INTRODUCTION TO INTEL MICROPROCESSOR 8086

Overview of Microcomputer Structure and Operation:

Evolution of Microprocessors:

→ Intel introduced its first 4-bit microprocessor 4004 in 1971 and its 8-bit microprocessor 8008 in 1972. These microprocessors could not survive as general purpose microprocessors due to their design and performance limitations.

→ The first general purpose 8-bit microprocessor 8080 was launched in 1974 and later in 1977 an updated version of 8080, the 8085 microprocessor was introduced with more added features which result in a functionally complete microprocessor.

→ The main limitations of the 8-bit microprocessors were their
  ▪ low speed,
  ▪ low memory addressing capability,
  ▪ limited number of general purpose registers and
  ▪ a less powerful instruction set.

→ In the family of 16-bit microprocessors, Intel’s 8086 was the first one to be launched in 1978.

<table>
<thead>
<tr>
<th></th>
<th>4004</th>
<th>8008</th>
<th>8080</th>
<th>8085</th>
<th>8086</th>
<th>80286</th>
<th>80386</th>
<th>Pentium</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Bits</td>
<td>4</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>Technology</td>
<td>PMOS</td>
<td>PMOS</td>
<td>NMOS</td>
<td>NMOS</td>
<td>NMOS</td>
<td>HMOS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory</td>
<td>4KB</td>
<td>16KB</td>
<td>64KB</td>
<td>64KB</td>
<td>1MB</td>
<td>1MB</td>
<td>1MB</td>
<td></td>
</tr>
<tr>
<td>Instruction Set</td>
<td>45</td>
<td>48</td>
<td>246</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speed</td>
<td>50KIPS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No. of Transistors</td>
<td>2300</td>
<td>3500</td>
<td>4500</td>
<td>6500</td>
<td>29,000</td>
<td>29,000</td>
<td>29,000</td>
<td></td>
</tr>
</tbody>
</table>
**Introduction to 8086 Microprocessor:**

- It is a 40 pin DIP chip based on N-channel, depletion load silicon gate technology (HMOS).
- The term 16-bit means that it supports a 16-bit ALU, its internal registers and most of the instructions are designed to work with 16 bit binary words.
- 8086 is available at different clock speeds Via, 5 MHz (8086); 8MHz (8086-2) and 10MHz (8086-1).
- 8086 microprocessor has a 16-bit data bus and 20-bit address bus. So, it can address any one of \(2^{20} = 1048576 = 1\) Megabyte (1MB) memory locations.
- The 8086 microprocessor can work in two modes of operations. They are Minimum mode and Maximum mode. In the minimum mode of operation the microprocessor do not associate with any co-processors and can not be used for multiprocessor systems. But in the maximum mode the 8086 can work in multi-processor or co-processor configuration. These minimum or maximum operations are decided by the pin MN/ MX (Active low). When this pin is high 8086 operates in minimum mode otherwise it operates in maximum mode.

**Microcontrollers Vs Microprocessors**

A microprocessor requires a external memory for program/data storage. Instruction execution requires movement of data from the external memory to the microprocessor or vice versa. Usually, microprocessors have good computing power and they have higher clock speed to facilitate faster computation.

A microcontroller has required on-chip memory with associated peripherals. A microcontroller can be thought of a microprocessor with inbuilt peripherals. A microcontroller does not require much additional interfacing ICs for operation and it functions as a stand-alone system. The operation of a microcontroller is multipurpose, just like a Swiss knife. Microcontrollers are also called embedded controllers. A microcontroller clock speed is limited only to a few tens of MHz Microcontrollers are numerous and many of them are application specific.

**8086 specifications:**

1. It is 16-bit microprocessor
2. It has 20 bit address bus and can access up to 220 memory locations (1 MB).
3. It can support up to 64K I/O ports
4. It provides 14, 16-bit registers
5. It has multiplexed address and data bus AD0-AD15 & A16-A19
6. It requires single phase clock with 33% duty cycle to provide internal timing.
7. Prefetches up to 6 instruction bytes from memory and queues them in order to speed up the processing.
8. It requires +5V supply
9. 40 pin dual inline package
10. 8086 supports 2 modes of operation
    a. Minimum mode
    b. Maximum mode

- a rich instruction set
- powerful interrupt structure
- Fetched instruction queue for overlapped fetching and execution etc.

→ To improve the performance by implementing the parallel processing concept the CPU of the 8086 is divided into two independent sections.

    They are
    - Bus Interface Unit (BIU) and
    - Execution Unit (EU).

→ The BIU sends out addresses, fetches instructions, read data from ports and memory and writes data to ports and memory. i.e the BIU handles all transfers of data and addresses on the buses required by the Execution Unit whereas the Execution Unit tells the BIU where to fetch instructions or data from, decodes the instructions and executes the instructions.

→ The BIU contains
  - the circuit for physical address calculations
  - a pre-decoding instruction byte queue (6 bytes long)
  - four 16-bit segment registers (ES, CS, SS, DS)
  - 16-bit instruction pointer (IP)

→ The EU contains
  - control circuitry, instruction decoder and ALU
  - 16-bit flag registers
  - four 16-bit general purpose registers (AX, BX, CX, DX)
  - 16-bit pointer registers (SP, BP) and
  - 16-bit index registers (SI, DI)
Execution Unit:

Control Circuitry, Instruction Decoder and ALU:

→ The EU contains control circuitry which directs internal operations.

→ A decoder in the EU translates instructions fetched from memory into a series of actions which the EU carries out.

→ The EU has a 16-bit arithmetic logic unit which can add, subtract, AND, OR, XOR, increment, decrement, complement, or shift binary numbers.

Flag Registers:

→ The 8086 16-bit flag register contents indicate the results of computations in the ALU. It also contains some flag bits to control the CPU operations.

→ A flag is a flip-flop that indicates some condition produced by the execution of an instruction or controls certain operations of the EU.

General Purpose Registers:

General Registers: All general registers of the 8086 microprocessor can be used for arithmetic and logic operations. The general registers are:
**AX (Accumulator):** This is accumulator register. It gets used in arithmetic, logic and data transfer instructions. In manipulation and division, one of the numbers involved must be in AX or AL.

**BX (Base Register):** This is base register. BX register is an address register. It usually contains a data pointer used for based, based indexed or register indirect addressing.

**CX (Count register):** This is Count register. This serves as a loop counter. Program loop constructions are facilitated by it. Count register can also be used as a counter in string manipulation and shift/rotate instruction.

**DX (Data Register):** This is data register. Data register can be used as a port number in I/O operations. It is also used in multiplication and division.

**SP (Stack Pointer):** This is stack pointer register pointing to program stack. It is used in conjunction with SS for accessing the stack segment.

**BP (Base Pointer):** This is base pointer register pointing to data in stack segment. Unlike SP, we can use BP to access data in the other segments.

**SI (Source Index):** This is source index register which is used to point to memory locations in the data segment addressed by DS. By incrementing the contents of SI one can easily access consecutive memory locations.

**DI (Destination Index):** This is destination index register performs the same function as SI. There is a class of instructions called string operations, that use DI to access the memory locations addressed by ES.

**ALU (Arithmetic & Logic Unit):** This unit can perform various arithmetic and logical operation, if required, based on the instruction to be executed. It can perform arithmetical operations, such as add, subtract, increment, decrement, convert byte/word and compare etc and logical operations, such as AND, OR, exclusive OR, shift/rotate and test etc.
Addition of IP to CS to produce physical address of code byte
a) Diagram (b) computation.

→Advantages of Memory segmentation:
Addition of SS and SP to produce physical address of top of stack.

(a) Diagram (b) Computation
SEGMENTATION - The memory Addressing Scheme For 8086:
Address bus size=20 bit
Total addressable locations=220 =1MB
Total physical address=1MB
By using segmentation, 1MB divided into 16 segments of each segment size 64Kb.
1. Physical address of 8086 is 20 bit wide. So it can access 1 MB memory (220*8=1 MB or 16*64 KB). This 1 MB memory is divided into 16 Segment memories. The capacity of each memory segment is 64 KB. But 8086 can access at a time only memory segment. They are CS memory, DS memory, SS memory and ES memory.
2. Instruction fetch operations are performed in DS memory. String operations are performed in ES memory.
3. For the selection of each segment memory, 8086 has 4 segment registers. They are known as CS Register, DS Register, SS Register, and SS Register. The content of each segment register is known as the Base Register.
4. BIU generates 20-bit physical Address by using segment Address and offset Address.

Physical address of next instruction = **segment address** (given by segment registers) + **Offset address** (given by either pointers or index or base registers)

<table>
<thead>
<tr>
<th>S.No</th>
<th>Type</th>
<th>Register width</th>
<th>Name of the Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>General purpose Registers(2)</td>
<td>16-bit</td>
<td>AX,BX,CX,DX</td>
</tr>
<tr>
<td>2</td>
<td>General purpose Registers(2)</td>
<td>8-bit</td>
<td>AL,AH,BL,BH,CL,CH,DH</td>
</tr>
<tr>
<td>2</td>
<td>Pointer Registers</td>
<td>16-bit</td>
<td>Stack Pointer(SP)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Base Pointer(BP)</td>
</tr>
<tr>
<td>3</td>
<td>Index Registers</td>
<td>16-bit</td>
<td>Source Index(SI)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Destination Index(DI)</td>
</tr>
<tr>
<td>4</td>
<td>Segment Registers</td>
<td>16-bit</td>
<td>Code Segment(CS)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Data Segment(DS)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Stack Segment(SS)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Extra Segment(ES)</td>
</tr>
<tr>
<td>5</td>
<td>Instruction</td>
<td>16-bit</td>
<td>Instruction Pointer (IP)</td>
</tr>
<tr>
<td>6</td>
<td>Flag (PSW)</td>
<td>16-bit</td>
<td>Flag Register</td>
</tr>
</tbody>
</table>

**8086 Microprocessor Registers.**
The register set of 8086 can be categorized into 4 different groups. The register organization of 8086 is shown in the figure.

<table>
<thead>
<tr>
<th>AX</th>
<th>AH</th>
<th>AL</th>
<th>CS</th>
<th>SP</th>
<th>BP</th>
</tr>
</thead>
<tbody>
<tr>
<td>BX</td>
<td>BH</td>
<td>BL</td>
<td>SS</td>
<td>SI</td>
<td>DI</td>
</tr>
<tr>
<td>CX</td>
<td>CH</td>
<td>CL</td>
<td>DS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DX</td>
<td>DH</td>
<td>DL</td>
<td>ES</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

General data registers | Segment Registers | Pointers and index registers

Register organisation of 8086

**REGISTER ORGANISATION:**

The 14 registers of 8086 microprocessor are categorized into four groups. They are general purpose data registers, Pointer & Index registers, Segment registers and Flag register as shown in the table below.

**8086 Microprocessor Registers.**

<table>
<thead>
<tr>
<th>S.No</th>
<th>Type</th>
<th>Register width</th>
<th>Name of the Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>General purpose Registers(2)</td>
<td>16-bit</td>
<td>AX,BX,CX,DX</td>
</tr>
<tr>
<td>2.</td>
<td>General purpose Registers(2)</td>
<td>8-bit</td>
<td>AL,AH,BL,BH,CL,CH,DL,DH</td>
</tr>
<tr>
<td>2</td>
<td>Pointer Registers</td>
<td>16-bit</td>
<td>Stack Pointer(SP)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Base Pointer(BP)</td>
</tr>
<tr>
<td>3</td>
<td>Index Registers</td>
<td>16-bit</td>
<td>Source Index(SI)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Destination Index(DI)</td>
</tr>
<tr>
<td>4</td>
<td>Segment Registers</td>
<td>16-bit</td>
<td>Code Segment(CS)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Data Segment(DS)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Stack Segment(SS)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Extra Segment(ES)</td>
</tr>
</tbody>
</table>
### General purpose registers:
There are four 16-bit general purpose registers namely (AH, AL); (BH, BL); (CH, CL); (and DH, DL) which are part of the Execution unit. These registers can be used individually for storing 16-bit data temporarily. The AL register is also called the accumulator. The pairs of registers can be used together to store 16-bit data words. It is always advantageous to store the data in these registers because the data can be accessed much more easily as these registers are already in the execution unit. Here L indicates the lower byte and H indicates the higher byte. X indicates the extended register. The general purpose data registers are used for data manipulations. The use of these registers is more dependent on the mode of addressing also.

The other four registers of EU are referred to as index / pointer registers. They are Stack Pointer register, Base Pointer register, Source Index register and Destination Index registers. The pointer registers contain the offset within a particular segment.
The BP & SP registers hold the offsets within the data and stack segments respectively. The Index registers are used as general purpose registers as well as for holding the offset in case of indexed based and relative indexed addressing modes. The source Index register is generally used to store the offset of source data in data segment while the Destination Index register used to store the offset of destination in data or extra segment. These index registers are specifically used in string manipulations.

**Segment Registers**: There are four 16-bit segment registers namely code segment register (CS), Stack segment register (SS), Data segment register (DS) and Extra segment register (ES). The code segment register is used for addressing the 64kB memory location in the code segment of the memory, where the code of the executable program is stored. Similarly the DS register points to the data segment of the 64kB memory where the data is stored. The Extra segment register also refers to essentially another data segment of the memory space. The SS register is useful for addressing stack segment of memory. So, the CS, DS, SS and ES segment registers respectively contains the segment addresses for the code, data, stack and extra segments of the memory.

**Instruction Pointer Register**: It is a 16-bit register which always points to the next instruction to be executed within the currently executing code segment. So, this register contains the 16-bit offset address pointing to the next instruction code within the 64kB of the code segment area. Its content is automatically incremented as the execution of the next instruction takes place.

**Flag Register**: This register is also called status register. It is a 16-bit register which contains six status flags and three control flags. So, only nine bits of the 16-bit register are defined and the remaining seven bits are undefined. Normally this status flag bits indicate the status of the ALU after the arithmetic or logical operations. Each bit of the status register is a flip/flop. The Flag register contains Carry flag, Parity flag, Auxiliary flag Zero flag, Sign flag, Trap flag, Interrupt...
flag, Direction flag and overflow flag as shown in the diagram. The CF, PF, AF, ZF, SF, OF are the status flags and the TF, IF and CF are the control flags.

<table>
<thead>
<tr>
<th>X</th>
<th>X</th>
<th>X</th>
<th>X</th>
<th>OF</th>
<th>DF</th>
<th>IF</th>
<th>TF</th>
<th>SF</th>
<th>ZF</th>
<th>X</th>
<th>AF</th>
<th>X</th>
<th>PF</th>
<th>X</th>
<th>CF</th>
</tr>
</thead>
</table>

Flag Register

**CF** - Carry Flag: This flag is set, when there is a carry out of MSB in case of addition or a borrow in case of subtraction.

**PF** - Parity Flag: This flag is set to 1, if the lower byte of the result contains even number of 1’s else (for odd number of 1s) set to zero.

**AF** - Auxiliary Carry Flag: This is set, if there is a carry from the lowest nibble, i.e, bit three during addition, or borrow for the lowest nibble, i.e, bit three, during subtraction.

**ZF** - Zero Flag: This flag is set, if the result of the computation or comparison performed by the previous instruction is zero.

**SF** - Sign Flag: This flag is set, when the result of any computation is negative

**TF** - Tarp Flag: If this flag is set, the processor enters the single step execution mode.

**IF** - Interrupt Flag: If this flag is set, the mask able interrupt INTR of 8086 is enabled and if it is zero, the interrupt is disabled. It can be set by using the STI instruction and can be cleared by executing CLI instruction.

**DF** - Direction Flag: This is used by string manipulation instructions. If this flag bit is ‘0’, the string is processed beginning from the lowest address to the highest address, i.e., auto incrementing mode. Otherwise, the string is processed from the highest address towards the lowest address, i.e., auto incrementing mode.
OF- Over flow Flag: This flag is set, if an overflow occurs, i.e., if the result of a signed operation is large enough to accommodate in a destination register. The result is of more than 7-bits in size in case of 8-bit signed operation and more than 15-bits in size in case of 16-bit sign operations, and then the overflow will be set.

ADDRESSING MODES:

The different ways in which a source operand is denoted in an instruction are known as the addressing modes. There are 8 different addressing modes in 8086 programming. They are

1. Immediate addressing mode
2. Register addressing mode
3. Direct addressing mode
4. Register indirect addressing mode
5. Based addressing mode
6. Indexed addressing mode.
7. Based indexed addressing mode
8. Based, Indexed with displacement.

Immediate addressing mode: The addressing mode in which the data operand is a part of the instruction itself is called Immediate addressing mode.

For Ex: MOV CX, 4847 H
       ADD AX, 2456 H
       MOV AL, FFH

Register addressing mode: Register addressing mode means, a register is the source of an operand for an instruction.

For Ex: MOV AX, BX copies the contents of the 16-bit BX register into the 16-bit AX register.
**EX: ADD CX, DX**

**Direct addressing mode:** The addressing mode in which the effective address of the memory location at which the data operand is stored is given in the instruction. i.e. the effective address is just a 16-bit number is written directly in the instruction.

For Ex: MOV BX, [1354H]
MOV BL,[0400H]

The square brackets around the 1354H denote the contents of the memory location. When executed, this instruction will copy the contents of the memory location into BX register. This addressing mode is called direct because the displacement of the operand from the segment base is specified directly in the instruction.

**Register indirect addressing mode:** Register indirect addressing allows data to be addressed at any memory location through an offset address held in any of the following registers: BP, BX, DI and SI.

Ex: MOV AX, [BX]. Suppose the register BX contains 4675H, the contents of the 4675H are moved to AX.
ADD CX, [BX]

**Based addressing mode:** The offset address of the operand is given by the sum of contents of the BX (or) BP registers and an 8-bit or 16-bit displacement.
Ex: MOV DX, [BX+04]
ADD CL, [BX+08]

**Indexed Addressing mode:** The operands offset address is found by adding the contents of SI or DI register and 8-bit or 16-bit displacements.
Ex: MOV BX, [SI+06]
ADD AL, [DI+08]

**Based-index addressing mode:** The offset address of the operand is computed by summing the base register to the contents of an Index register.
Ex: ADD CX,[BX+SI]

MOV AX,[BX+DI]

**Based Indexed with displacement mode:** The operands offset is computed by adding the base register contents, an Index registers contents and 8 or 16-bit displacement.
Ex: MOV AX, [BX+DI+08]
ADD CX, [BX+SI+16]

---

**8086 PIN DIAGRAM – PIN DESCRIPTION**

Intel 8086 is a 16-bit HMOS microprocessor. It is available in 40 pin DIP chip. It uses a 5V d.c. supply for its operation. The 8086 uses 20-line- address bus. It uses a 16-line data bus. The 20 lines of the address bus operate in multiplexed mode. The 16-low order address bus lines are multiplexed with data and 4 high-order address bus lines are multiplexed with status signals. The pin diagram of Intel 8086 is shown in Fig.4.

**AD<sub>0</sub>-AD<sub>15** (Bidirectional): Address/Data bus. These are low order address bus. They are multiplexed with data. When AD lines are used to transmit memory address the symbol A is used instead of AD, for example A<sub>0</sub>-A<sub>15</sub>. When data are transmitted over AD lines the symbol D is used in place of AD, for example D<sub>0</sub>-D<sub>7</sub>, D<sub>8</sub>-D<sub>15</sub> or D<sub>0</sub>-D<sub>15</sub>.

**A<sub>16</sub>-A<sub>19** (Output): High order addresses bus. These are multiplexed with status signals.
A_{16}/S_3, A_{17}/S_4, A_{18}/S_5, A_{19}/S_6: The specified address lines are multiplexed with corresponding status signals.

**BHE (Active Low)/S_7 (Output):** Bus High Enable/Status. During T1 it is low. It is used to enable data onto the most significant half of data bus, D8-D15. 8-bit device connected to upper half of the data bus use BHE (Active Low) signal. It is multiplexed with status signal S7. S7 signal is available during T2, T3 and T4.

**RD (Read) (Active Low):** The signal is used for read operation. It is an output signal. It is active when low.
**READY:** This is the acknowledgement from the slow device or memory that they have completed the data transfer. The signal made available by the devices is synchronized by the 8284A clock generator to provide ready input to the 8086. The signal is active high.

**INTR-Interrupt Request:** This is a triggered input. This is sampled during the last clock cycles of each instruction to determine the availability of the request. If any interrupt request is pending, the processor enters the interrupt acknowledge cycle. This can be internally masked by resulting the interrupt enable flag. This signal is active high and internally synchronized.

**NMI (Input) –NON-MASKABLE INTERRUPT:** It is an edge triggered input which causes a type 2 interrupt. A subroutine is vectored to via an interrupt vector lookup table located in system memory. NMI is not maskable internally by software. A transition from LOW to HIGH initiates the interrupt at the end of the current instruction. This input is internally synchronized.

**INTA:** INTA: Interrupt acknowledges. It is active LOW during T₂, T₃ and T₆ of each interrupt acknowledge cycle.

**MN/MX MINIMUM / MAXIMUM:** This pin signal indicates what mode the processor is to operate in.

**RQ/GT RQ/GT0:** REQUEST/GRANT: These pins are used by other local bus masters to force the processor to release the local bus at the end of the processor's current bus cycle. Each pin is bidirectional with RQ/GT having higher priority than RQ/GT₁.

**LOCK:** It’s an active low pin. It indicates that other system bus masters are not to allowed to gain control of the system bus while LOCK is active LOW. The LOCK signal remains active until the completion of the next instruction.

**TEST:** This input is examined by a ‘WAIT’ instruction. If the TEST pin goes low, execution will continue, else the processor remains in an idle state. The input is synchronized internally during each clock cycle on leading edge of clock.

**CLK- Clock Input:** The clock input provides the basic timing for processor operation and bus control activity. It’s an asymmetric square wave with 33% duty cycle.

**RESET (Input) :** RESET: causes the processor to immediately terminate its present activity. The signal must be active HIGH for at least four clock cycles.
Vcc – Power Supply (+5V D.C.)

GND – Ground

QS₁, QS₀ (Queue Status) These signals indicate the status of the internal 8086 instruction queue according to the table shown below

<table>
<thead>
<tr>
<th>QS₁</th>
<th>QS₀</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (LOW)</td>
<td>0</td>
<td>No Operation</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>First Byte of Op Code from Queue</td>
</tr>
<tr>
<td>1 (HIGH)</td>
<td>0</td>
<td>Empty the Queue</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Subsequent Byte from Queue</td>
</tr>
</tbody>
</table>

DT/R : DATA TRANSMIT/RECEIVE: This pin is needed in minimum system that desires to use an 8286/8287 data bus transceiver. It is used to control the direction of data flow through the transceiver.

DEN: DATA ENABLE. This pin is provided as an output enable for the 8286/8287 in a minimum system which uses the transceiver. DEN is active LOW during each memory and I/O access and for INTA cycles.

HOLD/HOLDA: HOLD indicates that another master is requesting a local bus. This is an active HIGH. The processor receiving the "hold" request will issue HLDA (HIGH) as an acknowledgement in the middle of a T₄ or T₃ clock cycle.
Minimum Mode 8086 System and Timings:

→ In a minimum mode 8086 system, the microprocessor 8086 is operated in minimum mode by connecting its MN/MX pin to logic 1.

→ In this mode, all the control signals are given out by the microprocessor chip itself.

→ There is a single microprocessor in the single mode system.

→ The remaining components in the system are latches, transceivers, clock generator, memory and I/O devices, chip selection logic for selecting memory or I/O devices.

→ The latches are generally buffered output D-type flip-flops like 74LS373.

→ They are used for separating the valid address from the multiplexed address/data signals and are controlled by the ALE signal generated by 8086.

→ Transceivers are the bi-directional buffers and are sometimes called data amplifiers.

→ They are required to separate the valid data from the time multiplexed address/data signals.

→ They are controlled by two signals namely DEN and DT/R.

→ The DEN signal indicates the valid data available on the data bus while DT/R indicates the direction of data, i.e., from / to the processor.

→ The system contains memory (RAM or ROM), I/O devices for the communication with the processor.

→ The clock generator (IC8284) generates the clock from the crystal oscillator and is used as an accurate timing reference for the system.

→ The clock generator also synchronizes some external signals with the system clock.

→ Since it has 20 address lines and 16 data lines, the 8086 CPU requires three octal address latches and two octal data buffers for the complete address and data separation.

→ The working of the minimum mode configuration system can be better described in terms of the timing diagrams.

→ The timing diagram can be categorized into two parts. The first is the timing diagram for read cycle and second is the write cycle.

→ The read cycle begins in T1 with the assertion of the Address Latch Enable(ALE) signal and M/IO signal.
During the negative going edge of this signal, the valid address is latched on the local bus.

The $BHE$ and $A_o$ signals address low, high or both bytes.

From $T_1$ to $T_4$, the $M / I\!O$ signal indicates a memory or I/O operation.

At $T_2$, the address is removed from the local bus and is sent to the output. The bus is then tristated.

The Read ($RD$) control signal is also activated in $T_2$.

The read ($RD$) signal causes the addressed device to enable its data bus drivers.

After $RD^*$ goes low, the valid data is available on the data bus.

The addressed device will drive the READY line high, when the processor returns the read signal to high level, the addressed device will again tri-state its bus drivers.