Unit 7
Advanced Microprocessors
(9 Hrs.)

8086: logical block diagram and segments,

80286: Architecture, Registers, (Real/Protected mode), Privilege levels, descriptor cache, Memory access in GDT and LDT, multitasking, addressing modes, flag register

80386: Architecture, Register organization, Memory access in protected mode, Paging

Intel 8086

Introduction

- The Intel 8086 is a 16-bit microprocessor developed by Intel in 1978.
- It is the successor to the 8085 microprocessor and was a major step forward in microprocessor architecture.
- It introduced pipelining, segmented memory, and 16-bit data and address buses, enabling more powerful and efficient computing.
- It became the foundation for the x86 architecture, which is still widely used in modern computing systems.

Back with of Intel 8085 and 8086

Feature	**8085**	**8086**
Data Bus	8-bit	16-bit
Address Bus	16-bit (64 KB)	20-bit (1 MB)
ALU Size	8-bit	16-bit
Pipelining	No	Yes
Registers	Fewer	More (and wider)
Operating Modes	Single mode	Min/Max mode
Memory Segmentation	No	Yes
Instruction Set	Basic	Advanced

Features of 8086

- 16-bit data bus & 20-bit address bus
- Pipelined architecture (fetch & execute simultaneously)
- 14 registers (general-purpose, segment, pointer, index, flag)
- Operating frequency: 5 MHz (8086), 8 MHz (8086-2), 10 MHz (8086-1)
- Multiplexed address/data bus (AD0–AD15)
- Supports hardware interrupts (NMI & INTR)
- **Two modes of operation:**
  - Minimum mode (single processor)
  - Maximum mode (multiprocessor)

Why Pipeline?

In computing, **pipelining** improves the efficiency of instruction execution by allowing multiple instructions to overlap in execution—like an assembly line in a factory.

Basic Concept:

When an application runs:

APP → List of Programs → Instructions → Process → Thread

Each **instruction** (I) typically goes through four stages:

- **F**: Opcode Fetch
- **D**: Decode
- **E**: Execute
- **S**: Store

Let’s consider 4 instructions: **I1, I2, I3, I4**

◆ For Single Processor (No Pipelining):

Each instruction is executed one after the other:

Time to execute = I1 + I2 + I3 + I4

= (F1 + D1 + E1 + S1) + (F2 + D2 + E2 + S2) +

(F3 + D3 + E3 + S3) + (F4 + D4 + E4 + S4)

= 4 ns + 4 ns + 4 ns + 4 ns

= **16 ns**

◆ For Parallel Processor (With Pipelining):

Instructions are overlapped using pipeline stages:

Time to execute = I1 + I2 + I3 + I4							
P1	F1	D1	E1	S1			
P2		F2	D2	E2	S2		
P3			F3	D3	E3	S3	
P4				F4	D4	E4	S4

Time to execute = I1 + I2 + I3 + I4							
P1	**F1**	**D1**	**E1**	**S1**			
P2		**F2**	**D2**	**E2**	**S2**		
P3			**F3**	**D3**	**E3**	**S3**	
P4				**F4**	**D4**	**E4**	**S4**
	1	**1**	**1**	**1**	**1**	**1**	**1**

Total Time = 7 cycles × 1 ns (per stage) = **7 ns**

```
Cycle 1: F1
Cycle 2: D1   F2
Cycle 3: E1   D2   F3
Cycle 4: S1   E2   D3   F4
Cycle 5:      S2   E3   D4
Cycle 6:           S3   E4
Cycle 7:                S4
```

Total Time = 7 cycles × 1 ns (per stage) = **7 ns**

✅ **Benefit of Pipelining:**

- **Non-pipelined execution:** 4 instructions → 16 ns
- **Pipelined execution:** 4 instructions → 7 ns
- **Time saved:** 9 ns
- **Efficiency increased** by overlapping instruction stages.

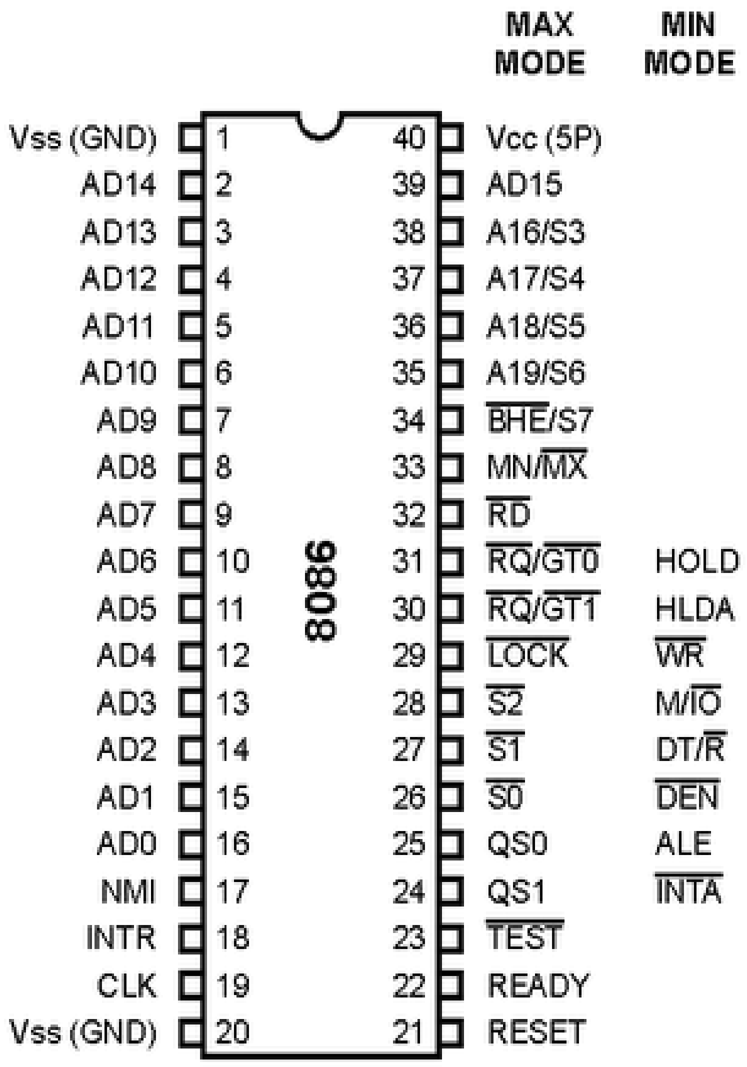
Pin Diagram of the Intel 8086 Microprocessor

Intel 8086 Microprocessor, which consists of **40 pins**. Each pin serves a specific function depending on whether the processor is operating in **minimum mode** (single processor system) or **maximum mode** (multiprocessor system).

🔷 Power & Clock Pins

- **Vcc (Pin 40)**: +5V power supply.
- **GND (Pins 1, 20)**: Ground connection.

- **CLK (Pin 19)**: Clock input. Provides timing for internal operations.
- **RESET (Pin 21)**: Resets the processor and sets program counter to zero.



Address/Data Bus (Multiplexed)

- **AD0 – AD15 (Pins 2–16, 39)**: These are multiplexed lines used to carry address (A0–A15) during the first part of the bus cycle and data (D0–D15) during the second part.

Address Bus (High)

- **A16/S3 – A19/S6 (Pins 35–38)**: These are multiplexed with status signals (S3–S6) and used for addressing (A16–A19) in memory access.

Control & Status Pins

- **ALE (Pin 25)**: Address Latch Enable. Indicates when AD0–AD15 contains an address.
- **DEN (Pin 26)**: Data Enable. Enables the external data bus transceivers.
- **DT/R (Pin 27)**: Data Transmit/Receive. Controls direction of data flow.
- **M/IO (Pin 28)**: Distinguishes memory or I/O operation.
- **RD (Pin 32)**: Read control signal.
- **WR (Pin 29)**: Write control signal.
- **READY (Pin 22)**: Indicates if peripheral is ready for data transfer.

- **TEST (Pin 23)**: Used for synchronization with external devices.

🔷 Interrupt Pins

- **INTR (Pin 18)**: Interrupt request (maskable).
- **NMI (Pin 17)**: Non-maskable interrupt. Has higher priority than INTR.

🔷 Minimum/Maximum Mode Selection

- **MN/M$\overline{X}$ (Pin 33)**:
    - Logic 1 → Minimum mode
    - Logic 0 → Maximum mode

🔷 Minimum Mode Specific Pins

(used only when MN/M$\overline{X}$ = 1)

- **INTA (Pin 24)**: Interrupt acknowledge.
- **HOLD (Pin 31)**: DMA controller requests control of buses.
- **HLDA (Pin 30)**: DMA controller granted access to buses.
- **WR, RD, ALE, DEN, DT/R, M/IO** → Active in this mode.

🔷 Maximum Mode Specific Pins

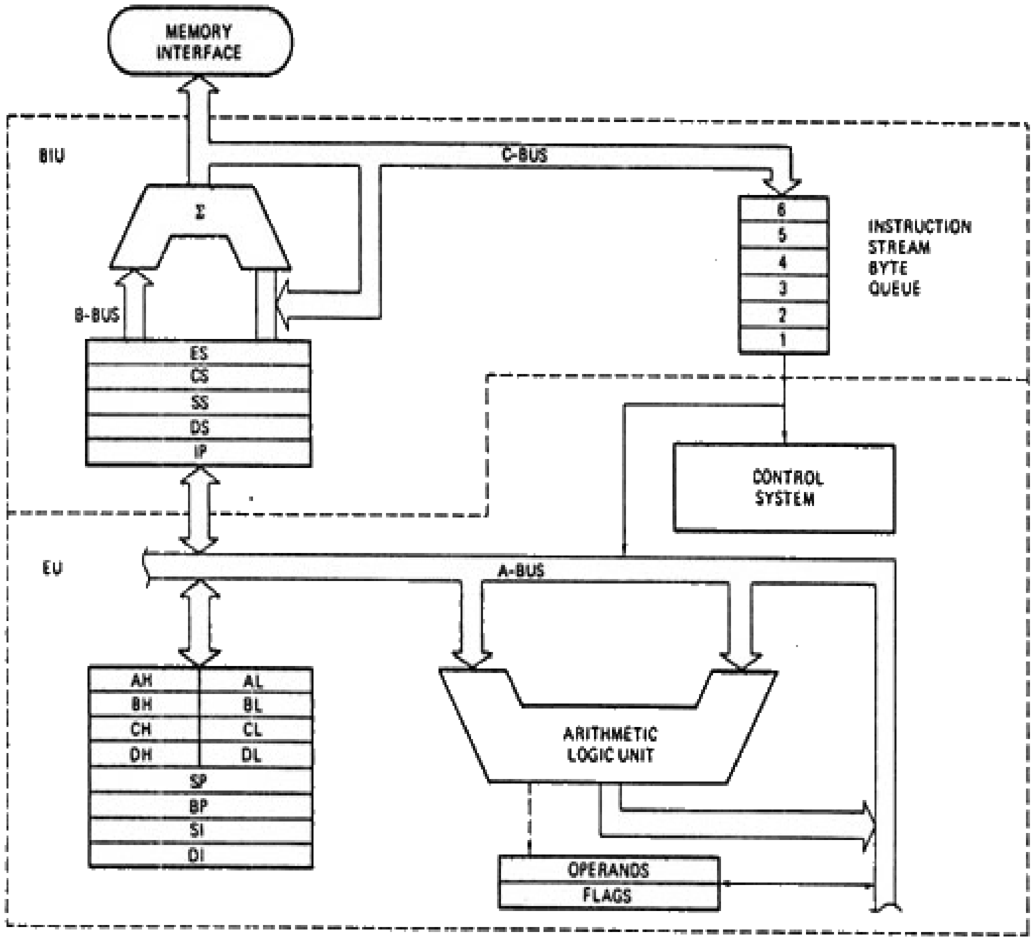
(used only when MN/M$\overline{X}$ = 0)

- **S2̅, S1̅, S0̅ (Pins 26–28)**: Status lines for controlling bus cycles.
- **RQ/GT1̅, RQ/GT0̅ (Pins 34, 35)**: Request/Grant for bus control in multiprocessor mode.
- **LOCK (Pin 29)**: Ensures exclusive access to shared resources.
- **QS1, QS0 (Pins 24, 25)**: Queue status signals.

Intel 8086 architecture

The block diagram of the Intel 8086 architecture, showing the **internal structure** of the microprocessor. The 8086 is divided into two main units:

1. **Bus Interface Unit (BIU)**
2. **Execution Unit (EU)**



🧠 1. Bus Interface Unit (BIU)

Key Responsibilities:

- Handles **address generation**, **instruction fetching**, and **communication with memory and I/O**.
- Feeds the **Execution Unit (EU)** with a stream of instructions.

Components:

- **Segment Registers (ES, CS, SS, DS)**
  These hold segment addresses for Code (CS), Data (DS), Stack (SS), and Extra segment (ES).
- **Instruction Pointer (IP)**
  Points to the next instruction in the Code Segment (CS).
- **Adder (Σ)**
  Combines segment base and offset to form the **physical address**.

- **Instruction Queue**
  Pre-fetches up to **6 bytes** of instructions (using pipelining). This allows the Execution Unit to execute instructions faster.
- **Memory Interface**
  Controls the **communication between memory and processor** using the address and data buses.

⚙️ 2. Execution Unit (EU)

Key Responsibilities:

- **Decodes and executes instructions**
- Performs **arithmetic and logic operations**
- Controls flags and operands

Components:

- **General Purpose Registers**
  Used for data storage and manipulation:
  - AH/AL, BH/BL, CH/CL, DH/DL → Can be used as 8-bit or combined into AX, BX, CX, DX (16-bit)
  - SP (Stack Pointer), BP (Base Pointer), SI (Source Index), DI (Destination Index)
- **Arithmetic and Logic Unit (ALU)**
  Performs all **arithmetic (add, subtract, etc.)** and **logic (AND, OR, NOT, etc.)** operations.
- **Operands and Flags**
  - **Operands**: Data inputs to the ALU.
  - **Flags**: Special-purpose bits reflecting the outcome of ALU operations (e.g., Zero, Carry, Overflow).

🔄 Data Buses:

- **A-Bus**:
  Transfers data between **registers and ALU** in the Execution Unit.
- **C-Bus**:
  Transfers **addresses and instructions** from the BIU to memory and instruction queue.

🧮 Summary:

Component	**Purpose**
BIU	Handles memory interaction, instruction fetching
Segment Registers	Holds memory segment addresses
Instruction Pointer	Points to next instruction
Instruction Queue	Stores pre-fetched instructions (6 bytes max)
EU	Executes instructions using ALU
General Registers	Temporary storage and data manipulation
ALU	Arithmetic and logic processing
Flags	Reflect results of operations
A-Bus, C-Bus	Internal data and control signal pathways

Registers in 8086

General Purpose Registers (8)

Pointer and Index Registers (4)

Segment Registers (4)

Instruction Pointer (IP)

Flag Register (1 total)

A. General Purpose Registers (8)

Used for arithmetic, logic, data transfer operations.

Register	**16-bit**	**8-bit Parts**	**Usage**
AX	AH + AL	AH, AL	Accumulator
BX	BH + BL	BH, BL	Base register
CX	CH + CL	CH, CL	Counter (loops, shifts)
DX	DH + DL	DH, DL	Data register (I/O, MUL/DIV)

B. **Pointer and Index Registers (4)**

Register	**Function**
SP	Stack Pointer
BP	Base Pointer
SI	Source Index (used in string operations)
DI	Destination Index

C. Segment Registers (4)

Used to access different memory segments (20-bit address space)

Register	**Segment**
CS	Code Segment (for IP)
DS	Data Segment
SS	Stack Segment
ES	Extra Segment

🔶 *These segment registers hold the base addresses used to compute the 20-bit physical address.*

D. Instruction Pointer (IP)

- Holds offset of the next instruction in the **code segment (CS)**.

E. Flag Register (1 total)

- 16-bit register with 9 active flags (explained below)

Physical Address Calculation in 8086

8086 uses **Segment: Offset** addressing, combining **a 16-bit segment register** and **a 16-bit offset** to form a **20-bit physical address**.

Physical Address = (Segment × 10h) + Offset

Example: Let's say:

Segment = 1234h   Offset = 0010h

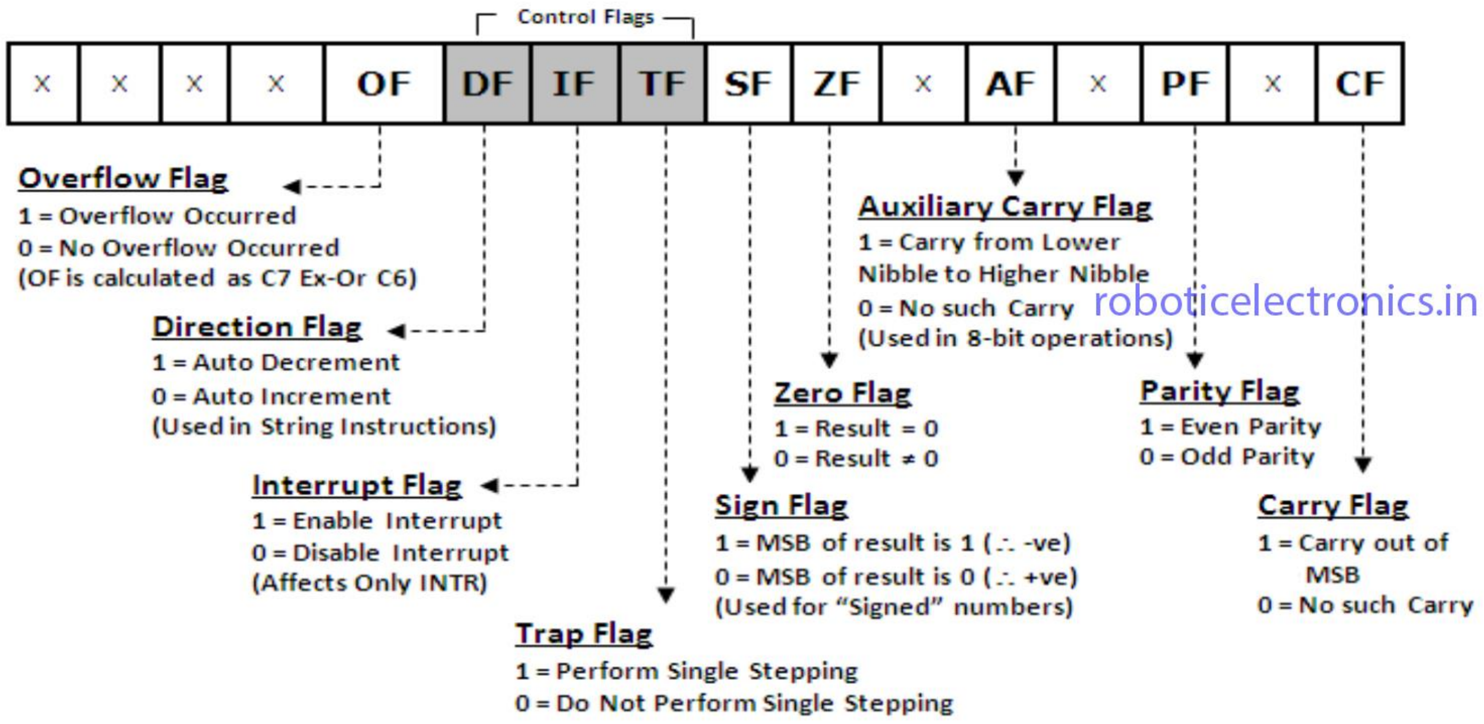
Then,

Physical Address = (1234h × 10h) + 0010h

= 12340h + 0010h

= **12350h**

Flags Register in 8086 (16-bit)



A. Status Flags

Indicate results of operations:

- **CF** – Carry Flag
- **PF** – Parity Flag
- **AF** – Auxiliary Carry Flag
- **ZF** – Zero Flag
- **SF** – Sign Flag
- **OF** – Overflow Flag

B. Control Flags

Control processor operations:

- **TF** – Trap Flag (for debugging)
- **IF** – Interrupt Enable Flag
- **DF** – Direction Flag (string operations)

Addressing Modes in 8086

An **addressing mode** specifies **how an operand (data)** for an instruction is accessed (i.e., whether it is in a register, memory, or part of the instruction itself).

8086 supports **several addressing modes** to allow flexible memory and data access.

1. Immediate Addressing Mode

- The operand is directly specified in the instruction.
- Data is given as a **constant value**.

```
MOV AX, 1234h     ; AX ← 1234h
```

Use: Simple data loading.

2. Register Addressing Mode

- The operand is located in a **register**.
- Both source and destination are registers.

  MOV AX, BX   ; AX ← BX

  Use: Fast data movement within CPU.

3. Direct Addressing Mode

- The **memory address** of the operand is **given explicitly** in the instruction.
- Offset is fixed; segment is usually **DS** by default.

  MOV AX, [1234h]   ; AX ← contents at memory address DS:1234h

  Use: Access fixed memory location.

4. Register Indirect Addressing Mode

- The **effective address** of the operand is stored in a register.
- Registers used: **BX**, **BP**, **SI**, or **DI**.
- Segment: **DS** for BX, SI, DI; **SS** for BP.

  MOV AX, [BX]   ; AX ← contents at DS:BX

  Use: Access variable memory locations dynamically.

5. Based Addressing Mode

- A **base register** holds the offset (BX or BP).
- Optional displacement can be added.
- Segment: **DS** (for BX), **SS** (for BP)

  MOV AX, [BP]   ; AX ← contents at SS:BP

  Use: Access stack or data memory.

6. Indexed Addressing Mode

- Uses index registers: **SI** or **DI**
- Effective address = contents of index register
- Segment default: **DS**

  MOV AX, [SI]   ; AX ← contents at DS:SI

  Use: Access array elements.

7. Based-Indexed Addressing Mode

- Combines base register (BX or BP) and index register (SI or DI).
- Effective address = base + index
- Segment default: **DS** or **SS** depending on base

  MOV AX, [BX + SI] ; AX ← contents at DS:(BX+SI)

  Use: Multidimensional arrays or struct access.

8. Based-Indexed with Displacement

- Most flexible addressing mode.
- Effective address = base + index + displacement
- Allows full offset control.

  MOV AX, [BX + SI + 10] ; AX ← contents at DS:(BX + SI + 10)

  Use: Array element + offset access.

9. Relative Addressing Mode

Used **only with jump (JMP) and branch instructions**.

- The target address is relative to the **current instruction pointer (IP)**.

  JMP SHORT LABEL ; IP ← IP + displacement to LABEL

  Use: Efficient code branching.

Mode	**Syntax Example**	**Effective Address Formula**
Immediate	MOV AX, 1234h	Operand is in instruction
Register	MOV AX, BX	Operand in register
Direct	MOV AX, [1234h]	EA = 1234h
Register Indirect	MOV AX, [BX]	EA = contents of BX
Based	MOV AX, [BP]	EA = BP
Indexed	MOV AX, [SI]	EA = SI
Based + Indexed	MOV AX, [BX + SI]	EA = BX + SI
Based + Indexed + Disp	MOV AX, [BX + SI + 5]	EA = BX + SI + 5
Relative (Jumps only)	JMP SHORT LABEL	New IP = Old IP + displacement

80286: Architecture, Registers, (Real/Protected mode), Privilege levels, descriptor cache, Memory access in GDT and LDT, multitasking, addressing modes, flag register

80386: Architecture, Register organization, Memory access in protected mode, Paging