
# KTU NOTES

The learning companion.

**KTU STUDY MATERIALS | SYLLABUS | LIVE NOTIFICATIONS | SOLVED QUESTION PAPERS**

**Definition of Microprocessor:**

Microprocessor (µP) is a multipurpose, programmable device that accepts digital data as input, processes it according to instructions stored in its memory, and provides results as output.

or

A microprocessor is a multipurpose, programmable, clock-driven, register-based electronic device that reads binary instructions from a storage device called memory, accepts binary data as input, processes data according to instructions, and provides results as output.

**MICROCONTROLLER:**

A **microcontroller** (sometimes abbreviated µC, uC or MCU) is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals.

or

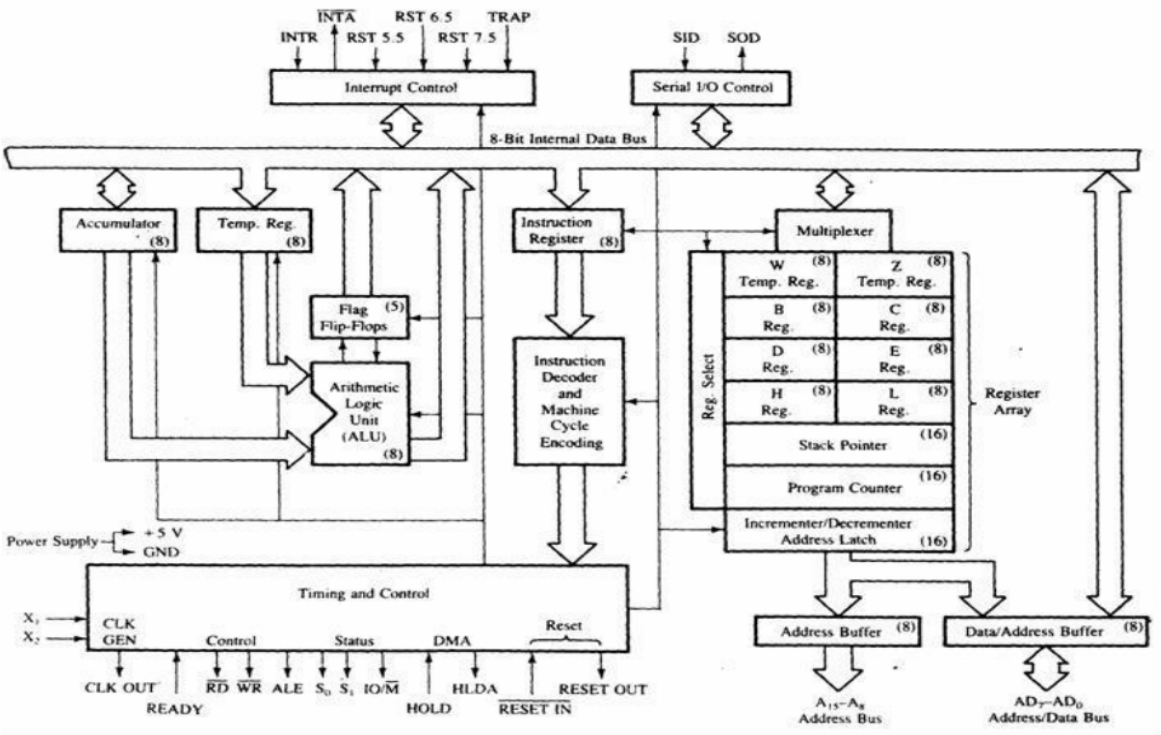
CPUs with integrated memory or peripheral interfaces

# 8085 MICROPROCESSOR ARCHITECTURE

## FEATURES OF 8085

- 8-bit general purpose µp
- Capable of addressing 64 k byte of memory (16 bit address => 2^16 = 64KB)
- Has 40 pins
- Requires +5 v power supply
- Can operate with 3 MHz clock
- 8085 upward compatible






## THE ALU

• In addition to the arithmetic & logic circuits, the ALU includes the accumulator, which is part of every arithmetic & logic operation.
• Also, the ALU includes a temporary register used for holding data temporarily during the execution of the operation. This temporary register is not accessible by the programmer.

## REGISTERS

### GENERAL PURPOSE REGISTERS

• B, C, D, E, H & L (8 bit registers)
• Can be used singly
• Or can be used as 16 bit register pairs BC, DE& HL
• HL used as a data pointer (holds memory address)

### ACCUMULATOR (8 BIT REGISTER)

• Store 8 bit data
• Store the result of an operation
• Store 8 bit data during I/O transfer Address

### FLAG REGISTER

8 bit register – shows the status of the microprocessor before/after an operation
.S (sign flag), Z (zero flag), AC (auxillary carry flag), P (parity flag) & CY (carry flag)

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| S | Z | X | AC | X | P | X | CY |



**SIGN FLAG**
• Used for indicating the sign of the data in the accumulator
• The sign flag is set if negative (1 – negative)
• The sign flag is reset if positive (0 –positive)

**ZERO FLAG**
• Is set if the result obtained after an operation is 0
• Is set following an increment or decrement operation of that register

**CARRY FLAG**
• Is set if there is a carry or borrow from the arithmetic operation

**AUXILLARY CARRY FLAG**
• Is set if there is a carry out of bit 3

**PARITY FLAG**
• Is set if parity is even
• Is cleared if parity is odd

### THE PROGRAM COUNTER (PC)

• This is a register that is used to control the sequencing of the execution of instructions.
• This register always holds the address of the next instruction.
• Since it holds an address, it must be 16 bits wide.

### THE STACK POINTER

• The stack pointer is also a 16-bit register that is used to point into the memory
• The memory this register points to is a special area called the stack. The stack is an area of memory used to hold data that will be retrieved soon.

• The stack is usually accessed in a Last in First out (LIFO) fashion.

**NON PROGRAMMABLE REGISTERS**

Instruction Register & Decoder

• Instruction is stored in IR after being fetched by the processor
• Decoder decodes instruction in IR

**INTERNAL CLOCK GENERATOR**

• 3.125 MHz internally
• 6.25 MHz externally

**THE ADDRESS AND DATA BUSES**

• The address bus has 8 signal lines A8 – A15, which are unidirectional.
• The other 8 address bits are multiplexed (time shared) with the 8 data bits.
• So, the bits AD0 – AD7 are bi-directional and serve as A0 – A7 and D0 – D7 simultaneously.
• During the execution of the instruction, these lines carry the address bits during the early part, then during the late parts of the execution, they carry the 8 data bits.
• In order to separate the address from the data, we can use a latch to save the value before the function of the bits changes.

**HARDWARE INTERRUPTS OF 8085**

These are the interrupts provided as signals to the microprocessor. There are five interrupt signals in 8085. They are Trap, RST 7.5, RST 6.5, RST 5.5, and INTR. The priority of the interrupts is from TRAP to INTR.



INTR - Interrupt Request - general purpose interrupt
INTA - Interrupt acknowledge

**DMA** stands for Direct Memory Access. It is designed by Intel to transfer data at the fastest rate. It allows the device to transfer the data directly to/from memory without any interference from the CPU.

**HOLD and HLDA**

HOLD − This signal indicates that another master is requesting the use of the address and data buses. HLDA (HOLD Acknowledge) − It indicates that the CPU has received the HOLD request and it will relinquish the bus in the next clock cycle.

**RESET IN**

The CPU is held in the reset condition as long as Reset is applied.
Reset sets the Program Counter to zero

**RESET OUT**

Indicates CPU is being reset and also used as a system RESET. The signal is synchronized to the processor clock.

**X1, X2**

For internal clock

**CLK**

Clock Output for use as a system clock

**ALE** (**Address Enable Latch**) is the control signal which goes high means ALE=1, it makes address bus enable and when ALE=0, means low pulse makes data bus enable.

**SID** Serial input data line

**SOD** Serial output data line.

**IO/M**
IO/M indicates whether the Read/Write is to memory or l/O

**S0, S1**
These status signals indicate the operation being done by the microprocessor.
I/O read, I/O write or Interrupt acknowledge

| *IO/M** | *S1* | *S0* | *Machine cycle* |
|---|---|---|---|
| 0 | 0 | 1 | Memory write (MW) |
| 0 | 1 | 0 | Memory read (MR)* |
| 0 | 1 | 1 | Opcode fetch (OF) |
| 1 | 0 | 1 | I/O write (IOW) |
| 1 | 1 | 0 | I/O read (IOR) |
| 1 | 1 | 1 | Interrupt acknowledge (INA)** |



**RD** - READ: indicates the selected memory or 1/0 device is to be read and that the Data Bus is available for the data transfer.

**WR** - WRITE: Indicates the data on the Data Bus is to be written into the selected memory or 1/0 location.

**READY** If Ready is high it indicates that the memory or peripheral is ready to send or receive data.

# INTRODUCTION TO 8086 MICROPROCESSOR

**8086 Microprocessor features:**

1. It is a 16-bit microprocessor
2. It has a 16-bit data bus, so it can read data from or write data to memory and ports either 16-bit or 8-bit at a time.
3. It has 20-bit address bus and can access up to $2^{20}$ memory locations (1 MB).
4. It can support up to 64K I/O ports
5. It provides 14, 16-bit registers
6. It has multiplexed address and data bus $AD_0$-$AD_{15}$ & $A_{16}$-$A_{19}$
7. Prefetches up to 6 instruction bytes from memory and queues them in order to speed up the

processing.

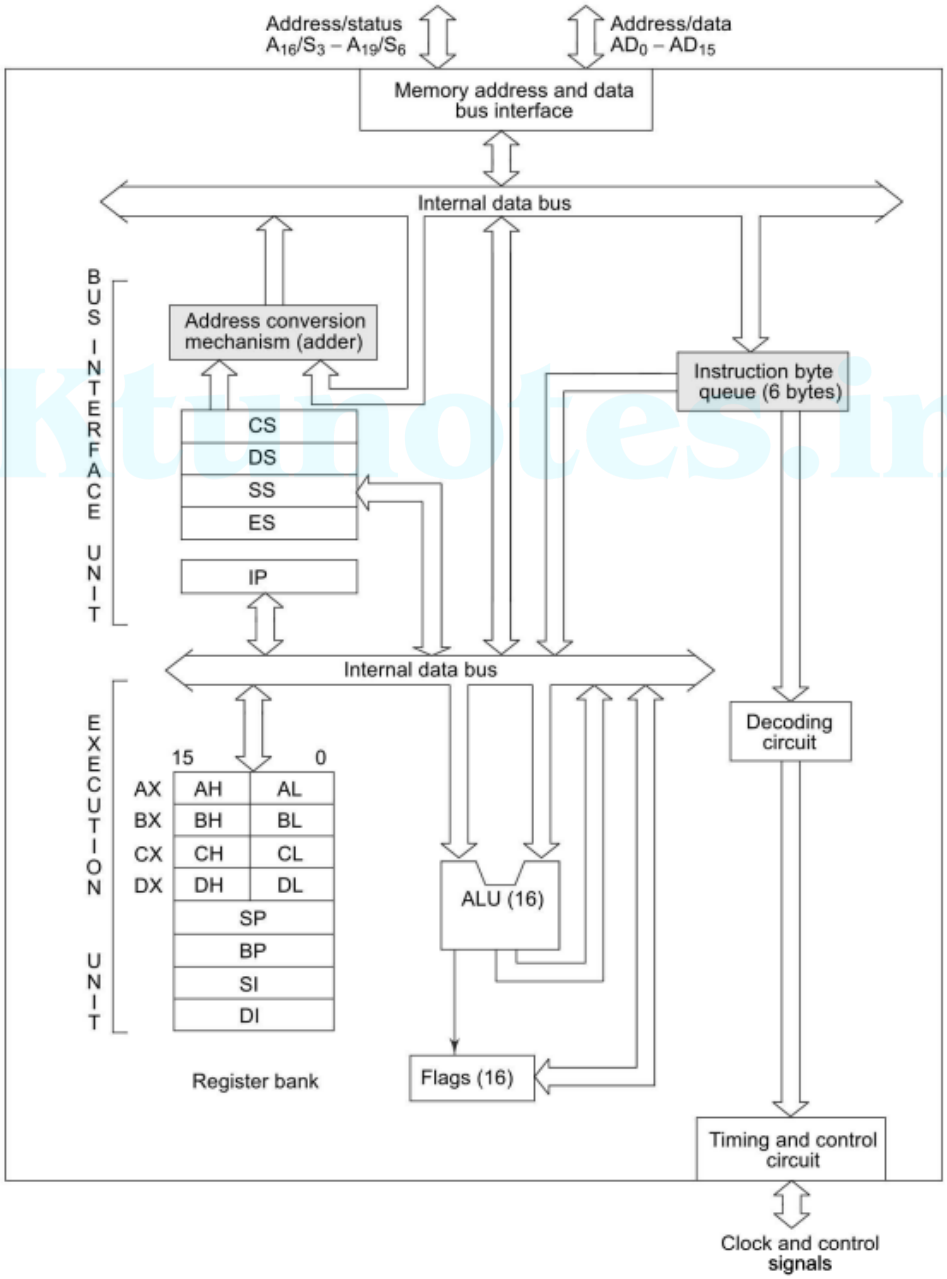
8. 8086 supports 2 modes of operation
    a. Minimum mode
    b. Maximum mode

## ARCHITECTURE OF 8086 MICROPROCESSOR:

As shown in the below figure, the 8086 CPU is divided into two independent functional parts

- Bus Interface Unit(BIU)
- Execution Unit(EU)

Dividing the work between these two units speeds up processing.




*8086 Architecture*

**The Execution Unit (EU):**

The execution unit of the 8086 tells the BIU where to fetch instructions or data from, decodes instructions, and executes instructions.

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** (ALU) that can add, subtract, AND, OR, XOR, increment, decrement, complement or shift binary numbers.
The main functions of the EU are:

- Decoding of Instructions
- Execution of instructions
  - ✔ Steps
    - EU extracts instructions from the top of the queue in BIU
    - Decode the instructions
    - Generates operands if necessary
    - Passes operands to BIU & requests it to perform read or write bus cycles to memory or I/O
    - Perform the operation specified by the instruction on operands

**Bus Interface Unit (BIU):**

The BIU sends out addresses, fetches instructions from memory, reads data from ports and memory, and writes data to ports and memory.

In simple words, the BIU handles all transfers of data and addresses on the buses for the execution unit.



**8086 HAS PIPELINING ARCHITECTURE:**

While the EU is decoding an instruction or executing an instruction, which does not require use of the buses, the BIU fetches up to six instruction bytes for the following instructions.
The BIU stores these **pre-fetched bytes in a first-in-first-out register set called a *queue***.
When the EU is ready for its next instruction from the queue in the BIU. This is much faster than sending out an address to the system memory and waiting for memory to send back the next instruction byte or bytes.
Except in the case of JMP and CALL instructions, where the queue must be dumped and then reloaded starting from a new address, this pre-fetch and queue scheme greatly speeds up processing.
Fetching the next instruction while the current instruction executes is called **pipelining**.

## REGISTER ORGANIZATION IN 8086

8086 has a powerful set of registers known as *general purpose registers* and *special purpose registers.*
All of them are 16-bit registers.
*General purpose registers:*

- These registers can be used as either 8-bit registers or 16-bit registers.
- They may be either used for holding data, variables and intermediate results temporarily or for other purposes like a counter or for storing offset address for some particular addressing modes etc.

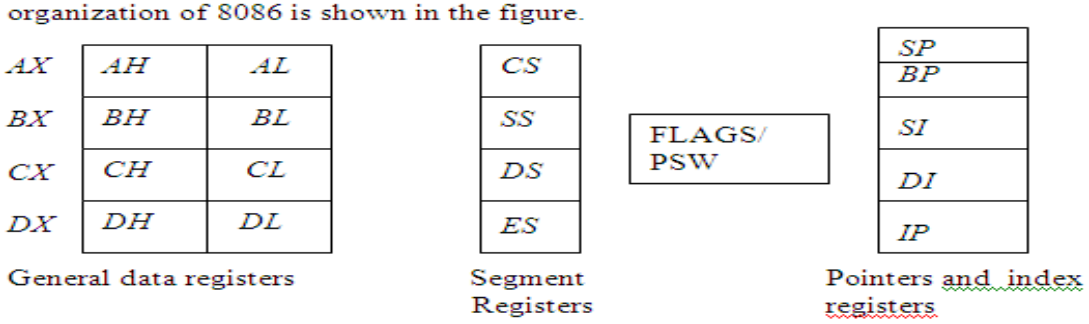
*Special purpose registers:*

- These registers are used as segment registers, pointers, index registers or as offset storage registers for particular addressing modes.

The 8086 registers are classified into the following types:

- General Data Registers
- Segment Registers
- Pointers and Index Registers
- Flag Register

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




Register organization of 8086

## General Purpose Registers:

The registers *AX, BX, CX* and *DX* are the general purpose 16-bit registers.

***AX*** is used as **16-bit accumulator**. The lower 8-bit is designated as *AL* and higher 8-bit is designated as *AH*. *AL* can be used as an 8-bit accumulator for 8-bit operation.

All data register can be used as either 16 bit or 8 bit. *BX* is a 16 bit register, but *BL* indicates the lower 8-bit of *BX* and *BH* indicates the higher 8-bit of *BX*.

The register ***BX*** is used as **offset** storage for forming physical address in case of certain addressing modes.

The register ***CX*** is used **default counter** in case of string and loop instructions.

***DX*** register is a general purpose register which may be used as an **implicit operand** or **destination** in case of a few instructions.

## Segment Registers:

There are 4 segment registers. They are:

- Code Segment Register(CS)
- Data Segment Register(DS)
- Extra Segment Register(ES)
- Stack Segment Register(SS)

The 8086 architecture uses the concept of **segmented memory**. 8086 able to address a memory capacity of 1 megabyte and it is byte organized. This 1 megabyte memory is divided into 16 logical segments. Each segment contains 64 kbytes of memory.

Code segment register (CS): is used for addressing memory location in the code segment of the memory, where the executable program is stored.

Data segment register (DS): points to the data segment of the memory where the data is stored.

Extra Segment Register (ES) : also refers to a segment in the memory which is another data segment in the memory.

Stack Segment Register (SS): is used for addressing stack segment of the memory. The stack segment is that segment of memory which is used to store stack data.

While addressing any location in the memory bank, the **physical address** is calculated from two parts:

*Physical address= 10H*segment address + offset address*

The first is segment address, the segment registers contain 16-bit segment base addresses, related to different segment.

The second part is the offset value in that segment.

## Pointers and Index Registers:

The index and pointer registers are given below:

- IP—Instruction pointer-store memory location of next instruction to be executed
- BP—Base pointer
- SP—Stack pointer
- SI—Source index
- DI—Destination index

**The pointers registers contain offset within the particular segments.**

- The pointer register *IP* contains offset within the code segment.
- The pointer register *BP* contains offset within the data segment.
- The pointer register *SP* contains offset within the stack segment.

The index registers are used as general purpose registers as well as for offset storage in case of indexed, base indexed and relative base indexed addressing modes.
The register *SI* is used to store the offset of source data in data segment.
The register *DI* is used to store the offset of destination in data or extra segment.
The index registers are particularly useful for string manipulation.

## 8086 flag register and its functions:

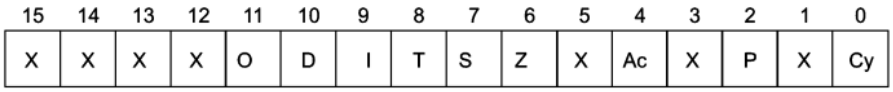
The 8086 flag register contents indicate the results of computation in the *ALU*. It also contains some flag bits to control the *CPU* operations.
A 16 bit flag register is used in 8086. It is divided into two parts .

- Condition code or status flags
- Machine control flags

The **condition code flag register** is the lower byte of the 16-bit flag register. The condition code flag register is identical to 8085 flag register, with an additional overflow flag.

The **control flag register** is the higher byte of the flag register. It contains three flags namely direction flag (*D*), interrupt flag (*I*) and trap flag (*T*).

Flag register configuration




O — Overflow flag
D — Direction flag
I — Interrupt flag
T — Trap flag
S — Sign flag
Z — Zero flag
Ac — Auxiliary carry flag
P — Parity flag
Cy — Carry flag
X — Not used

***Flag Register of 8086***

The description of each flag bit is as follows:

***SF*- Sign Flag:** This flag is set, when the result of any computation is negative. For signed computations the sign flag equals the MSB of the result.

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

***PF*- Parity Flag:** This flag is set to 1, if the lower byte of the result contains even number of 1's.

***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.

***AF*-Auxilary 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.

***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.

***TF*- Tarp Flag:** If this flag is set, the processor enters the single step execution mode. The processor executes the current instruction and the control is transferred to the Trap interrupt service routine.

***IF*- Interrupt Flag:** If this flag is set, the maskable interrupts are recognized by the CPU, otherwise they are ignored.

***D*- 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 decrementing mode.

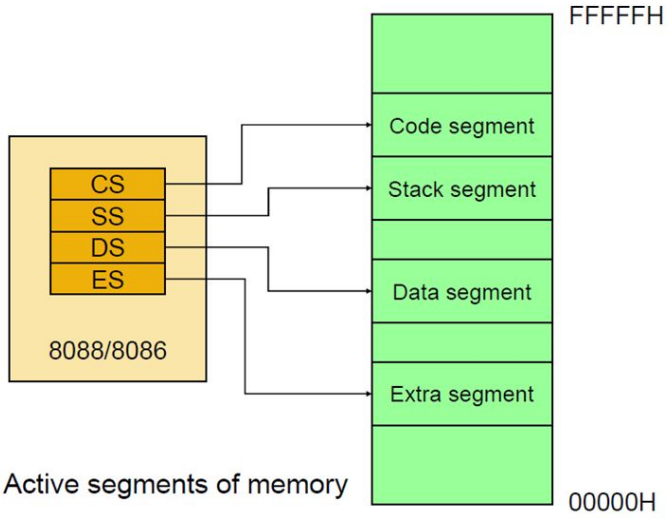
## Memory Segmentation in 8086

The memory in an 8086 based system is organized as segmented memory.

The CPU 8086 is able to access 1MB of physical memory. The complete 1MB of memory can be divided into 16 segments, each of 64KB size and is addressed by one of the segment register.

The 16-bit contents of the segment register actually point to the starting location of a particular segment. The address of the segments may be assigned as 0000H to F000h respectively.

To address a specific memory location within a segment, we need an offset address. The offset address values are from 0000H to FFFFH so that the physical addresses range from 00000H to FFFFFH.



**Physical address is calculated as below:**

Ex: Segment address—------→1005H

Offset address ----------→ 5555H

Segment address -------→ 1005H 0001 0000 0000 0101

Shifted left by 4 Positions-----→ 0001 0000 0000 0101 0000 +

Offset address-----→ 5555H -------→ 0101 0101 0101 0101

—----------

Physical address 0001 0101 0101 1010 0101 = 155A5H

**Physical address = Segment address * 10H + Offset address.**

**The main advantages of the segmented memory scheme are as follows:**

1. Allows the memory capacity to be 1MB although the actual addresses to be handled are of 16-bit size.
2. Allows the placing of code, data and stack portions of the same program in different parts (segments) of memory, for data and code protection.
3. Permits a program and/or its data to be put into different areas of memory each time the program is executed, i.e., provision for relocation is done.

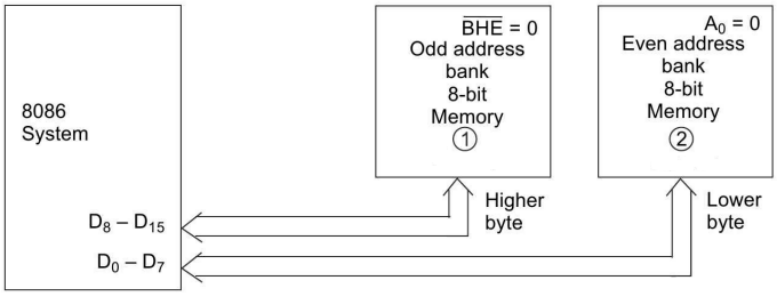


## Physical Memory Organization in 8086

- The total memory (1MB) of 8086 is arranged in two banks. An odd bank and an even bank. Both the banks have equal no. of locations.
- The odd bank contains odd numbered memory Locations. It is known as upper bank.
- The even bank contains only even numbered memory. locations. It is known as lower bank.
- This arrangement is done in order to speed up the operation. Every memory location has two kinds of address – physical and logical. A physical address is the 20-bit value that uniquely identifies each byte location in the Mega byte memory space. These may range from 0 to FFFFF Hex.
- A logical address (effective address/physical address) consists of a segment base value and an offset value. Byte data with even address is transferred on D7 - D0 and byte data with odd address is transferred on D15 - D8
- The processor provides two enable signals, BHE' and A0 for selecting of either even or odd or both the banks.

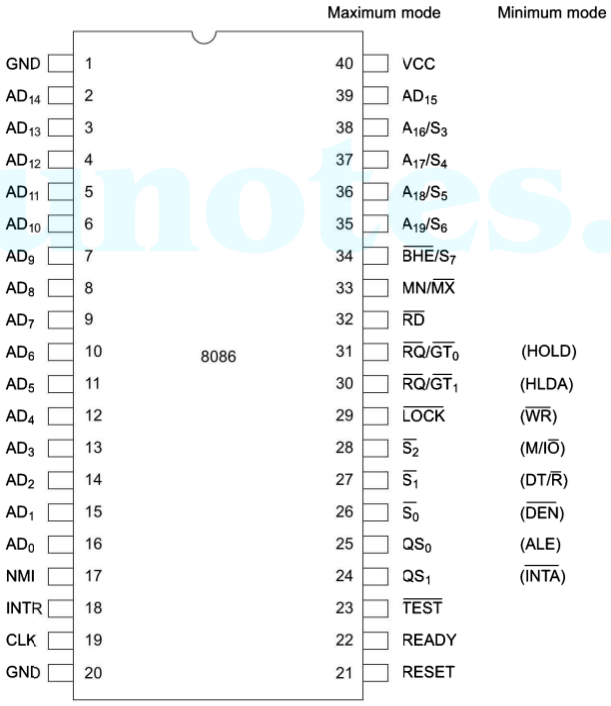
**Bus High Enable and $A_0$**

| $\overline{BHE}$ | $A_0$ | *Indication* |
|---|---|---|
| 0 | 0 | Whole word (2 bytes) |
| 0 | 1 | Upper byte from or to odd address. |
| 1 | 0 | Lower byte from or to even address |
| 1 | 1 | None |




***Physical Memory Organisation***

## Pin Diagram of 8086:




***Pin Configuration of 8086***

## SIGNAL DESCRIPTION OF 8086:

The 8086 is a 16-bit microprocessor. This microprocessor operates in single processor or multiprocessor configurations to achieve high performance.

The pin configuration of 8086 is shown in the figure. Some of the pins serve a particular function in minimum mode (single processor mode) and others function in maximum mode (multiprocessor mode).

**The 8086 signals are categorized into 3 types:**

1. Common signals for both minimum mode and maximum mode.

2. Special signals which are meant only for minimum mode
3. Special signals which are meant only for maximum mode

## Common Signals for both Minimum mode and Maximum mode:

$AD_7 - AD_0$ : The address/ data bus lines are the multiplexed address data bus and contain the right-most eight bit of memory address or data. The address and data bits are separated by using *ALE* signal.

$AD_{15} - AD_8$ : The address/data bus lines compose the upper multiplexed address/data bus. This lines contain address bits $A_{15} - A_8$ or data bus $D_{15} - D_8$ . The address and data bits are separated by using *ALE* signal.

$A_{19}/S_6 - A_{18}/S_3$ **:**The address/status bus bits are multiplexed to provide address signals $A_{19} - A_{16}$ and also status bits $S_6 - S_3$ . The address bits are separated from the status bits using the ALE signals. The status bit $S_6$ **is always a logic 0, bit** $S_5$ **indicates the condition of the interrupt flag bit. The** $S_4$ **and** $S_3$ **are being used for indicating type of memory access.**

| $S_4$ | $S_3$ | Type of segment register used |
|---|---|---|
| 0 | 0 | Extra segment |
| 0 | 1 | Stack segment |
| 1 | 0 | Code or no segment |
| 1 | 1 | Data Segment |

$\overline{BHE}/S_7$ **:** The bus high enable (BHE) signal is used to indicate the transfer of data over the higher order ($D_{15} - D_8$) data bus. It goes low for the data transfer over $D_{15} - D_8$ and is used to derive chip select of odd address memory bank or peripherals.

**Bus High Enable and $A_0$**

| $\overline{BHE}$ | $A_0$ | *Indication* |
|---|---|---|
| 0 | 0 | Whole word (2 bytes) |
| 0 | 1 | Upper byte from or to odd address. |
| 1 | 0 | Lower byte from or to even address |
| 1 | 1 | None |

***READY*:** This is the acknowledgement from the slow devices or memory that they have completed the data transfer operation. This signal is active high.

***INTR***: Interrupt Request: Interrupt request is used to request a hardware interrupt of *INTR* is held high when interrupt enable flag is set, the 8086 enters an interrupt acknowledgement cycle after the current instruction has completed its execution.

$\overline{TEST}$ : This input is tested by "*WAIT*" instruction. If the *TEST* input goes low; execution will continue. Else the processor remains in an idle state.

***NMI*-** Non-maskable Interrupt: The non-maskable interrupt input is similar to *INTR* except that the *NMI* interrupt does not check for interrupt enable flag is at logic 1, i.e, *NMI* is not maskable internally by software. If *NMI* is activated, the interrupt input uses interrupt vector 2.

***RESET*:** The reset input causes the microprocessor to reset itself. When 8086 reset, it restarts the execution from memory location *FFFF0H*. The reset signal is active high and must be active for at least four clock cycles.

***CLK***: Clock input: The clock input signal provides the basic timing input signal for processor and bus control operation.

**$V_{CC}$ (+5V):** Power supply for the operation of the internal circuit

***GND**:* Ground for the internal circuit

***MN* / $\overline{MX}$** : The minimum/maximum mode signal to select the mode of operation either in minimum or maximum mode configuration. Logic 1 indicates minimum mode. Logic 0 indictes maximum mode.

**Minimum mode Signals: The following signals are for minimum mode operation of 8086.**

***M* / $\overline{IO}$ -** Memory/*IO* - $\overline{M}$ / *IO* signal selects either memory operation or I/O operation. This line indicates that the microprocessor address bus contains either a memory address or an I/O port address. Signal high at this pin indicates a memory operation.

$\overline{INTA}$ - Interrupt acknowledge: The interrupt acknowledge signal is a response to the INTR input signal. The *INTA* signal is normally used to gate the interrupt vector number onto the data bus in response to an interrupt request.

***ALE*-** Address Latch Enable: This output signal indicates the availability of valid address on the address/data bus, and is connected to latch enable input of latches.

***DT* / $\overline{R}$** : Data transmit/Receive: This output signal is used to decide the direction of date flow through the bi-directional buffer. *DT* / *R* = 1 Indicates transmitting and *DT* / *R* = 0 indicates receiving the data.

$\overline{DEN}$ Data Enable: Data bus enable signal indicates the availability of valid data over the address/data lines.

$\overline{WR}$ - Write: whenever the write signal is at logic 0, the data bus transmits (writes) the data to the memory or I/O devices connected to the system.

$\overline{RD}$ : Read signal when low indicates the data bus receives the data from the memory or I/O devices connected to the system.

***HOLD***: The hold input request a direct memory access (DMA). If the hold signal is at logic 1, the micro process stops its normal execution and places its address, data and control bus at the high impedance state.

***HLDA***: Hold acknowledgement indicates that 8086 has entered into the hold state (ready to allocate bus for DMA operation)

**Maximum mode signal: The following signals are for maximum mode operation of 8086.**

**$\overline{S_2}$, $\overline{S_1}$, $\overline{S_0}$ -** Status lines: These are the status lines that reflect the type of operation being carried out

by the processor.

These status lines are encoded as follows

| $\overline{S}_2$ | $\overline{S}_1$ | $\overline{S}_0$ | *Indication* |
|---|---|---|---|
| 0 | 0 | 0 | Interrupt acknowledge |
| 0 | 0 | 1 | Read I/O port |
| 0 | 1 | 0 | Write I/O port |
| 0 | 1 | 1 | Halt |
| 1 | 0 | 0 | Code access |
| 1 | 0 | 1 | Read memory |
| 1 | 1 | 0 | Write memory |
| 1 | 1 | 1 | Passive |

***LOCK*** : The lock output is used to lock peripherals off the system, i.e, the other system bus masters will be prevented from gaining the system bus.

***$QS_1$* and *$QS_0$*** - Queue status: The queue status bits shows the status of the internal instruction queue. The encoding of these signals is as follows

| $QS_1$ | $QS_0$ | *Indication* |
|---|---|---|
| 0 | 0 | No operation |
| 0 | 1 | First byte of opcode from the queue |
| 1 | 0 | Empty queue |
| 1 | 1 | Subsequent byte from the queue |

***$\overline{RQ}/\overline{GT}1$* and *$\overline{RQ}/\overline{GT}0$*** - request/Grant: The request/grant pins are used by other local bus masters to force the processor to release the local bus at the end of the processors current bus cycle. These lines are bi- directional and are used to both request and grant a *DMA* operation.

$\overline{RQ}/\overline{GT}_0$ having higher priority than $\overline{RQ}/\overline{GT}_1$

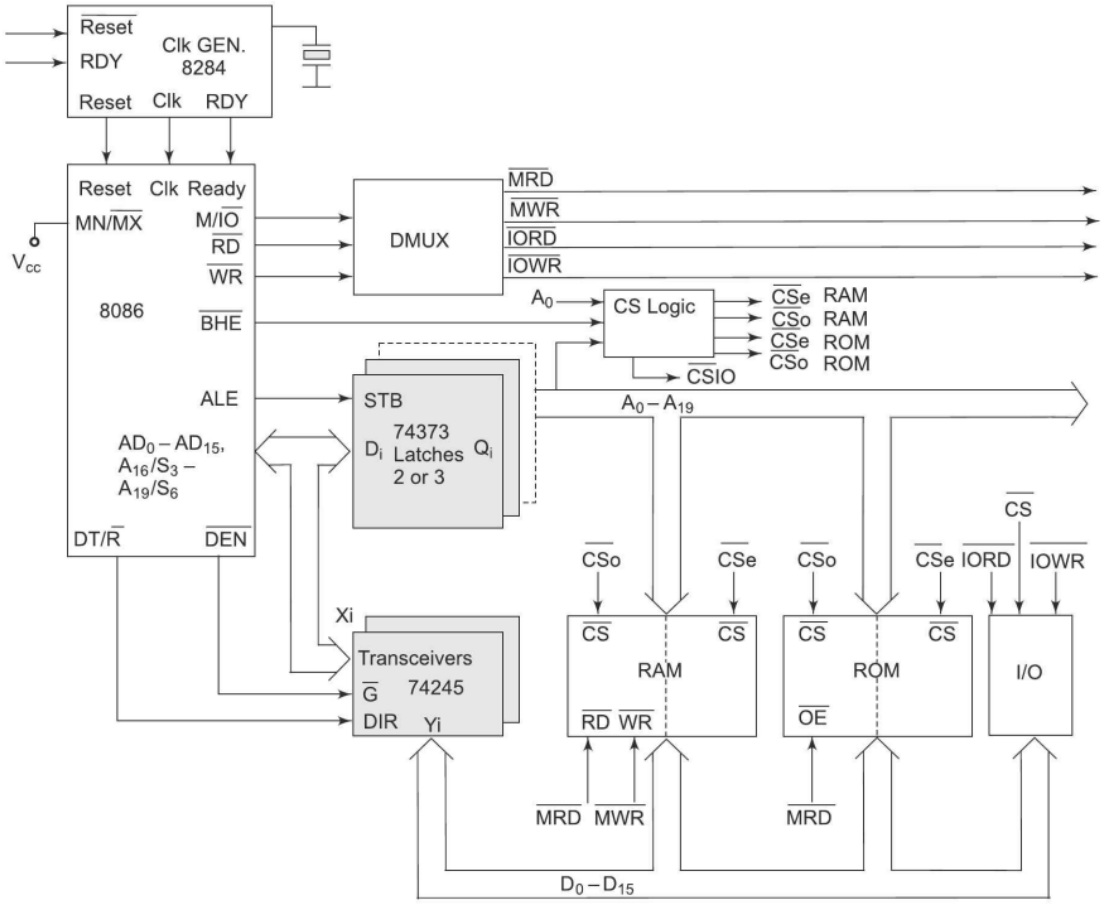
## 8086 MINIMUM MODE SYSTEM OPERATION WITH TIMING DIAGRAMS:

In a minimum mode 8086 system, the microprocessor 8086 is operated in minimum mode by strapping its MN/$\overline{MX}$ pin to logic1.

In this mode, all the control signals are given out by the microprocessor chip itself. There is a single microprocessor in the minimum mode system. The remaining components in the system are latches, transreceivers, clock generator, memory and I/O devices.

Some type of chip selection logic may be required for selecting memory or I/O devices, depending upon the address map of the system.

The general system organization is shown in below figure.




***Minimum Mode 8086 System***

The latches are generally buffered output D-type flip-flops, like, 74LS373 or 8282.
They are used for separating the valid address from the multiplexed address/data signals and are controlled by the ALE signal generated by 8086.
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.
Transreceivers are the bidirectional buffers and sometimes they are called as data amplifiers. They are required to separate the valid data from the time multiplexed address/data signal.
They are controlled by two signals, namely, DEN' and DT/R'. The DEN' signal indicates that the valid data is available on the data bus, while DT/R' indicates the direction of data, i.e. from or to the processor.
The system contains memory (RAM & ROM) for the data and users program storage.
A system may contain I/O devices for communication with the processor as well as some special purpose I/O devices.
The clock generator generates the clock from the crystal oscillator and then shapes it and divides to make it more precise so that it can be used as an accurate timing reference for the system.
The clock generator also synchronizes some external signals with the system clock.
The working of the minimum mode configuration system can be better described in terms of the timing diagrams rather than qualitatively describing the operations.

The opcode fetch and read cycles are similar. Hence the timing diagram can be categorized in two parts, the first is the timing diagram for read cycle and the second is the timing diagram for write cycle.

**Timing Diagrams:**

Timing diagram is graphical representation of the operations of microprocessor with respect to the time.

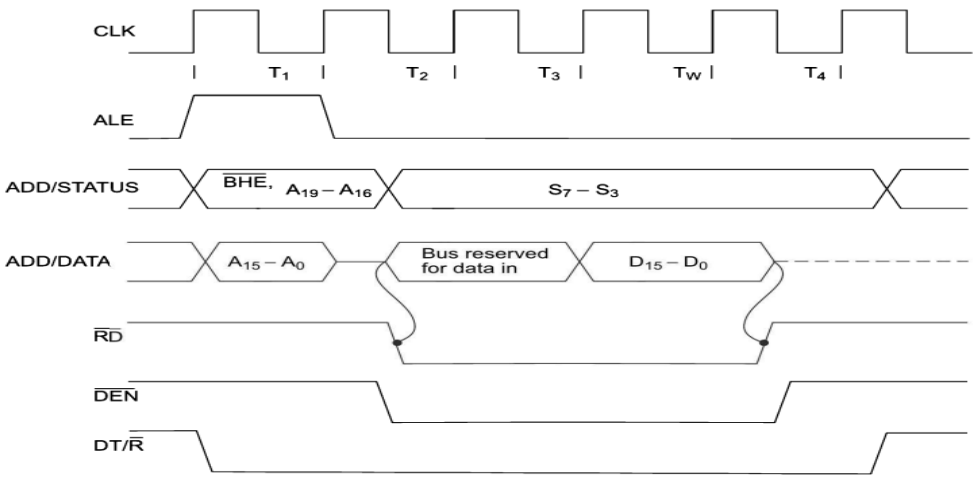
**State (clock cycle)**: one cycle of the clock is called state.

**Machine cycle:** The basic microprocessor operation such as reading a byte from memory or writing a byte to a port is called machine cycle and made up of more than one state.

**Instruction cycle:** The time required for microprocessor to fetch and execute an entire instruction is called Instruction cycle and made up of more than one machine cycle.

**Note:** An instruction cycle is made up of machine cycles, and a machine cycle is made up of states. The time for a state is determined by the frequency of the clock signal.

## Read cycle timing diagram for Minimum mode:




*Read Cycle Timing Diagram for Minimum Mode*

Basically, all the processor bus cycles consist of at least four clock cycles. These are referred to as $T_1$, $T_2$,$T_3$ and $T_4$. The address is transmitted by the processor during $T_1$. It is present on the bus only for one cycle. During $T_2$, i.e. the next cycle, the bus is tristated for changing the direction of bus for the following data read cycle. The data transfer takes place during $T_3$ and $T_4$. In case, an addressed device is slow and shows 'NOT READY' status the wait states $T_w$ are inserted between $T_3$ and $T_4$. These clock states during wait period are called *idle states* ($T_i$), *wait states* ($T_w$) or *inactive states*.

The M/IO', RD' and WR' signals indicate the types of data transfer as specified in Table

| $M/\overline{IO}$ | $\overline{RD}$ | $\overline{DEN}$ | *Transfer Type* |
|---|---|---|---|
| 0 | 0 | 1 | I/O read |
| 0 | 1 | 0 | I/O write |
| 1 | 0 | 1 | Memory read |
| 1 | 1 | 0 | Memory write |

**The read cycle** begins in $T_1$ with the assertion of the address latch enable (ALE) signal and also M/IO' signal.

During the negative going edge of this signal, the valid address is latched on the local bus.

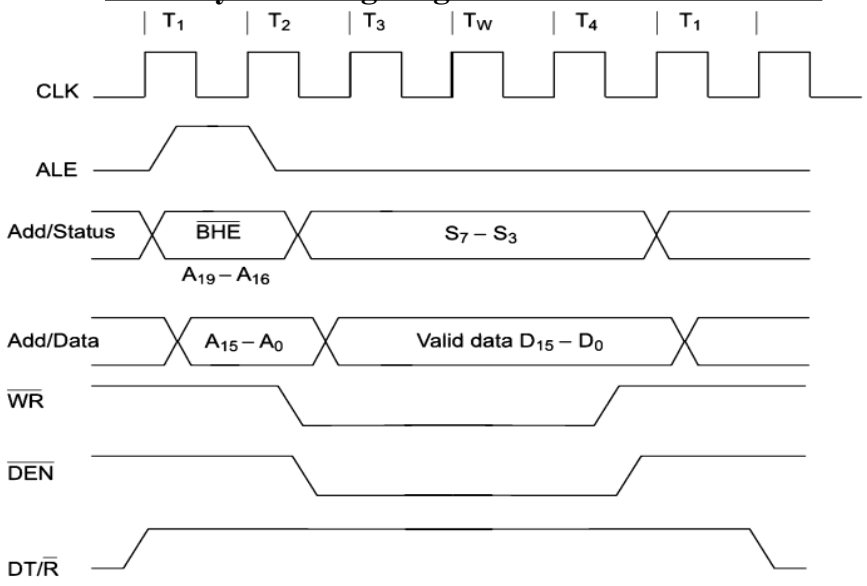
The BHE' and A0 signals address low, high or both bytes.

From $T_1$ to $T_4$, the M/IO' signal indicate a memory or I/O operation. At $T_2$, the address is removed from thelocal 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 tristate its bus drivers.

## Write cycle timing diagram for Minimum mode:



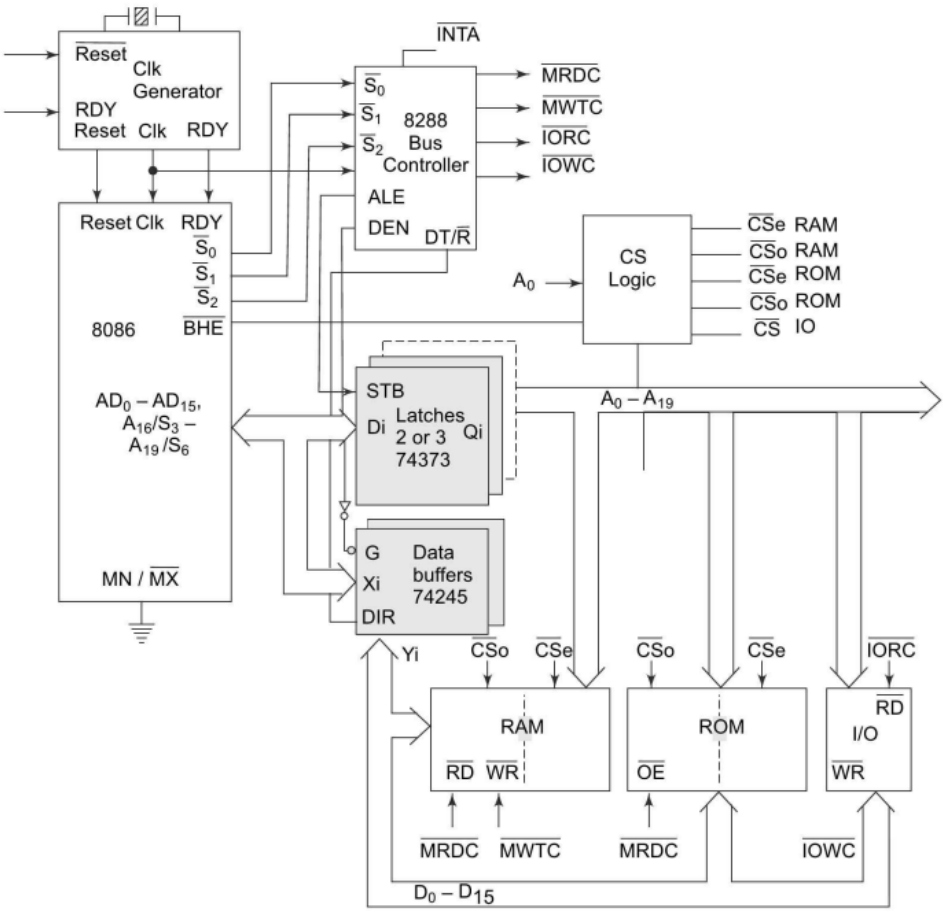

*Write Cycle Timing Diagram for Minimum Mode Operation*

**A write cycle** also begins with the assertion of ALE and the emission of the address. The M/IO' signal is again asserted to indicate a memory or I/O operation.
In $T_2$, after sending the address in $T_1$, the processor sends the data to be written to the addressed location. The data remains on the bus until middle of $T_4$ state. The WR' becomes active at the beginning of $T_2$ .

## 8086 MAXIMUM MODE SYSTEM OPERATION WITH TIMING DIAGRAMS:

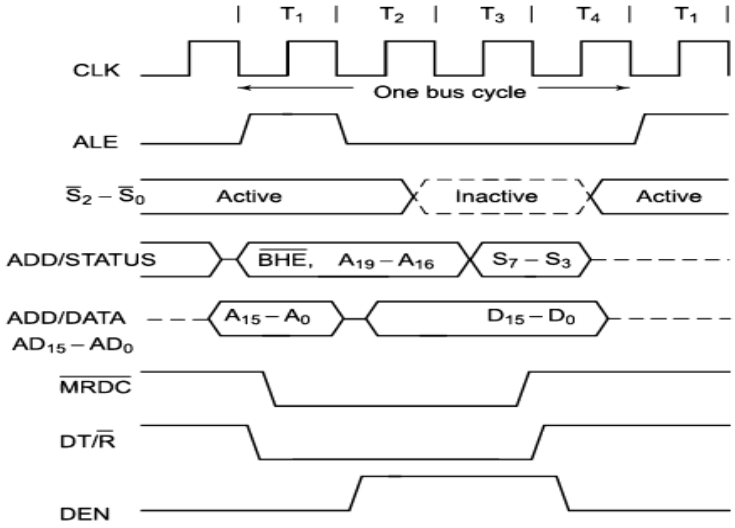
In the maximum mode, the 8086 is operated by strapping the MN/MX' pin to ground. In this mode, the processor derives the status signals S2', S1' and S0'. Another chip called bus controller derives the control signals using this status information.
In the maximum mode, there may be more than one microprocessor in the system configuration. The other components in the system are the same as in the minimum mode system. The general system organization is as shown in the below figure.

The basic functions of the bus controller chip IC 8288, is to derive control signals like RD'and WR' (for memory and I/O devices), DEN, DT/R', ALE, etc. using the information made available by the processor on the status lines.
The bus controller chip has input lines S2', S1' and S0' and CLK. These inputs to 8288 are driven by the CPU. It derives the outputs ALE, DEN, DT/R', MWTC', MRDC', IORC', IOWC' and INTA'.
INTA' pin is used to issue two interrupt acknowledge pulses to the interrupt controller or to an interrupting device.
IORC', IOWC' are I/O read command and I/O write command signals respectively. These signals enable an IO interface to read or write the data from or to the addressed port. The MRDC', MWTC'are memory read command and memory write command signals respectively and may be used as memory read and write signals. All these command signals instruct the memory to accept or send data from or to the bus.

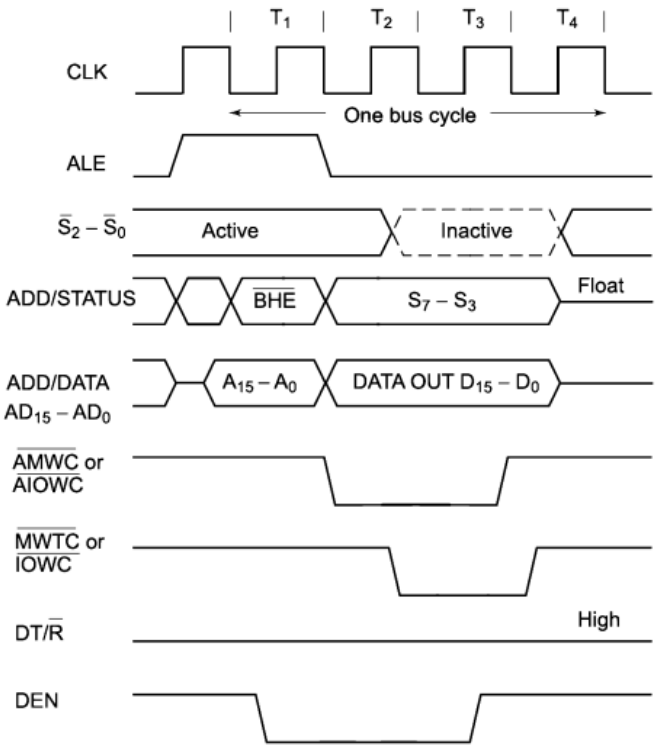



***Maximum Mode 8086 System***

The maximum mode system timing diagrams are also divided in two portions as read (input) and write (output) timing diagrams. The address/data and address/status timings are similar to the minimum mode. ALE is asserted in T1, just like minimum mode. The only difference lies in the status signals used and the available control and advanced command signals.




***Memory Read Timing in Maximum Mode***




*Memory Write Timing in Maximum Mode*

## COMPARISON OF 8086 AND 8088

| Sl.no | Key | 8086 | 8088 |
|---|---|---|---|
| 1 | Data and Address Bus | In the case of 8086 MPU the data bus is of 16 bits and the address bus is of 20 bits. | On other hand in 8088 MPU the data bus is of 8 bits (with internal 16-bit processing capability) and the address bus is of 20 bits. |
| 2 | Processing | 8086 has 3 available clock speeds (5 MHz, 8 MHz (8086-2) and 10 MHz (8086-1)). | 8088 has 2 available clock speeds (5 MHz, 8 MHz) |
| 3 | Signal Type | 8086 has memory control pin (M'/IO) signal. | 8088 has complemented memory control pin (IO'/M) signal of 8086. |
| 4 | Execution time | Less - due to 16 bit data lines | Takes more time for execution because only 8 - bits data are accessed at a time. All 16 - bit operations now require additional 4 clock cycles. |

| | | | |
|---|---|---|---|
| 5 | Instruction queue | It has 6-byte instruction queue. | It has 4-byte instruction queue as it can fetch only 1 byte at a time. |
| 6 | BHE signal | It has Bus High Enable (BHE) signal. | BHE' signal has no meaning as the data bus is of only 8-bits. |

## MACHINE LANGUAGE INSTRUCTION FORMAT

For every **instruction that is executed in the 8086 microprocessor**, an **instruction format** is available that is the binary representation of that instruction.

This instruction format can be coded from 1 to 6 bytes depending upon the addressing modes used for instructions.

Various fields used 8086 instruction format are:

1. **OPCODE:** The Opcode stands for Operation Code. It specifies the operation to be performed.
2. **MOD (2-bits):** Specifies the addressing mode used.
3. **REG (3-bits):** Specifies the register used.
4. **R/M (3-bits):** Register/Memory used
5. **D (1-bit):** Direction bit D = 0 — Source operand

   D = 1 — Destination operand
6. **W (1-bit):** W = 0 — 8-bit operand

   W = 1 — 16-bit operand
7. **S (1-bit):** Sign bit, used along with W bit.

   S = 1 — Signed operand
8. **V (1-bit):** used in shift and rotation instructions. V = 0 - Shift count = 1

   V = 1 - Shift count in CL register
9. **Z (1 - bit):** used in repeat instructions (eg:loop)

   Various instruction formats are:

**1. One Byte Instruction:**

- ➢ This format is only one byte long and may have the implied data or register operands.
- ➢ The least significant 3-bits of the opcode are used for specifying the register operand if

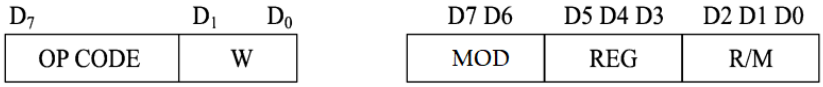
any.

- Otherwise, all the 8 bits form an opcode, and the operands are implied


Eg: CLC

**2. Register to Register:**

- This format is 2 bytes long
- The first byte of the code specifies the operation code and width of the operand specified by 'w' bit.
- The second byte of the code shows the register operands and R/M field, as shown below.
- The register represented by the REG field is one of the operands.
- The R/M field specifies another register or memory location i.e. the other operand.




Eg: MOV AL , CL

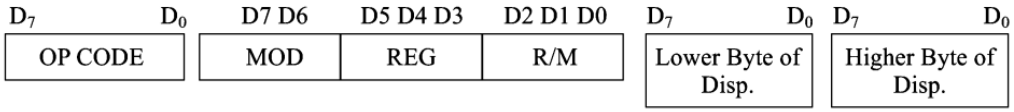
**3. Register to/from memory with no displacement:**

- This format is also 2 bytes long and similar to the Register to Register format except for the MOD field as shown.
- The MOD field shows the mode of addressing.


Eg: MOV AL , [BX]

**4. Register to/from Memory with Displacement:**

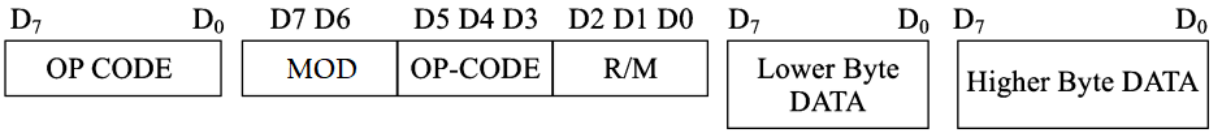
- This type of instruction format contains 1 or 2 additional bytes for displacement along with 2 byte format of the register to/from memory without displacement. The format is as shown below.




Eg: MOV [1234H + SI] , DH

**5. Immediate Operand to Register:**

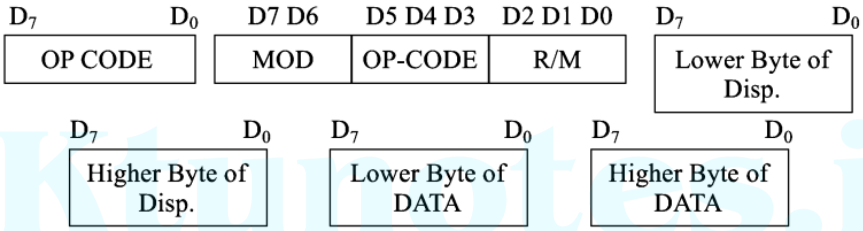
- ➢ In this format, the first byte as well as the 3-bits from the second byte which are used for REG field in case of register to register format are used for opcode.
- ➢ It also contains one or two bytes of immediate data.
- ➢ The complete instruction format is as shown below.




Eg: MOV AX , 5020H

**6. Immediate Operand to Memory with 16-bit displacement:**

- ➢ This type of instruction format requires 5 or 6 bytes for coding.
- ➢ The first 2 bytes contain the information regarding OPCODE, MOD and R/M fields.
- ➢ The remaining 4 bytes contain 2 bytes of displacement and 2 bytes of data as shown.




Eg: MOV [2000H + SI] , FF34H