TOPIC : 8086 Micro-Processor

Subject: Digital Logic Design & Micro-Processor

Name : Aaditya Pravin Kolhapure

Class : S.Y B.Tech

Branch : Computer Science & Engineering

Roll no. : 21212017

Prn No. :2167971242025

Index

Introduction & Features of 8086 micro-processor :

8086 Microprocessor is an enhanced version of 8085Microprocessor 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.

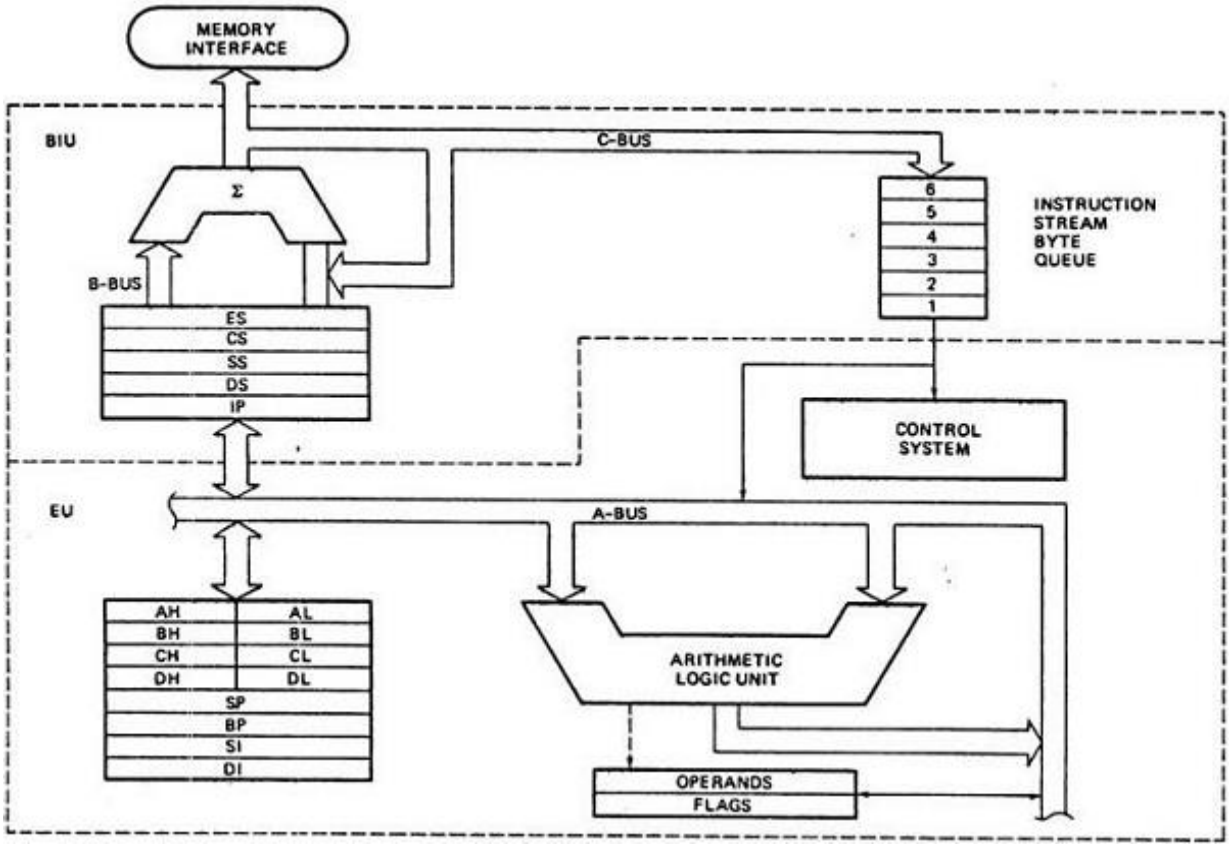
Features of 8086 micro-processor :

The most prominent features of a 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.

Architecture of 8086 micro-processor :

The following diagram depicts the architecture of a 8086 Microprocessor −



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

1. The Bus Interface Unit (BIU):

It provides the interface of 8086 to external memory and I/O devices via the System Bus. It performs various machine cycles such as memory read, I/O read, etc. to transfer data between memory and I/O devices.

BIU performs the following functions are as follows:

- It generates the 20-bit physical address for memory access.
- It fetches instructions from the memory.
- It transfers data to and from the memory and I/O.
- Maintains the 6-byte pre-fetch instruction queue(supports pipelining).

BIU mainly contains the 4 Segment registers, the Instruction Pointer, a pre-fetch queue, and an Address Generation Circuit.

Instruction Pointer (IP):

- It is a *16-bit register*. It holds offset of the next instructions in the *Code Segment.*
- IP is incremented after every instruction byte is fetched.
- IP gets a new value whenever a branch instruction occurs.
- CS is multiplied by 10H to give the 20-bit physical address of the Code Segment.
- The address of the next instruction is calculated by using the formula CS x 10H + IP.

Example:

CS = 4321H IP = 1000H

then CS x 10H = 43210H + offset = *44210H*
Here Offset = Instruction Pointer(IP)
This is the address of the next instruction.

Code Segment register: (16 Bit register): CS holds the base address for the Code Segment. All programs are stored in the Code Segment and accessed via the IP.
Data Segment register: (16 Bit register): DS holds the base address for the Data Segment.
Stack Segment register: (16 Bit register): SS holds the base address for the Stack Segment.
Extra Segment register: (16 Bit register): ES holds the base address for the Extra Segment.
Please note that segments are present in memory and segment registers are present in Microprocessor. Segment registers store starting address of each segments in memory.

Address Generation Circuit:

- The BIU has a Physical Address Generation Circuit.
- It generates the 20-bit physical address using Segment and Offset addresses using the formula:
- In Bus Interface Unit (BIU) the circuit shown by the Σ symbol is responsible for the calculation unit which is used to calculate the physical address of an instruction in memory.

Physical Address = Segment Address x 10H + Offset Address
6 Byte Pre-fetch Queue:

- It is a 6-byte queue (FIFO).
- Fetching the next instruction (by BIU from CS) while executing the current instruction is called pipelining.
- Gets flushed whenever a branch instruction occurs.
- The pre-Fetch queue is of 6-Bytes only because the maximum size of instruction that can have in 8086 is 6 bytes. Hence to cover up all operands and data fields of maximum size instruction in 8086 Microprocessor there is a Pre-Fetch queue is 6 Bytes.
- The pre-Fetch queue is connected with the control unit which is responsible for decoding op-code and operands and telling the execution unit what to do with the help of timing and control signals.
- The pre-Fetch queue is responsible for pipelining and because of that 8086 microprocessor is called fetch, decode, execute type microprocessor. Since there are always instructions present for decoding and execution in this queue the speed of execution in the microprocessor is gradually increased.
- When there is a 2-byte space in the instruction pre-fetch queue then only the next instruction will be pushed into the queue otherwise if only a 1-byte space is vacant then there will not be any allocation in the queue. It will wait for a spacing of 2 bytes in subsequent queue decoding operations.
- Instruction pre-fetch queue works in a sequential manner so if there is any branch condition then in that situation pre-fetch queue fails. Hence to avoid chaos instruction queue is flushed out when any branch or conditional jumps occur.

2.prefetch unit:

The Prefetch Unit in the 8086 microprocessor is a component responsible for fetching instructions from memory and storing them in a queue. The prefetch unit allows the 8086 to perform multiple instruction fetches in parallel, improving the overall performance of the microprocessor.

The prefetch unit is an important component of the 8086 microprocessor, as it allows the microprocessor to work more efficiently and perform more instructions in a given amount of time. This improved performance helps to ensure that the 8086 remains competitive in its performance and capabilities, even as technology continues to advance.

3. The Execution Unit (EU):

The main components of the EU are General purpose registers, the ALU, Special purpose registers, the Instruction Register and Instruction Decoder, and the Flag/Status Register.

1. Fetches instructions from the Queue in BIU, decodes, and executes arithmetic and logic operations using the ALU.
2. Sends control signals for internal data transfer operations within the microprocessor.(Control Unit)
3. Sends request signals to the BIU to access the external module.

Flag in 8086 micro-processor :

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.

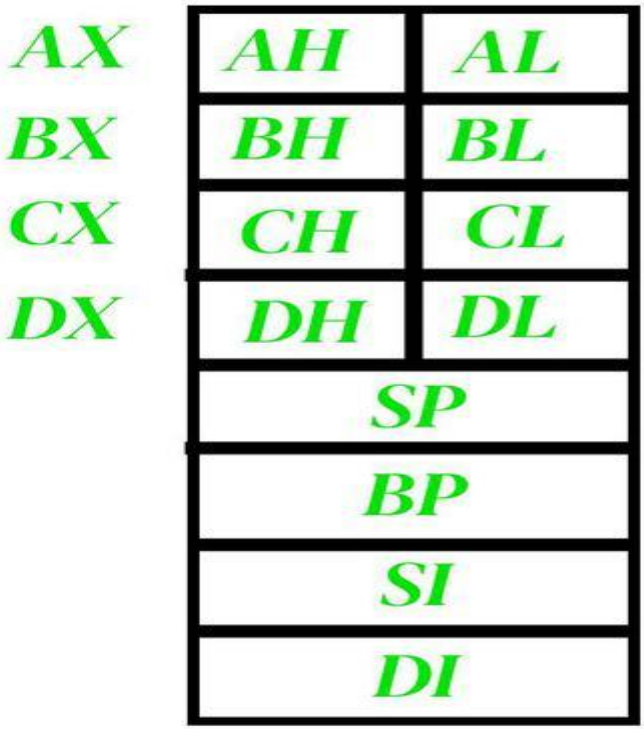
2) Control Flags

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.

General purpose register in 8086 micro-processor :

General-purpose registers are used to store temporary data within the microprocessor. There are 8 general-purpose registers in the 8086 microprocessor.



1. AX: This is the accumulator. It is of 16 bits and is divided into two 8-bit registers AH and AL to also perform 8-bit instructions. It is generally used for arithmetical and logical instructions but in 8086 microprocessor it is not mandatory to have an accumulator as the destination operand. Example:
ADD AX, AX (AX = AX + AX)

2. BX: This is the base register. It is of 16 bits and is divided into two 8-bit registers BH and BL to also perform 8-bit instructions. It is used to store the value of the offset. Example:
MOV BL, [500] (BL = 500H)

3. CX: This is the counter register. It is of 16 bits and is divided into two 8-bit registers CH and CL to also perform 8-bit instructions. It is used in looping and rotation. Example:
MOV CX, 0005

4. DX: This is the data register. It is of 16 bits and is divided into two 8-bit registers DH and DL to also perform 8-bit instructions. It is used in the multiplication and input/output port addressing. Example:
MUL BX (DX, AX = AX * BX)

5. SP: This is the stack pointer. It is of 16 bits. It points to the topmost item of the stack. If the stack is empty the stack pointer will be (FFFE)H. Its offset address is relative to the stack segment.
6. BP – This is the base pointer. It is of 16 bits. It is primarily used in accessing parameters passed by the stack. Its offset address is relative to the stack segment.
7. SI – This is the source index register. It is of 16 bits. It is used in the pointer addressing of data and as a source in some string-related operations. Its offset is relative to the data segment.

8. DI – This is the destination index register. It is of 16 bits. It is used in the pointer addressing of data and as a destination in some string-related operations. Its offset is relative to the extra segment

Instruction in 8086 micro-processor:

- Data Transfer Instructions
- Arithmetic Instructions
- Bit Manipulation Instructions
- String Instructions
- Program Execution Transfer Instructions (Branch & Loop Instructions)
- Processor Control Instructions
- Iteration Control Instructions
- Interrupt Instructions

1. Data 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 −

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.

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.

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.

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 −

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.

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.

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.

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, etc.

Following is the list of instructions under this group –

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.

Instructions to perform shift operations

- SHL − 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.

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].

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.

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 −

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.

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

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

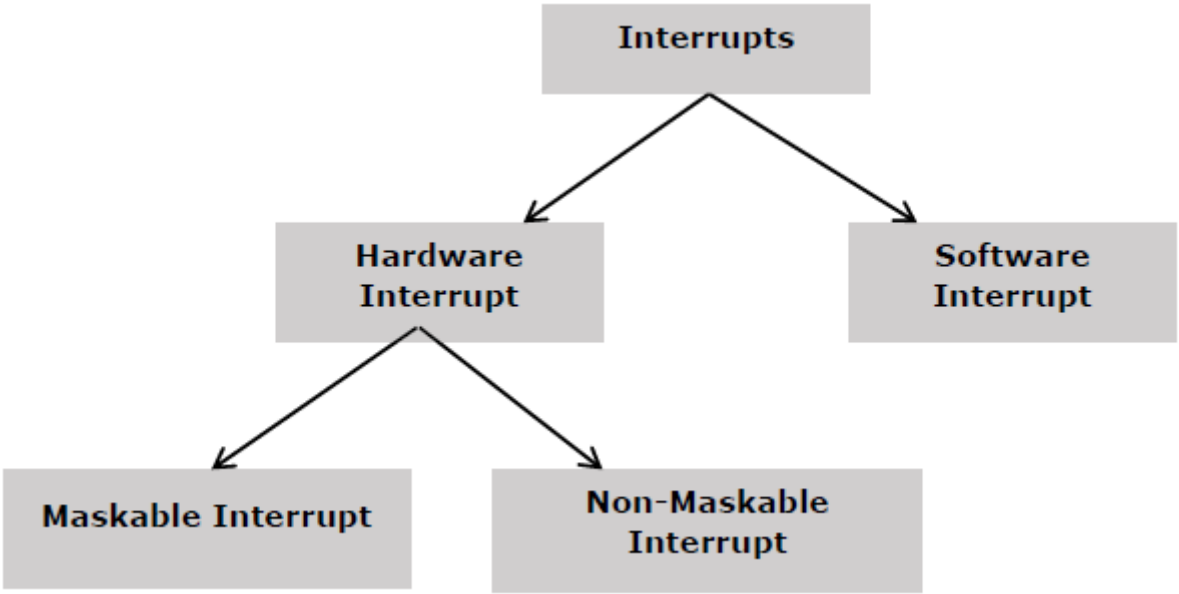
8. Interrupt Instructions

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

Types of interrupt :

The following image shows the types of interrupts we have in a 8086 microprocessor −



1. **Hardware Interrupts**

Hardware interrupt is caused by any peripheral device by sending a signal through a specified pin to the microprocessor.

The 8086 has two hardware interrupt pins, i.e. NMI and INTR. NMI is a non-maskable interrupt and INTR is a maskable interrupt having lower priority. One more interrupt pin associated is INTA called interrupt acknowledge.

NMI :-

It is a single non-maskable interrupt pin (NMI) having higher priority than the maskable interrupt request pin (INTR)and it is of type 2 interrupt.

When this interrupt is activated, these actions take place −

- Completes the current instruction that is in progress.
- Pushes the Flag register values on to the stack.
- Pushes the CS (code segment) value and IP (instruction pointer) value of the return address on to the stack.
- IP is loaded from the contents of the word location 00008H.
- CS is loaded from the contents of the next word location 0000AH.
- Interrupt flag and trap flag are reset to 0.

INTR :-

The INTR is a maskable interrupt because the microprocessor will be interrupted only if interrupts are enabled using set interrupt flag instruction. It should not be enabled using clear interrupt Flag instruction.

The INTR interrupt is activated by an I/O port. If the interrupt is enabled and NMI is disabled, then the microprocessor first completes the current execution and sends ‘0’ on INTA pin twice. The first ‘0’ means INTA informs the external device to get ready and during the second ‘0’ the microprocessor receives the 8 bit, say X, from the programmable interrupt controller.

These actions are taken by the microprocessor −

- First completes the current instruction.
- Activates INTA output and receives the interrupt type, say X.
- Flag register value, CS value of the return address and IP value of the return address are pushed on to the stack.
- IP value is loaded from the contents of word location X × 4
- CS is loaded from the contents of the next word location.
- Interrupt flag and trap flag is reset to 0

2. **Software Interrupts**

Some instructions are inserted at the desired position into the program to create interrupts. These interrupt instructions can be used to test the working of various interrupt handlers. It includes −

INT- Interrupt instruction with type number

It is 2-byte instruction. First byte provides the op-code and the second byte provides the interrupt type number. There are 256 interrupt types under this group.

- TYPE 0 interrupt represents division by zero situation.
- TYPE 1 interrupt represents single-step execution during the debugging of a program.
- TYPE 4 interrupt represents overflow interrupt.

The interrupts from Type 5 to Type 31 are reserved for other advanced microprocessors, and interrupts from 32 to Type 255 are available for hardware and software interrupts.

Addressing mode in 8086 micro-processor:

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]

Pin diagram of 8086 micro-processor:

Here is the pin diagram of 8086 microprocessor −