A logo of a university

Description automatically generated

**قــســــــــــم الامــــــــــــــــــــــــــن الــــــــــــــــــــــــــــــــــــسيبرانــــــــــــــــــــــــــــــــــــي**

**Department of Cyber Security**

**MICROPROCESSORS**

**Lecture: (4)**

**microprocessor**

**Class: Second**

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

# C:\Users\HP\Desktop\photo_2025-02-21_21-46-59.jpg

# **Control & Status Signals:**

# ALE: Address latch enable

# RD: Read control signal.

# WR: Write control signal.

# IO/M, S1 and SO: Status signals

# **Power Supply & Clock Frequency:**

# Vec: +5 V power supply

# Vss: Ground reference

# X1, X2: A crystal having frequency of 6 MHz is connected at these

# two pins

# CLK: Clock output

# **Externally Initiated and Interrupt Signals:**

# RESET IN: When the signal on this pin is low, the PC is set to 0,

# the buses are tri-stated and the processor is reset.

# RESET OUT: This signal indicates that the processor is being

# reset. The signal can be used to reset other devices.

# READY: When this signal is low, the processor waits for an

# integral number of clock cycles until it goes high.

# • HOLD: This signal indicates that a peripheral like DMA (direct

# memory access) controller is requesting the use of address and data bus.

# • HLDA: This signal acknowledges the HOLD request.

# • INTR: Interrupt request is a general-purpose interrupt.

# • INTA: This is used to acknowledge an interrupt.

# • RST 7.5, RST 6.5, RST 5,5 - restart interrupt: These are vectored

# interrupts and have highest priority than INTR interrupt.

# • TRAP: This is a non-mask able interrupt and has the highest priority. •

# Serial I/O Signals:

# • SID: Serial input signal. Bit on this line is loaded to D7 bit of register

# A using RIM instruction.

# • SOD: Serial output signal. Output SOD is set or reset by using SIM

# instruction.

# 

# **INSTRUCTION SET AND EXECUTION IN 8086**

# Based on the design of the ALU and decoding unit, the microprocessor

# manufacturer provides instruction set for every microprocessor. The

# instruction set consists of both machine code and mnemonics.

# An instruction is a binary pattern designed inside a microprocessor to

# perform a specific function. The entire group of instructions that a

# microprocessor supports is called instruction set. Microprocessor

# instructions can be classified based on the parameters such functionality,

# length and operand addressing.

# **Classification based on functionality:**

# **1-Data transfer operations:** This group of instructions copies data

# from source to destination. The content of the source is not altered.

# **2- Arithmetic operations:** Instructions of this group perform

# operations like addition, subtraction, increment & decrement. One

# of the data used in arithmetic operation is stored in accumulator

# and the result is also stored in accumulator.

# **3- Logical operations:** Logical operations include AND, OR, EXOR,

# NOT. The operations like AND, OR and EXOR uses two

# operands, one is stored in accumulator and other can be any

# register or memory location. The result is stored in accumulator.

# NOT operation requires single operand, which is stored in

# accumulator.

# **4- Branching operations:** Instructions in this group can be used to

# transfer program sequence from one memory location to another

# either conditionally or unconditionally.

# 5- Machine control operations: Instruction in this group control

# execution of other instructions and control operations like interrupt, halt

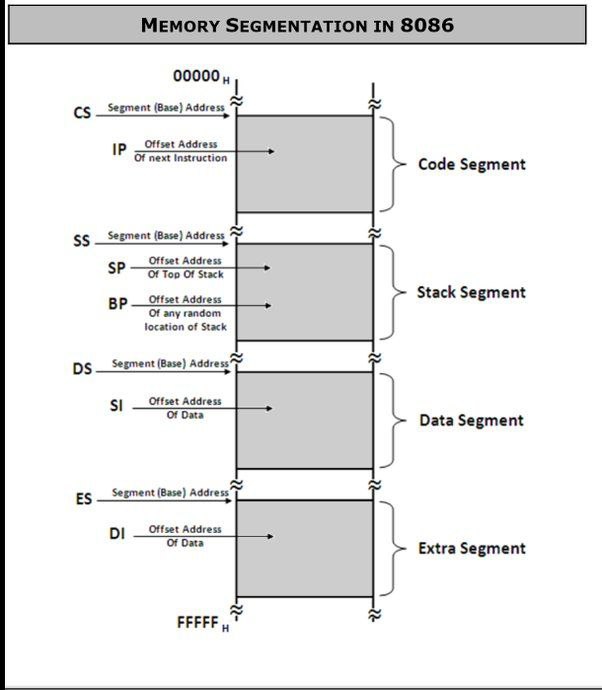
# etc.

# **Memory segmentation:**

# Segmentation is the process in which the main memory of the computer

# is logically divided into different segment and each segment has it own

# base address.



# **Segment Registers:**

# These are used to hold the the starting addresses of a specific

# memory segments inside the 8086 microprocessor.

# There is no 20 bit register inside 8086 microprocessor and this

# processor has only 16 bit register.

# The (1MB) memory was divided into equal segment of size 64KB.

# There are four segment register: Code segment (CS) register, the

# Data segment (DS) register, the stack segment (SS) register, and the

# extra segment (ES) register

# **1-Code segment (CS):** is a 16-bit register containing the start address of

# 64 KB code segment of memory. The processor uses CS segment for all

# accesses to instructions referenced by instruction pointer (IP) register.

# **2-Data segment (DS):** is a 16-bit register containing the start address of

# 64 KB data segment of memory. DS register with the source index (SI)

# OR destination index (DI) are used to access the data inside the data

# segment memory.

# **3-Extra segment (ES):** used to hold the starting address of Extra segment.

# EX register with the destination index (DI) are used to access the data

# inside the data segment memory.

# **4-Stack segment (SS):** is a 16-bit register containing address of 64KB

# segment with program stack. By default, the processor assumes that all

# data referenced by the stack pointer (SP) and base pointer (BP) registers

# is located in the stack segment. Segment

# 

# **Index or Pointer Registers**

# **1. Base Pointer (BP)** is a 16-bit register pointing to data in

# stack segment. It is usually used to store the destination of

# memory location (random address). BP register is usually

# used with (SS)register.

# **2. Stack Pointer (SP)** is a 16-bit register pointing to program

# stack, it is used to hold the address of the top of stack inside

# the stack segment.

# **3. Source Index (SI)** is a 16-bit register. Sl is used for store the

# offset memory location (source data) inside the data

# segment(DS).

# **4. Destination Index (DI**) is a 16-bit register. Used store the

# offset of memory location (destination data) inside the extra

# segment(ES).

# 

# **1. Instruction Pointer (IP)** is a 16-bit register. The IP, or

# program counter, is used to store the memory location of the

# next instruction to be executed.

# **Flag Register**

# determines the current state of the processor.

# They are modified automatically by CPU after mathematical operations, this allows to determine the type of the result, and to determine conditions to transfer control to other parts of the program. 8086 has 9 flags and they are divided into two categories:

# Status Flags : Status Flags represent result of last arithmetic or logical instruction

# executed. Conditional flags are as follows:

# **Carry Flag (CY):** This is carry bit. If some operations are generating

# carry after the operation this flag is set to 1

# **Auxiliary Flag (AY):** If an operation performed in ALU generates a

# carry/barrow from lower nibble.

# **Parity Flag (PF):** This is even parity flag. When result has even number

# of 1, it will be set to 1, otherwise 0 for odd number of 1s.

# **Zero Flag (ZF):** It is set if the result of arithmetic or logical operation is zero.

# **Sign Flag (SF):** After any operation if the MSB is 1, then it indicates

# that the number is negative. And this flag is set to 1.

# **Overflow Flag (OF):** It occurs when signed operation is too large to fit

# in the number of bits available to represent it.

# 

# **Logical address and physical address**

# physical address(PA) = shifted segment register + offsite

# EX1// If the data segment register contain 1000H and the offset register

# contain 2000H. find bellow:

# 1- Start address of this segment

# 2- Physical address.

# EX2// If the code Segment register contain 1400H and the IP

# register

# contain 1200H,find:

# 1- Start address of this segment

# 2- Physical address.

# EX3// the contents of the following register are

# CS=1111H,DS=3333H,SS=2526H,IP=1232H,SP=1100H,DI=0020H

# find bellow the Physical address for the address bytes in CS, DS,SS