UNIT-I
INTRODUCTION TO 8086

Unit-I

8086 Microprocessor

"Microprocessor" is an integrated circuit that contains all the functions of a central processing unit of a computer.

Microprocessor is nothing but CPU on a single chip.

"Evolution of Microprocessors":-

* The world's first microprocessor was Intel 4004 which is a 4-bit microprocessor. It uses PMOS Technology. It was introduced in 1971. Address width 12 (multiplexed)

* The Intel 8008 was introduced in 1972. It was an 8-bit CPU with an external 14-bit address bus that could address 16 KB of memory. It used NMOS technology.

* In 1974 Intel came with 8080 and Motorola with 6800 which are 8-bit microprocessors. These improved the speed and increased the memory size.
  speed - 500k instructions per second i.e., 2μs for one instruction
  Memory - 64KB

* In 1977 Intel came with 8085 and Zilog with Z80.
  Speed increased further.
  speed - 1.3 μs for one instruction.

* In 1978 Intel came with 8088 and 8086. Speed, memory and width of data increased.
  speed - 400 ns for one instruction
  Memory - 1 MB
  Data bus - 8 bits for 8088 and 16-bits for 8086.

(*). In 1983 Intel came with 80186. It increased the speed and memory further.
speed - 50 ns per instruction
Memory - 16MB.

(*). In 1986 Intel came with 80386. Increased speed, data width and memory and introduced Memory Management Unit concept. It uses HMOS technology.
speed - 100 ns per instruction (10 MIPs)
Databus - 32 bit
Memory - 4 GB

(*). In 1989 Intel came with 80486. This microprocessor included 80387 co-processor and 8KB cache memory.
speed increased to (50MIPs).

(*). In 1993 Intel introduced Pentium with increased data width and larger internal cache. It introduced the super scalar technology i.e, two independent instructions are executed simultaneously.

History fo Microprocessors:

Processor	No. of bits	Clock speed (Hz)	Year of introduction
4004	4	740K	1971
8008	8	500K	1972
8080	8	2M	1974
8085	8	3M	1976

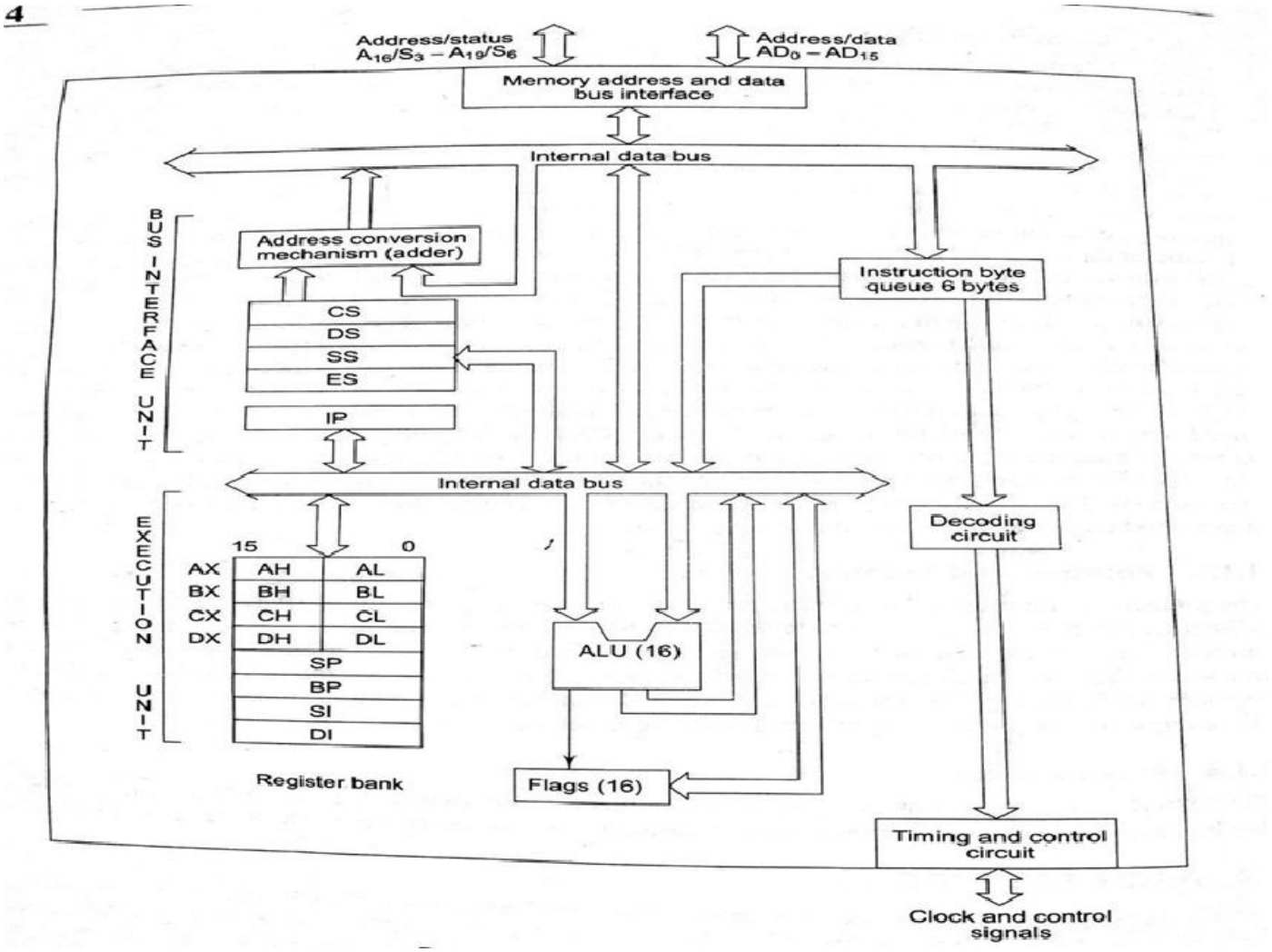
8086	16	5, 8 or 10M	1978
8088	16	5, 8 or 10M	1979
80186	16	6M	1982
80286	16	8M	1982
80386	32	16 to 33M	1986
80486	32	16 to 100M	1989
Pentium	32	66M	1993
Pentium II	32	233 to 500M	1997
Pentium III	32	500M to 1.4G	1999
Pentium IV	32	1.3 to 3.8G	2000
Dual core	32	1.2 to 3 G	2006
Core 2 Duo	64	1.2 to 3G	2006
i3, i5 and i7	64	2.4G to 3.6G	2010

8086 Microprocessor features:

1. It is 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~~ **IT PROVIDES 8,16 BIT REGISTERS**
6. It has multiplexed address and data bus AD_0-AD_{15} & A_{16}-A_{19}
7. It requires single phase clock with 33% duty cycle to provide internal timing.
8. Prefetches up to 6 instruction bytes from memory and queues them in order to speed up the processing.
9. 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.



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) which can add, subtract, AND, OR, XOR, increment, decrement, complement or shift binary numbers.
- The main functions of EU are:
  - Decoding of Instructions
  - Execution of instructions
    - ✓ Steps
      - EU extracts instructions from top of 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.

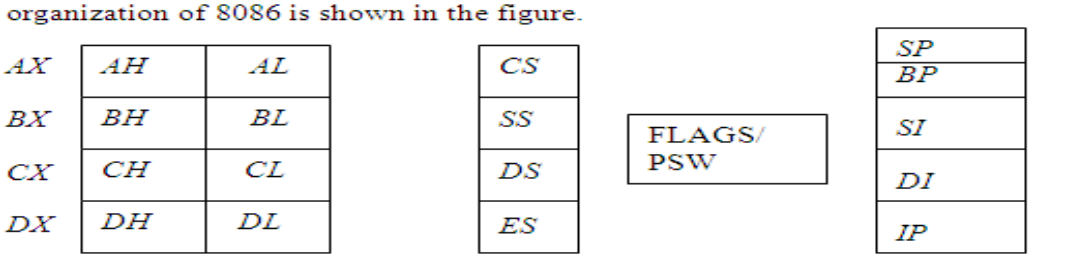
8086 HAS PIPELINING ARCHITECTURE:

- 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:

- 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 Data 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= 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:

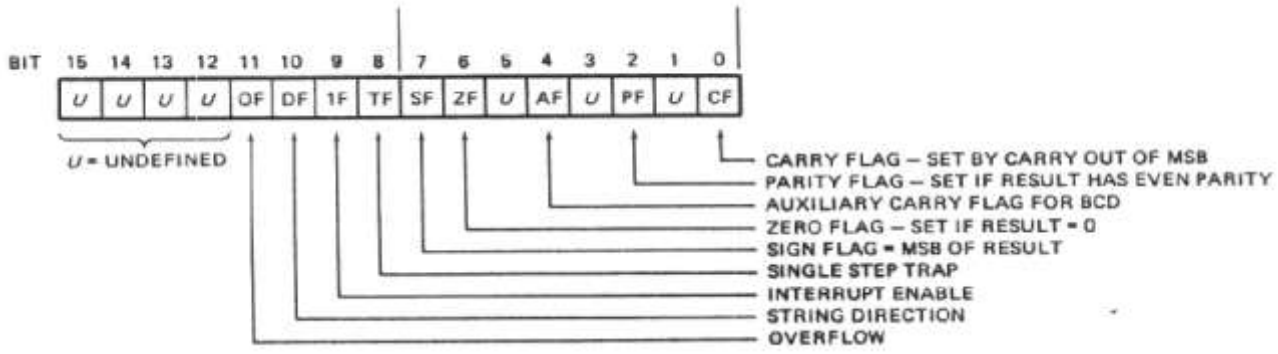
- The index and pointer registers are given below:
  - IP—Instruction pointer-store memory location of next instruction to be executed
  - BP—Base pointer
  - SP—Stack pointer
  - SI—Source index
  - DI—Destination index
- The pointers registers contain offset within the particular segments.
  - The pointer register *IP* contains offset within the code segment.
  - The pointer register *BP* contains offset within the data segment.
  - Thee 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



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 mask able 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:

- 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 -------155A5H 0001 0101 0101 1010 0101

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.

Overlapping and Non-overlapping Memory segments:

- In the overlapping area locations physical address = CS1+IP1 = CS2+IP2. Where '+' indicates the procedure of physical address formation.

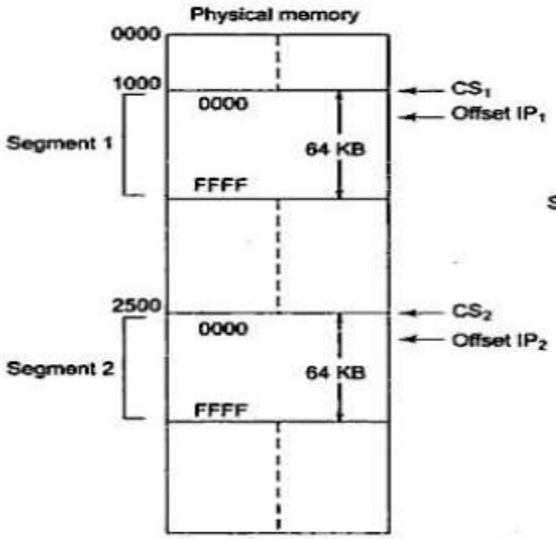


Fig. 1.3(a) *Non-overlapping Segments*

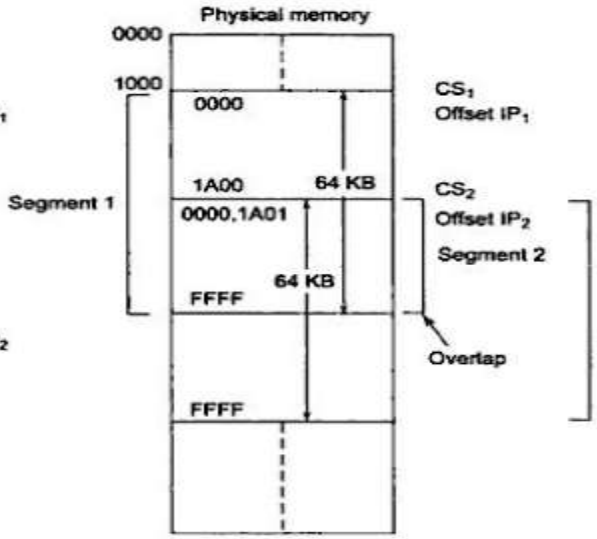


Fig. 1.3(b) *Overlapping Segments*

Addressing modes of 8086:

- Addressing mode indicates a way of locating data or operands.
- The addressing modes describe the types of operands and the way they are accessed for executing an instruction.
- According to the flow of instruction execution, the instructions may be categorized as
  - i) Sequential control flow instructions and
  - ii) Control transfer instructions

Sequential control flow instructions are the instructions, which after execution, transfer control to the next instruction appearing immediately after it (in the sequence) in the program. For example, the arithmetic, logic, data transfer and processor control instructions are sequential control flow instructions.

The **control transfer instructions**, on the other hand, transfer control to some predefined address or the address somehow specified in the instruction, after their execution. For example, INT, CALL, RET and JUMP instructions fall under this category.

The addressing modes for sequential control transfer instructions are:

1. **Immediate:** In this type of addressing, immediate data is a part of instruction and appears in the form of successive byte or bytes.

Ex: MOV AX, 0005H

In the above example, 0005H is the immediate data. The immediate data may be 8-bit or 16-bit in size.

2. **Direct:** In the direct addressing mode a 16-bit memory address (offset) is directly specified in the instruction as a part of it.

Ex: MOV AX, [5000H]

Here, data resides in a memory location in the data segment, whose effective address may be completed using 5000H as the offset address and content of DS as segment address. The effective address here, is 10H * DS + 5000H.

3. **Register:** In register addressing mode, the data is stored in a register and is referred using the particular register. All the registers, except IP, may be used in this mode.

Ex: MOV BX, AX

4. **Register Indirect:** Sometimes, the address of the memory location, which contains data or operand, is determined in an indirect way, using the offset register. This mode of addressing is known as register indirect mode. In this addressing mode, the offset address of data is in either BX or SI or DI register. The default segment is either DS or ES. The data is supposed to be available at the address pointed to by the content of any of the above registers in the default data segment.

Ex: MOV AX, [BX]

Here, data is present in a memory location in DS whose offset address is in BX. The effective address of the data is given as 10H * DS+[BX].

5. **Indexed:** In this addressing mode, offset of the operand is stored in one of the index registers. DS and ES are the default segments for index registers, SI and DI respectively. This is a special case of register indirect addressing mode.

Ex: MOV AX, [SI]

Here, data is available at an offset address stored in SI in DS. The effective address, in this case, is computed as 10*DS+[SI].

6. **Register Relative:** In this addressing mode, the data is available at an effective address formed by adding an 8-bit or 16-bit displacement with the content of any one of the registers BX, BP, SI and DI in the default (either DS or ES) segment.

Ex: MOV AX, 50H[BX]

Here, the effective address is given as 10H *DS+50H+[BX]

7. **Based Indexed:** The effective address of data is formed, in this addressing mode, by adding content of a base register (any one of BX or BP) to the content of an index register (any one of SI or DI). The default segment register may be ES or DS.

Ex: MOV AX, [BX][SI]

Here, BX is the base register and SI is the index register the effective address is computed as 10H * DS + [BX] + [SI].

8**. Relative Based Indexed:** The effective address is formed by adding an 8 or 16-bit displacement with the sum of the contents of any one of the base register (BX or BP) and any one of the index register, in a default segment.

Ex: MOV AX, 50H [BX] [SI]

Here, 50H is an immediate displacement, BX is base register and SI is an index register the effective address of data is computed as

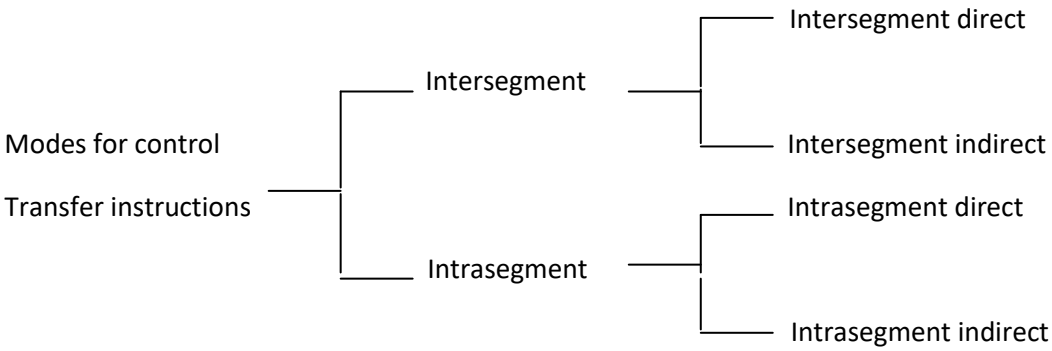
10H * DS + [BX] + [SI] + 50H

For control transfer instructions, the addressing modes depend upon whether the destination is within the same segment or different one. It also depends upon the method of passing the destination address to the processor.

Basically, there are two addressing modes for the control transfer instructions, **intersegment** addressing and **intrasegment** addressing modes.

If the location to which the control is to be transferred lies in a different segment other than the current one, the mode is called intersegment mode.

If the destination location lies in the same segment, the mode is called intrasegment mode.



Addressing modes for Control Transfer Instructions

9. **Intrasegment Direct Mode:** In this mode, the address to which the control is to be transferred lies in the same segment in which the control transfer instruction lies and appears directly in the instruction as an immediate displacement value. In this addressing mode, the displacement is computed relative to the content of the instruction pointer IP.

The effective address to which the control will be transferred is given by the sum of 8 or 16-bit displacement and current content of IP. In the case of jump instruction, if the signed displacement (d) is of 8-bits (i.e –128<d<+128) we term it as short jump and if it is of 16-bits (i.e-32, 768<d<+32,768) it is termed as long jump.

10. **Intrasegment Indirect Mode:** In this mode, the displacement to which the control is to be transferred, is in the same segment in which the control transfer instruction lies, but it is passed to the instruction indirectly. Here, the branch address is found as the content of a register or a memory location. This addressing mode may be used in unconditional branch instructions.

11. **Intersegment Direct:** In this mode, the address to which the control is to be transferred is in a different segment. This addressing mode provides a means of branching from one code segment to another code segment. Here, the CS and IP of the destination address are specified directly in the instruction.

12. **Intersegment Indirect:** In this mode, the address to which the control is to be transferred lies in a different segment and it is passed to the instruction indirectly, i.e contents of a memory block containing four bytes, i.e IP (LSB), IP(MSB), CS(LSB) and CS (MSB) sequentially. The starting address of the memory block may be referred using any of the addressing modes, except immediate mode.

Forming the effective Addresses:

The following examples explain forming of the effective addresses in the different modes.

Ex: 1. The contents of different registers are given below. Form effective addresses for different addressing modes.

Offset (displacement)=5000H

[AX]-1000H, [BX]- 2000H, [SI]-3000H, [DI]-4000H, [BP]-5000H, [SP]-6000H,

[CS]-0000H, [DS]-1000H, [SS]-2000H, [IP]-7000H

Shifting segment address four bits to the left is equivalent to multiplying it by 16_D or 10_H

i. **Direct addressing mode:**

MOV AX,[5000H]

DS : OFFSET ⇔ 1000H : 5000H

10H*DS ⇔ 10000—segment address

offset ⇔ +5000---offset address

_____

15000H – Effective address

_____

ii. **Register indirect:**

MOV AX, [BX]

DS: BX ⇔ 1000H: 2000H

10H*DS ⇔ 10000—segment address

[BX] ⇔ +2000---offset address

_____

12000H – Effective address

_____

iii. **Register relative:**

MOV AX, 5000 [BX]

DS : [5000+BX]

10H*DS ⇔ 10000

offset ⇔ +5000

[BX] ⇔ +2000

_____

17000H – Effective address

_____

iv. **Based indexed:**

MOV AX, [BX] [SI]

DS : [BX + SI]

10H*DS ⇔ 10000

[BX] ⇔ +2000

[SI] ⇔ +3000

_____

15000H – Effective address

_____

v. Relative based index:

MOV AX, 5000[BX][SI]

DS : [BX+SI+5000]

10H*DS ⇔ 10000

[BX] ⇔ +2000

[SI] ⇔ +3000

+5000

_____

1A000H – Effective address

_____

Pin Diagram 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

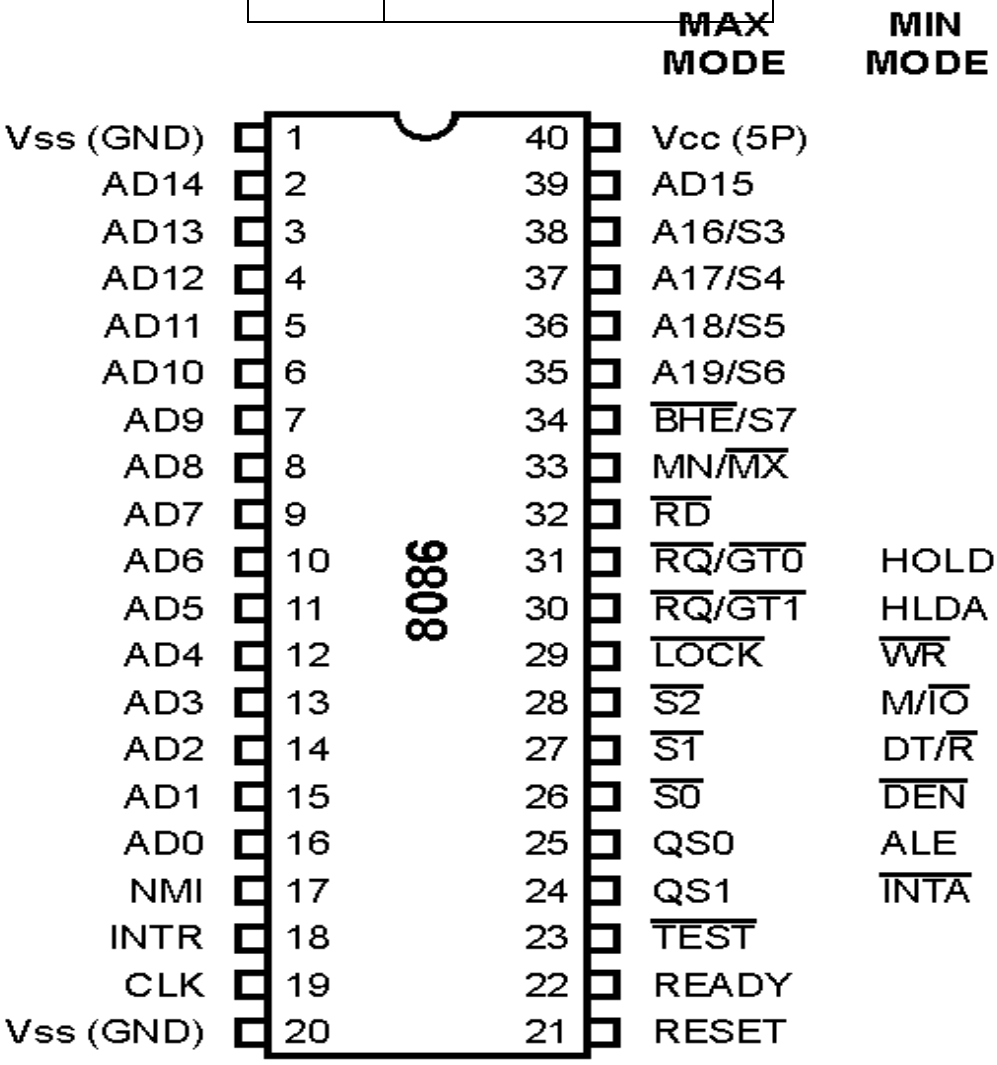
Common Signals for both Minimum mode and Maximum mode:

$AD_7 - AD_0$: 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 bit $A_{15} - A_8$ 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 indicate which segment register is presently being used for 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.

$\overline{BHE}$	A_0	Indication
0	0	Whole word
0	1	Upper byte from or to odd address
1	0	Lower byte from or to even address
1	1	None

$\overline{RD}$: Read: whenever the read signal is at logic

0, the data bus receives the data from the memory or I/O devices connected to the system

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. It is asymmetric square wave with 33% duty cycle.

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.

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

$M / \overline{IO}$ - Memory/*IO* $M / \overline{IO}$ 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. This line is logically equivalent to $\overline{S_2}$ in maximum mode.

$\overline{INTA}$ - Interrupt acknowledge: The interrupt acknowledge signal is a response to the INTR input signal. The $\overline{INTA}$ 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. $DT / \overline{R} = 1$ Indicates transmitting and $DT / \overline{R} = 0$ 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 the data to 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.

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

$\overline{S}_2$	$\overline{S}_1$	$\overline{S}_0$	Function
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 (In active)

$\overline{LOCK}$: 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	Function
0	0	No operation, queue is idle
0	1	First byte of opcode
1	0	Queue is empty
1	1	Subsequent byte of opcode

$\overline{RQ}/\overline{GT1}$ and $\overline{RQ}/\overline{GT0}$ - 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. $\overline{RQ}/\overline{GT0}$ is having higher priority than $\overline{RQ}/\overline{GT1}$

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/MX^{l} 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.

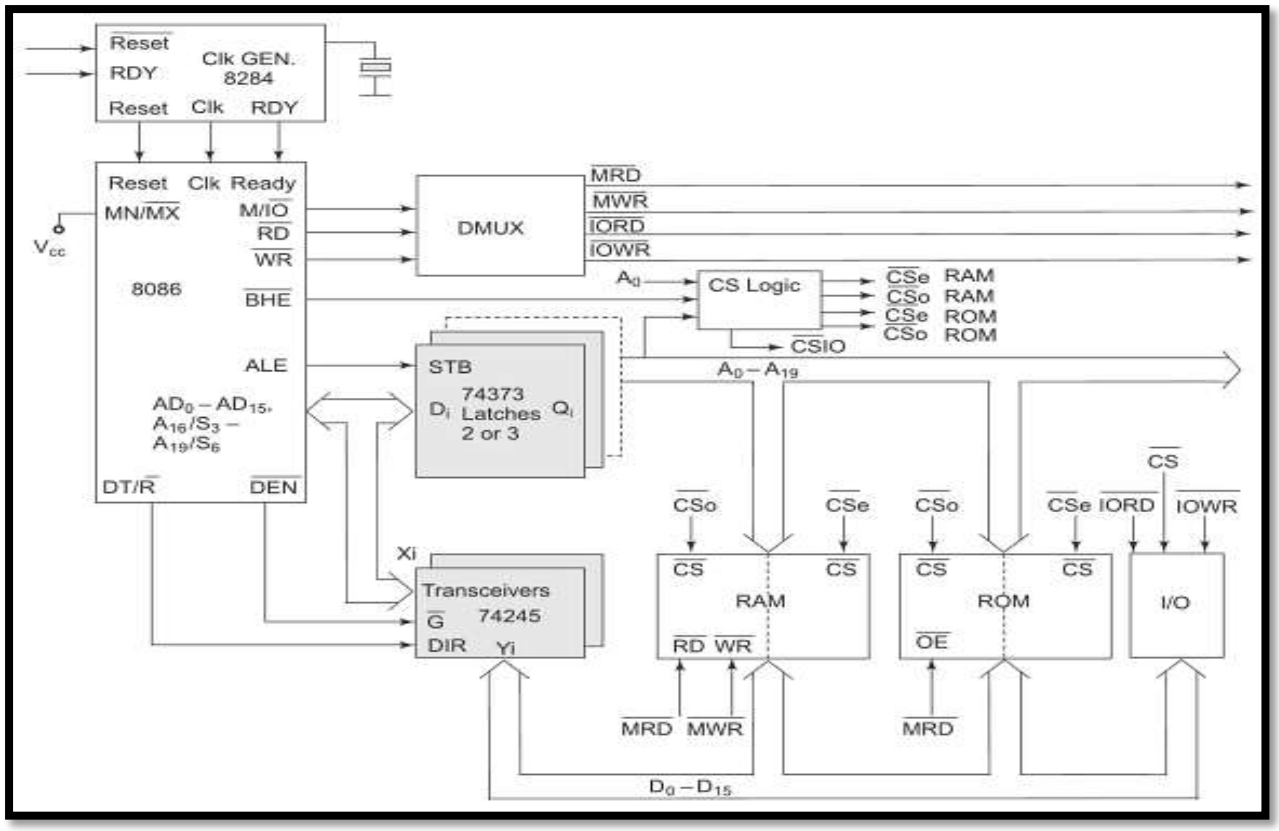


Figure: Minimum mode 8086 system

- 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 for the monitor and users program storage. Usually, EPROMS are used for monitor storage, while RAMs for 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**: 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:

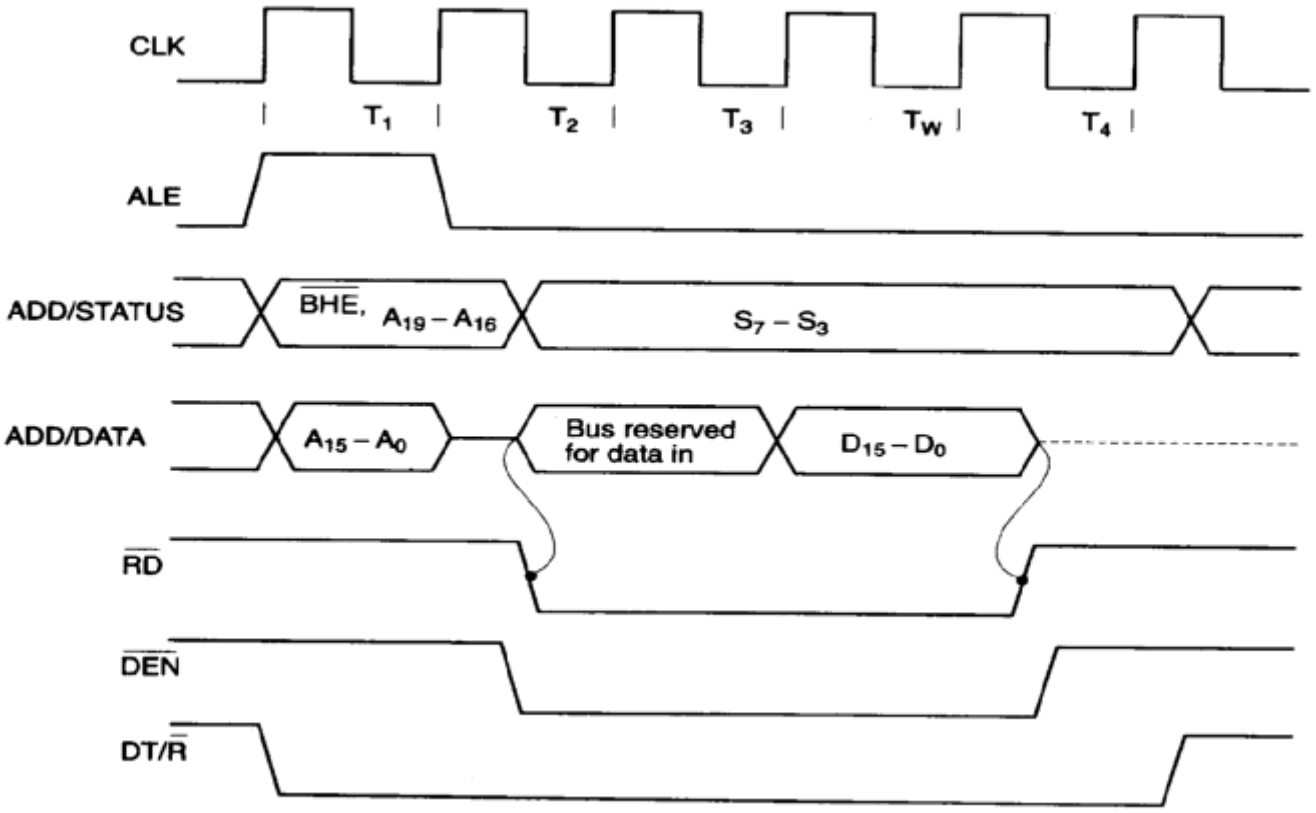


Fig. 1.9(a) *Read Cycle Timing Diagram for Minimum Mode*

- The best way to analyze a timing diagram such as the one to think of time as a vertical line moving from left to right across the diagram.
- **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 the local bus and is sent to the output. The bus is then tristated. The read ($\overline{RD}$) control signal is also activated in T_2.

- The read ($\overline{RD}$) signal causes the addressed device to enable its data bus drivers. After $\overline{RD}$ 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:

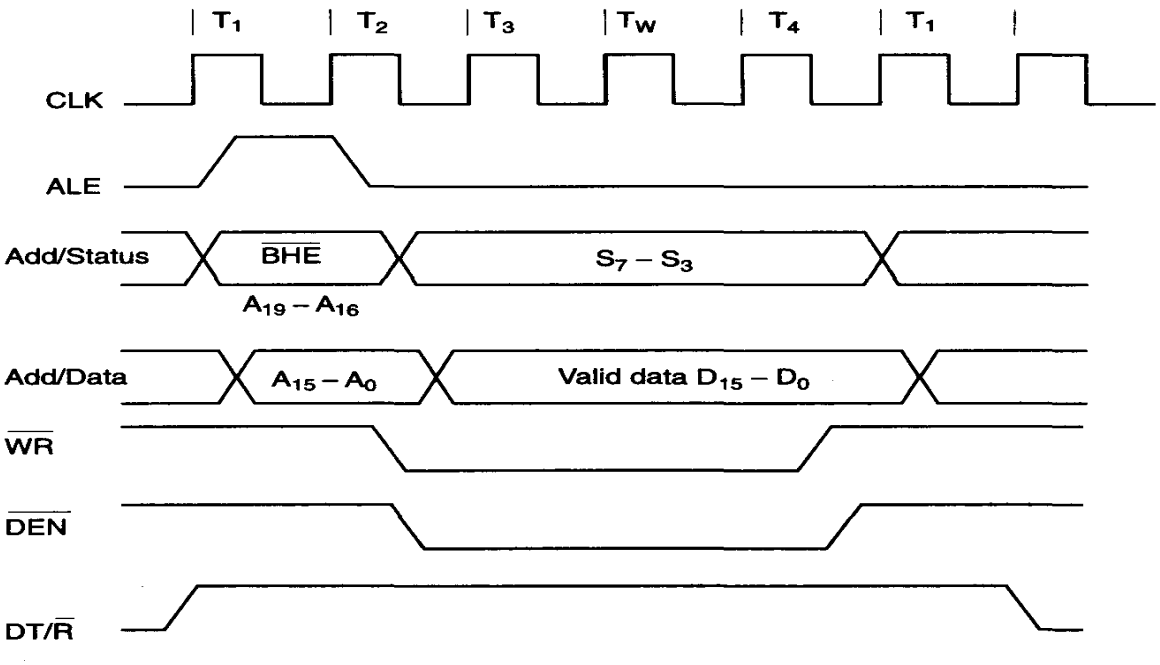


Fig. 1.9(b) ***Write Cycle Timing Diagram for Minimum Operation***

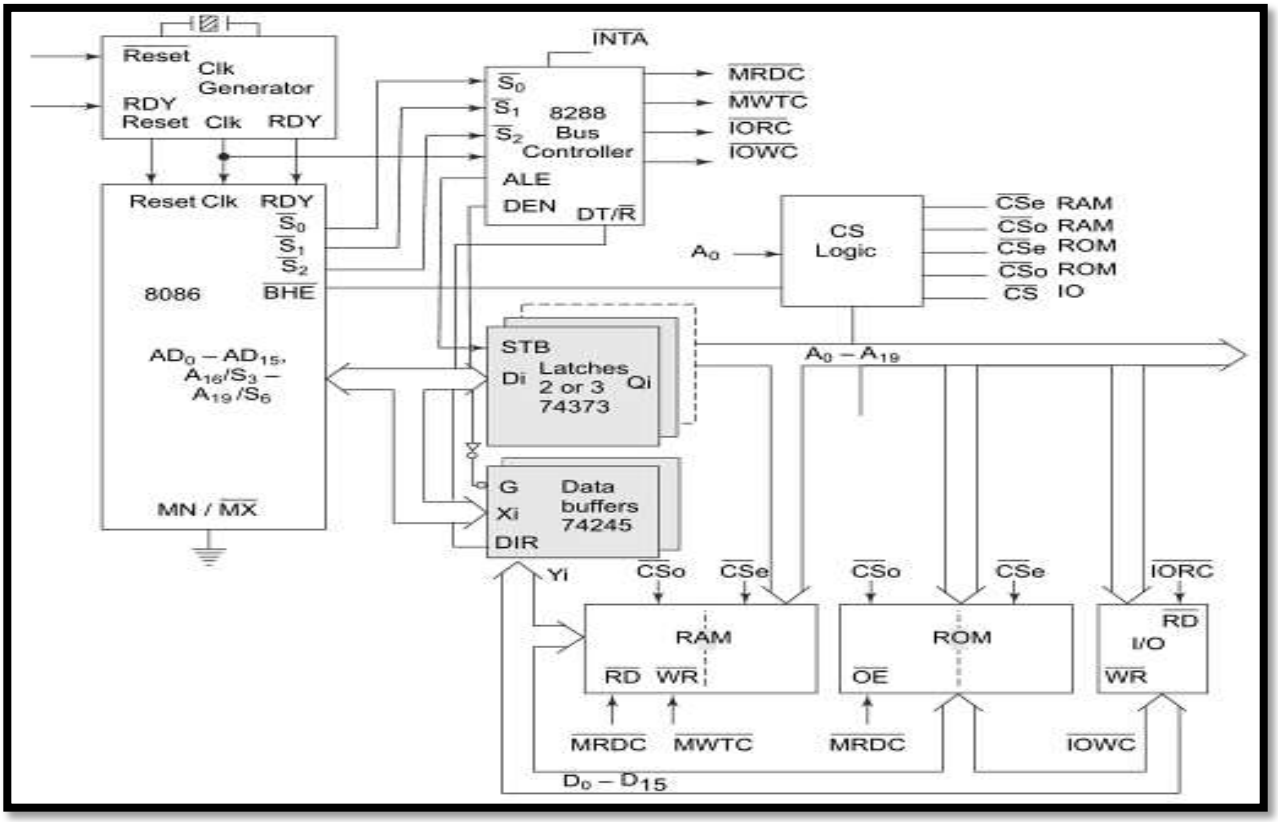
- **A write cycle** also begins with the assertion of ALE and the emission of the address. The M/IO' signal is again asserted to indicate a memory or I/O operation.
- In T_2, after sending the address in T_1, the processor sends the data to be written to the addressed location. The data remains on the bus until middle of T_4 state. The WR' becomes active at the beginning of T_2 (unlike RD' is somewhat delayed in T_2 to provide time for floating).
- The BHE' and A_0 signals are used to select the proper byte or bytes of memory or I/O word to be read or written.
- The M/IO', RD' and WR' signals indicate the types of data transfer as specified in Table

M/IO	RD	WR	Transfer Type
0	0	1	I/O read
0	1	0	I/O write
1	0	1	Memory read
1	1	0	Memory write

8086 Maximum mode system operation with timing diagrams:

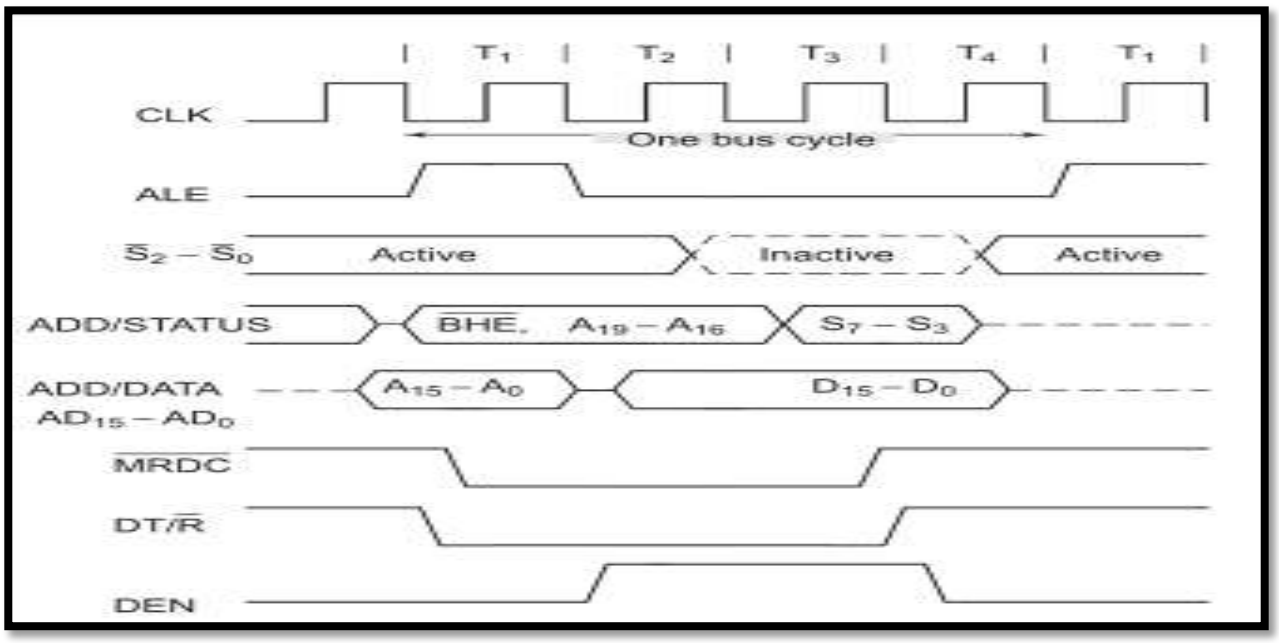
- In the maximum mode, the 8086 is operated by strapping the MN/MX' pin to ground. In this mode, the processor derives the status signals S2', S1' and S0'. Another chip called bus controller derives the control signals using this status information.
- In the maximum mode, there may be more than one microprocessor 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 IC8288, 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.
- 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.

Read cycle timing diagram for Maximum mode:



Write cycle timing diagram for Maximum mode: