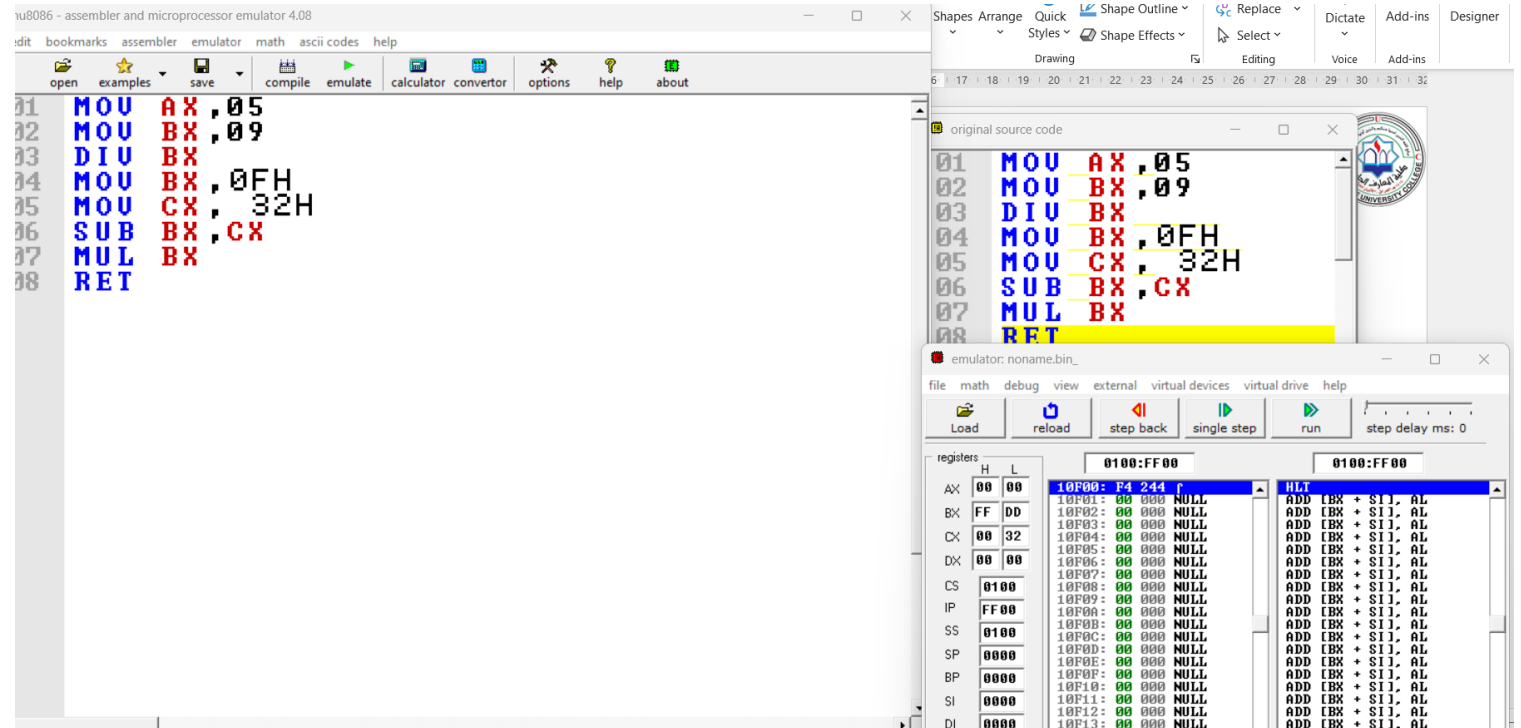
## Instructions DIV:

**Q: Write an assembly language program to compute  $C=(5/9)*(F-32)$**

```

MOV AX,05
MOV BX,09
DIV BX
MOV BX,0FH
MOV CX, 32H
SUB BX,CX
MUL BX
RET

```



The screenshot shows the 'original source code' window with the following assembly code:

```

01 MOV AX,05
02 MOV BX,09
03 DIV BX
04 MOV BX,0FH
05 MOV CX, 32H
06 SUB BX,CX
07 MUL BX
08 RET

```

The 'registers' window shows the state of the registers:

Register	H	L	Value
AX	00	00	10F00: F4 244 c
BX	FF	DD	10F01: 00 000 NULL
CX	00	32	10F02: 00 000 NULL
DX	00	00	10F03: 00 000 NULL
CS	0100		10F04: 00 000 NULL
IP	FF00		10F05: 00 000 NULL
SS	0100		10F06: 00 000 NULL
SP	0000		10F07: 00 000 NULL
BP	0000		10F08: 00 000 NULL
SI	0000		10F09: 00 000 NULL
DI	0000		10F0A: 00 000 NULL



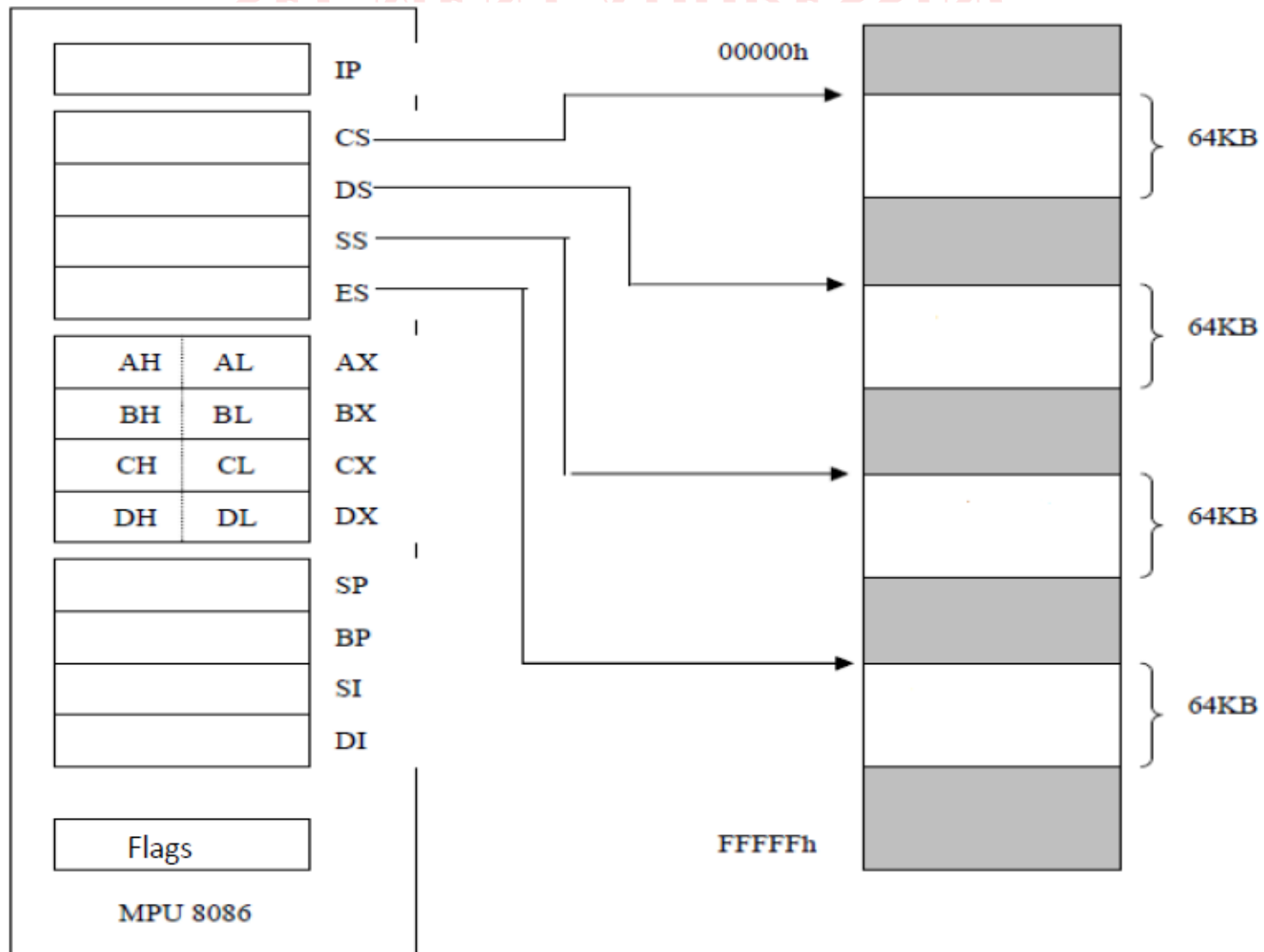
# SEGMENT ADDRESSING



A *segment* is an area of memory that includes up to 64K bytes and begins on an address evenly divisible by 16. The segment size of 64K bytes came because the 8085 microprocessor could address a maximum of 64k Bytes of physical memory since it had only 16 pins for the address lines ( $2^{16}=64K$ ).

Whereas in the 8085 there was only 64K bytes of memory for all *code*, *data* and *stack* information. For this reason, the 8086/88 microprocessor can only handle a maximum of 64K bytes of code and 64K bytes of data and 64K bytes of stack at any given time.

# SEGMENT ADDRESSING



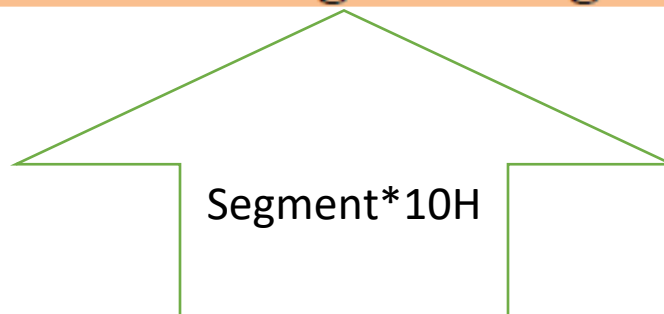


# LOGICAL ADDRESS AND PHYSICAL ADDRESS:



- The BIU contains a dedicated *Adder* which is used to generate the 20 bits physical address that is out put on the address bus.
- This address is formed by adding the 16-bit shifting segment address with the offset address.

**Physical address (PA) = Shifted Segment register + Offset**





# LOGICAL ADDRESS AND PHYSICAL ADDRESS:



- **Segment address:** It is located with one of the segment register and defines the beginning address of any 64 KB memory segment.
- **Offset address:** is a location within a 64 KB segment range, therefore, an offset address can range from 0000 – FFFF.
- **Logical address:** consists of a segment value and offset address. Sometimes the segment and offset is written as (**segment: offset**).



# LOGICAL ADDRESS AND PHYSICAL ADDRESS:

- The different combination used in 8086 MP for Segment: offset address shown in table below

Logical address	Segment	Offset	Purpose
CS:IP	CS	IP	Instruction address
SS:SP or SS:BP	SS	SP or BP	Stack address
DS:BX, DS:DI or DS:SI	DS	BX, DI or SI	Data address
ES:DI	ES	DI	String destination address

# LOGICAL ADDRESS AND PHYSICAL ADDRESS:

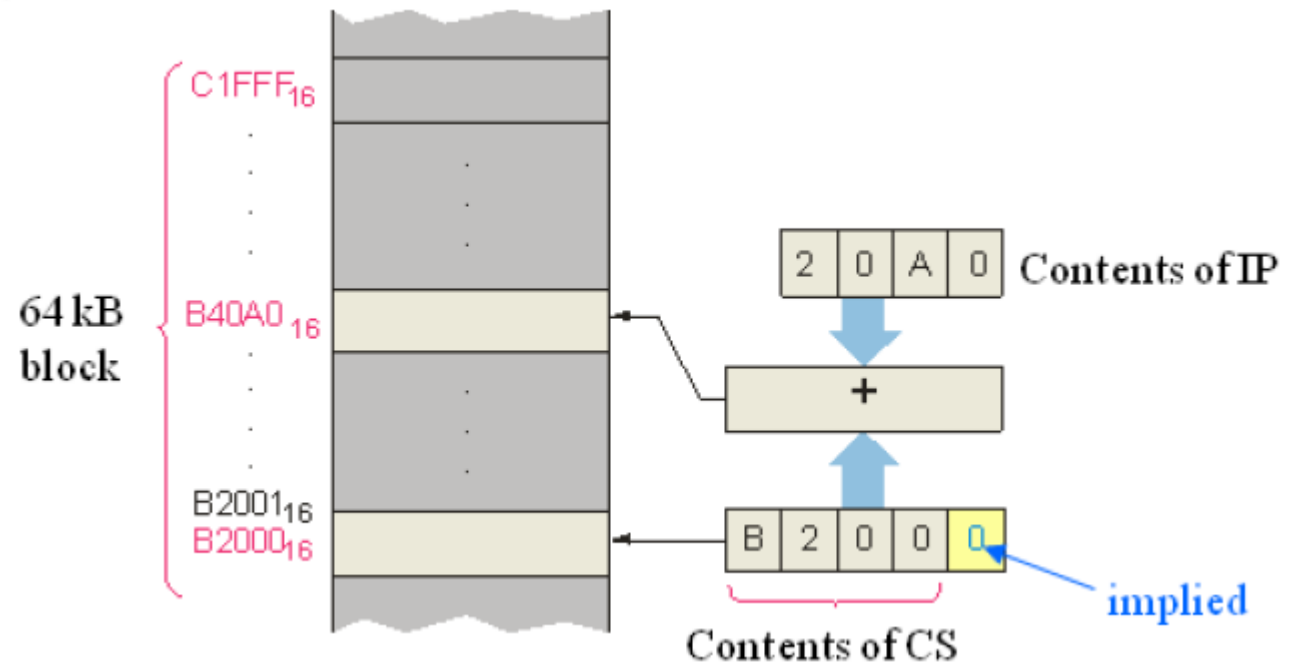
Assume  $IP = 20A0_{16}$  and  $CS=B200_{16}$ .

- What is the location of the start and end of the block?
- What physical address is formed?

The addressing is diagramed:

a) The start of the block is at  $B2000_{16}$ ; it ends at  $B2000_{16} + FFFF_{16} = C1FFF_{16}$

b) The physical address is  $B2000_{16} + 20A0_{16} = B40A0_{16}$





# LOGICAL ADDRESS AND PHYSICAL ADDRESS:



## • *Example 1:*

If CS=24F6h and IP=634Ah, show:

- a- The Logical address and the offset address.
- b- calculate the physical address.
- c- calculate the lower range and upper range of the code segment.

## • *Sol:*

a- 24F6:634A, the offset is 634A

b-  $PA = 24F60 + 634A \longrightarrow PA = 2B2AA$

c- The lower range  $24F60 + 0000 = 24F60$

The upper range of the code segment is  $24F60 + FFFF = 34F5F$





# LOGICAL ADDRESS AND PHYSICAL ADDRESS:



## • *Example2:*

If DS = 7FA2h and the offset is 438Eh, show:

- a- The logical address, and calculate :
- b- The physical address.
- c- The lower range and upper range of the code segment.

## • *Sol:*

a- 7FA2:438E

b- PA= 7FA20+438E=83DAE

c- The lower range 7FA20+0000=7FA20

The upper range of the code segment is 7FA20+FFFF=8FA1F



**Computer Science Dept.**

**THANK  
YOU**



**By:**

**Ali Saadoon Ahmed**