UNIT 1

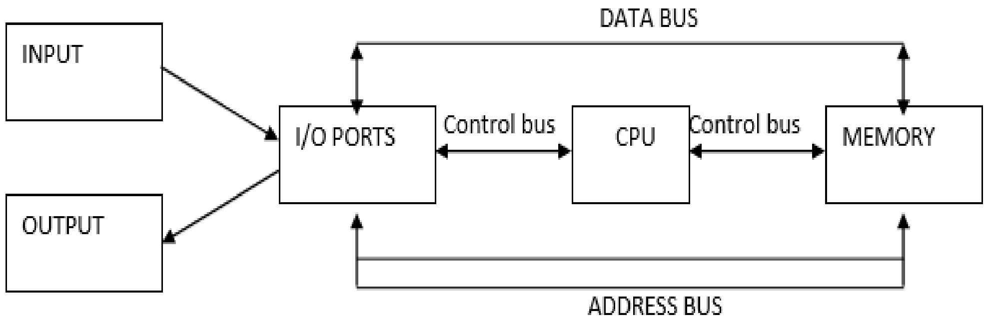
Unit 1 contents at a glance:

- 1. Architecture of 8086 microprocessor,
- 2. Register organization,
- 3. 8086 flag register and its functions,
- 4. addressing modes of 8086,
- 5. Pin diagram of 8086,
- 6. Minimum mode system operation,
- 7. Timing diagram.

Introduction to microprocessor

OVERVIEW OF A SIMPLE MICRO COMPUTER:

The major parts are the central processing unit or CPU, memory, and the input and output circuitry or I/O. Connecting these parts together are three sets of parallel lines called buses. The three buses are the address bus, the data bus, and the control bus.



Block diagram of simple computer or microcomputer.

i) MEMORY: The memory section usually consists of a mixture of RAM and ROM. It may also have magnetic floppy disks, magnetic hard disks, or laser optical disks. Memory has two purposes. The first purpose is to store the binary codes for the sequence of instructions you want the computer to carry out. When you write a computer program, what you are really doing is just writing a sequential list of instructions for the computer. The second purpose of the memory is to store the binary-coded data with which the computer is going to be working.

ii) INPUT/OUTPUT: The input/output or I/O section allows the computer to take in data from the outside world or send data to the outside world. These allow the user and the computer to communicate with each other. The actual physical devices used to interface the computer buses to external systems are often called ports.

iii) CPU: The central processing unit or CPU controls the operation of the computer. It fetches binary-coded instruction of the computer. It fetches binary-coded instructions from memory, decodes the instructions into a series of simple actions, and carries out these actions. The CPU contains an arithmetic logic unit, or ALU. Which can perform add, subtract, OR, AND, invert, or exclusive-OR operations on binary words when instructed to do so. The CPU also contains an address counter which is used to hold the address of the next instruction or data to be fetched from memory, general-purpose registers which are used for temporary storage of binary data, and circuitry which generates the control bus signals.

iv) ADDRESS BUS: The address bus consists of 16, 20, 24, or more parallel signal lines. On these lines the CPU sends out the address of the memory location that is to be written to or read from. The number of address lines determines the number of memory locations that the CPU can address. If the CPU has N address lines then it can directly address 2 to the N power memory locations.

v) DATA BUS: The data bus consists of 8, 16, 32 or more parallel signal lines. As indicated by the double-ended arrows on the data bus line, the data bus lines are bi-directional. This means that the CPU can read data in on these lines from memory or from a port as well as send data out on these lines to memory location or to a port. Many devices in a system will have their outputs connected to the data bus, but the outputs of only one device at a time will be enabled.

vi) CONTROL BUS: The control bus consists of 4-10 parallel signal lines. The CPU sends out signals on the control bus to enable the outputs of addressed memory devices or port devices. Typical control bus signals are memory read, memory write, I/O read, and I/O writer. To read a byte of data from a memory location, for example, the CPU sends out the address of the desired byte on the address bus and then sends out a memory read signal on the control bus.

OPERATIONS OF SIMPLE COMPUTER:

1. A simple computer CPU fetches instructions or reads data from memory (reads memory) by sending out an address on the address bus and a memory read signal on the control bus. The addressed instruction or data is sent from memory to the CPU on the data bus.

2. The CPU can write data in RAM by sending out an address on the address bus, sending out the data to be written on the data bus, and sending out a memory write signal on the control bus.

3. To read data from a port, the CPU sends the port address out on the address bus and sends an I/O read signal on the control bus. Data from the port comes into the CPU on the data bus.

4. To write data to a port, the CPU sends out the port address on the address bus, sends the data to be written to the port out on the data bus, and sends an I/O write signal out on the control bus.

5. A microcomputer fetches each program instruction in sequence, decodes the instruction, and executes it.

MICROPROCESSOR: A silicon chip that contains a CPU. A microprocessor (sometimes abbreviated µP) is a digital electronic component with miniaturized transistors on a single semiconductor integrated circuit (IC).

Microprocessors also control the logic of almost all digital devices, from clock radios to fuel-injection systems for automobiles.

Microprocessor 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 and processes data according to instructions, and provides result as output.

<u>**Three basic characteristics differentiate microprocessors**</u>

1. **Instruction set:** The set of instructions that the microprocessor can execute.
2. **Bandwidth:** The number of bits processed in a single instruction.
3. **Clock speed:** Given in megahertz (MHz), the clock speed determines how many instructions per second the processor can execute.
4. **Memory addressing capability**

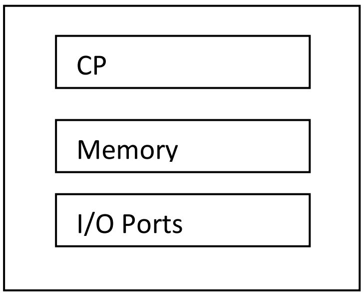
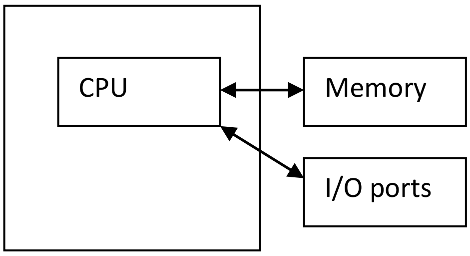
   The higher the values of above, the more powerful the CPU.

Microprocessors are categorized in additional as: RISC or CISC.

Microcontroller:

Controller: A device that controls the transfer of data from a computer to a peripheral device and vice versa.
Microcontroller is a highly integrated chip that contains all the components comprising a controller.

Typically this includes a CPU, RAM, some form of ROM, I/O ports, and timers.

Microcontroller	Microprocessor
It is a single chip	**It is a CPU**
Consists memory, I/o ports	**Memory , I/O ports to be connected externally**
	

Disadvantages of 8085 microprocessor:

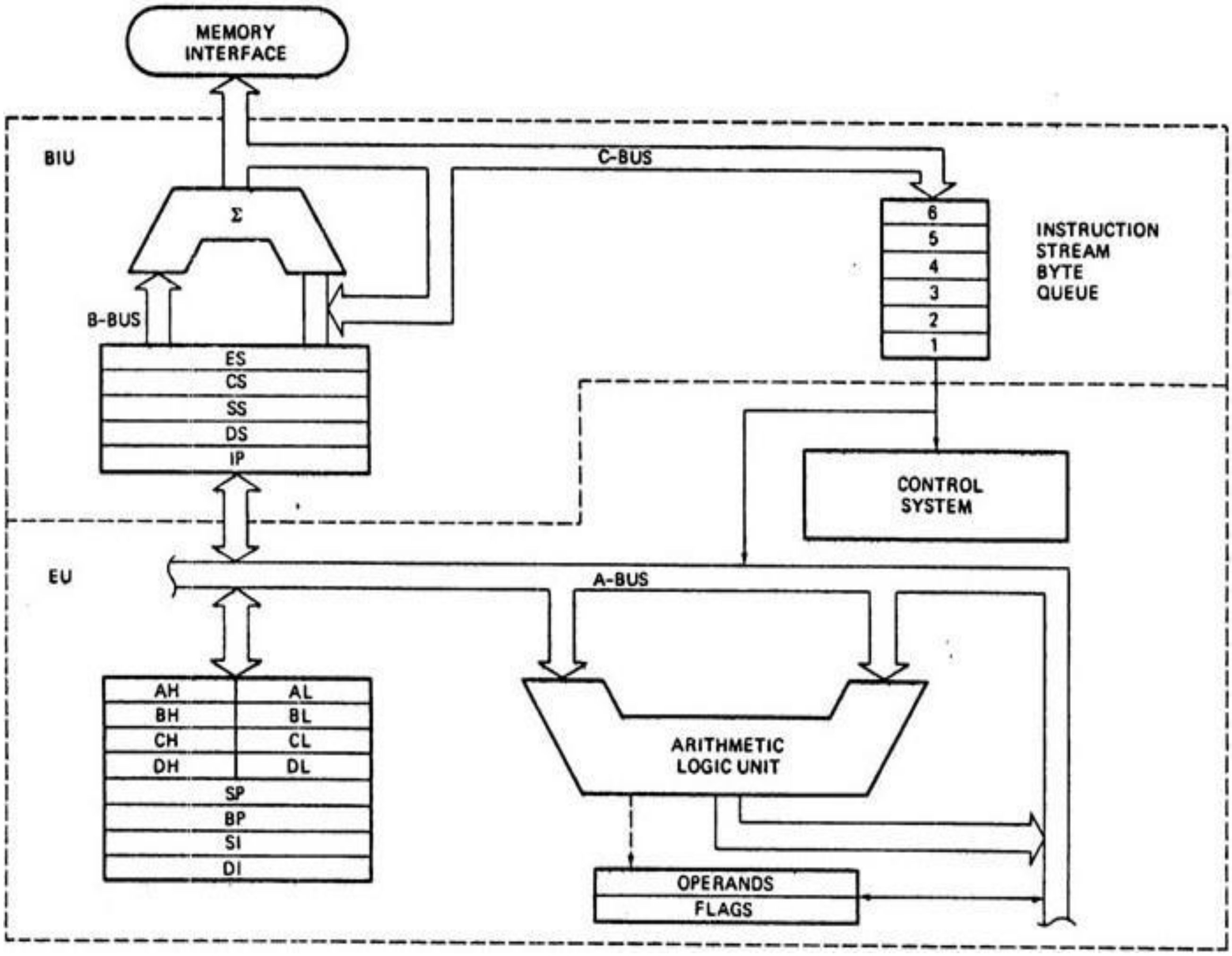
- 8085 is an 8 bit-functionally complete one. It has following disadvantages:
  - Slow speed
  - No protection or parallel execution
  - Limited memory addressing capacity
  - Complex instruction set

The above 8085 disadvantages are overcome by 8086 microprocessor.

1. 8086 specifications:

1. It is 16-bit microprocessor
2. It has 20 bit address bus and can access up to 2^{20} 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 AD_0-AD_{15} & A_{16}-A_{19}
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

Architecture of 8086 Microprocessor:



Explanation for 8086 architecture:

The 8086 architecture has two parts:

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

- The BIU performs all bus operations such as instruction fetching, reading and writing operands for memory and calculating the addresses of the memory operands. The instruction bytes are transferred to the instruction queue.
- EU executes instructions from the instruction system byte queue.
- Both units operate asynchronously to give the 8086 an overlapping instruction fetch and execution mechanism which is called as pipelining. This results in efficient use of the system bus and system performance.
- BIU contains Instruction queue, Segment registers, Instruction pointer, and Address adder.
- EU contains Control circuitry, Instruction decoder, ALU, Pointer and Index register, Flag register.

Bus Interface Unit:

- Responsible for performing external bus operations
- The functions of BIU are:
  - Instruction Fetch
  - Instruction Queuing
  - Operand Fetch & storage
  - Address Relocation
  - Bus control
  - Address adder – fetching of physical address of next instruction(CS+IP)

The Execution Unit:

The 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

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

8086 HAS PIPELINING ARCHITECTURE:

a. 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.

b. The BIU stores these pre-fetched bytes in a first-in-first-out register set called a queue.

c. 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.

d. 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.

e. Fetching the next instruction while the current instruction executes is called **pipelining**.

SEGMENTATION- The memory Addressing Scheme For 8086:

Address bus size=20 bit

Total addressable locations=2^{20} =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 (2^{20}*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)

Advantages of Memory segmentation

1. In 8086 microprocessor architecture, It permits its programmer to access 1MB memory even though address associated with the Instruction is 16 bit.
2. Instruction, data, sack of a program can be more than 64KB memory in 8086 microprocessor architecture
3. 8086 microprocessor architecture permits separate memory area for instruction, data and stack. So one program can work on different sets of data.
4. This method is very useful during the multitasking in 8086 microprocessor architecture

Memory Organization in 8086:

• The memory address space of the 8086-based microcomputers has different logical and physical organizations.

• Logically, memory is implemented as a single 1M × 8 memory chunk. The byte-wide storage locations are assigned consecutive addresses over the range from 0000016 through FFFFF16.

• Physically, memory is implemented as two independent 512Kbyte banks: **the low (even) bank and the high (odd) bank.** Data bytes associated with an even address (0000016, 0000216, etc.) reside in the low bank, and those with odd addresses (0000116, 0000316, etc.) reside in the high bank.

• Address bits A1 through A19 select the storage location that is to be accessed. They are applied to both banks in parallel. A0 and bank high enable (BHE) are used as bank-select signals.

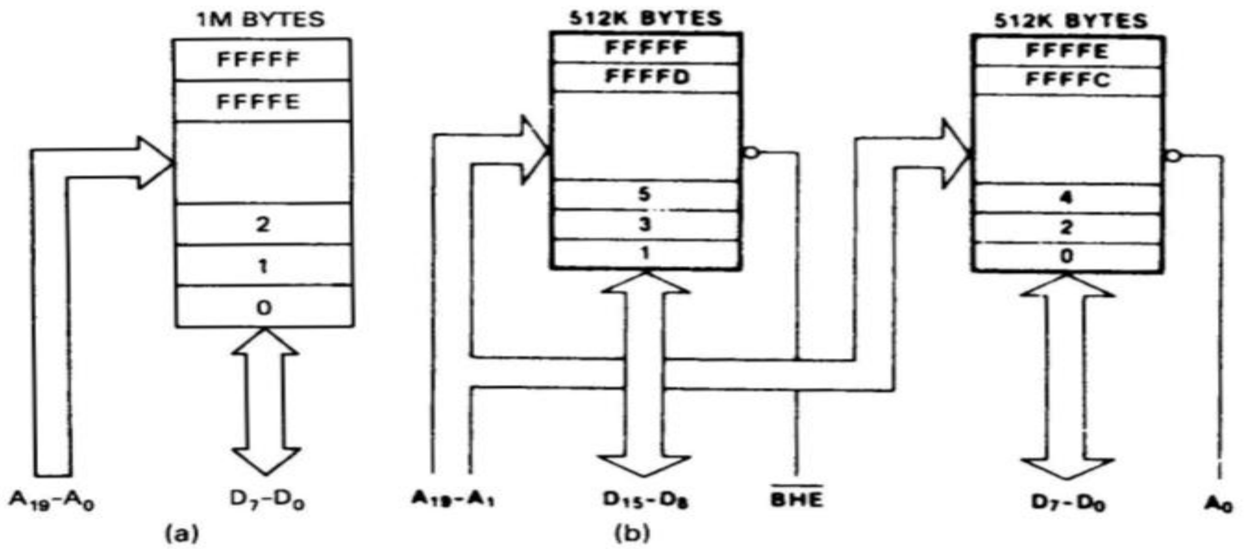


Figure 8-15 (a) Logical memory organization, and (b) Physical memory organization (high and low memory banks) of the 8086 microprocessor.

• Each of the memory banks provides half of the 8086's 16-bit data bus. The lower bank transfers bytes of data over data lines D0 through D7, while data transfers for a high bank use D8 through D15.

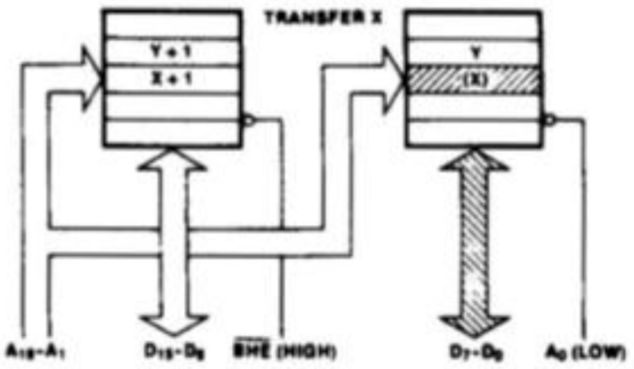
• **The 8086 microprocessor accesses memory as follows:**

• Fig. 8-17(a) shows how a byte-memory operation is performed to address X, an even-addressed storage location. A0 is set to logic 0 to enable the low bank of memory and BHE to logic 1 to disable the high bank. Data are transferred to or from the lower bank over data bus lines D0 through D7.

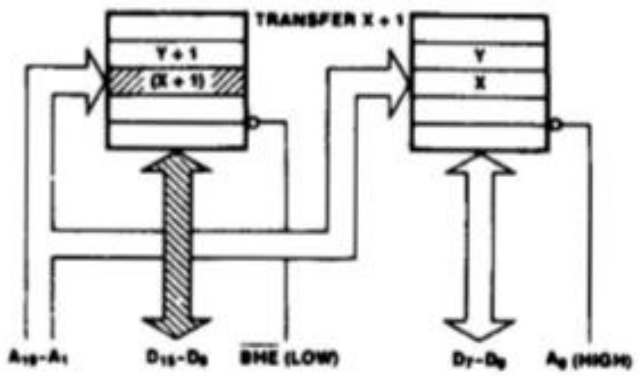
• Fig. 8-17(b) shows how a byte-memory operation is performed to an odd- addressed storage location such as X + 1. A0 is set to logic 1 and BHE to logic 0. This enables the high bank of memory and disables the low bank. Data are transferred over bus lines D8 through D15. D8 represents the LSB.

• Fig. 8-17(c) illustrates how an aligned word (at even address X) is accessed. Both the high and low banks are accessed at the same time. Both A0 and BHE are set to 0. This 16-bit word is transferred over the complete data bus D0 through D15 in just one bus cycle.

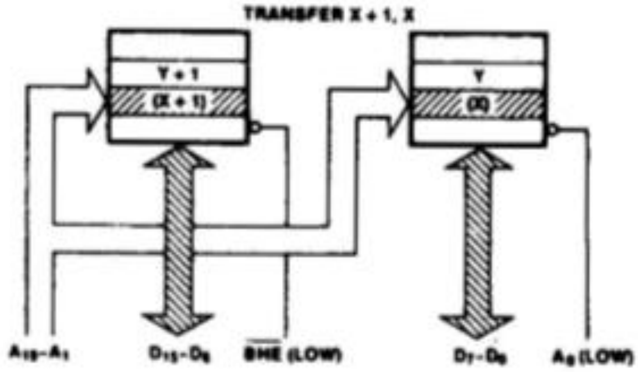
• Fig. 8-17(d) illustrates how a misaligned word (at address X + 1) is accessed. Two bus cycles are needed. During the first bus cycle, the byte of the word located at address X + 1 in the high bank is accessed over D8 through D15. Even though the data transfer uses data lines D8 through D15, to the processor it is the low byte of the addressed data word. In the second memory bus cycle, the even byte located at X + 2 in the low bank is accessed over bus lines D0 through D7.


TRANSFER X
Y + 1
X + 1
Y
(X)
A19-A1
D15-D8
BHE (HIGH)
D7-D0
A0 (LOW)

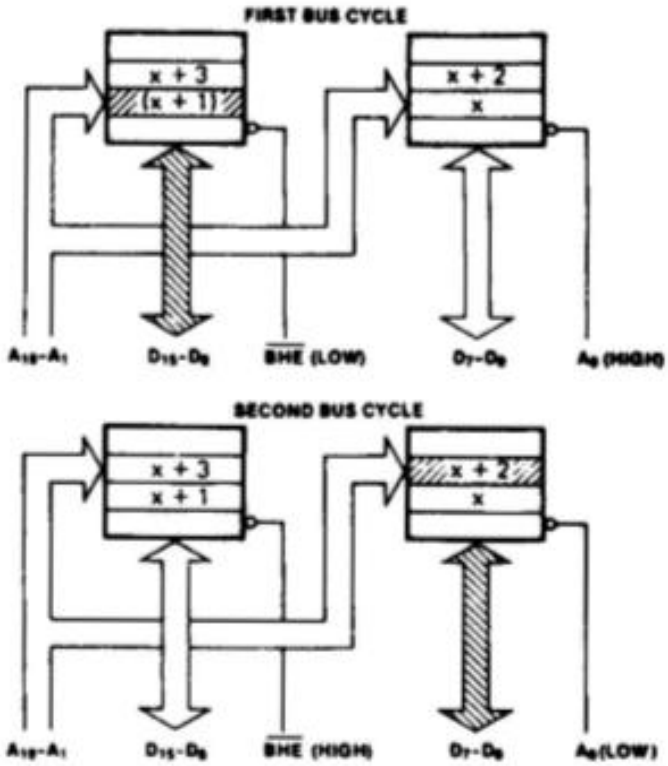
(a)


TRANSFER X + 1
Y + 1
(X + 1)
Y
X
A19-A1
D15-D8
BHE (LOW)
D7-D0
A0 (HIGH)

(b)


TRANSFER X + 1, X
Y + 1
(X + 1)
Y
(X)
A19-A1
D15-D8
BHE (LOW)
D7-D0
A0 (LOW)

(c)


FIRST BUS CYCLE
x + 3
(x + 1)
x + 2
x
A19-A1
D15-D8
BHE (LOW)
D7-D0
A0 (HIGH)
SECOND BUS CYCLE
x + 3
x + 1
x + 2
x
A19-A1
D15-D8
BHE (HIGH)
D7-D0
A0 (LOW)

(d)

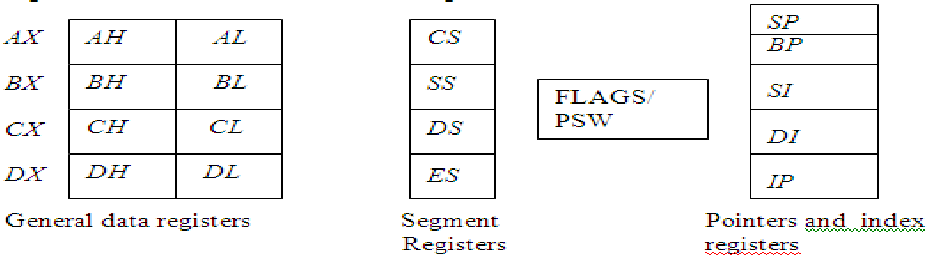
2. Register Organization:

All the registers are of 16 bit.

The 8086 registers are classified into the following types:

1. General Purpose Data Registers(AX, BX, CX, DX)
2. Segment Registers(CS, DS, ES, SS)
3. Pointers and Index Registers(IP, BP, SP)
4. Flag Registers(S,Z,P,C,T,I,D,AC,O)

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 Data Registers (AX, BX, CX, DX):

Here register is of 16 bit which is divided into two 8-bits halfs. One is higher 8 bits denoted by (AH or BH or CH or DH) and another is lower 8 bits denoted by (AL or BL or CL or DL).

- AX—used as 16 bit accumulator register for storing data(AH+AL)
- BX-gives offset storage address(BH+BL)
- CX-used as default counter in case of string and loop instructions(CH+CL)
- DX-used as General purpose register (DH+DL)

Note:

1. The registers *AX, BX, CX* and *DX* are the general purpose 16-bit registers.
2. *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.
3. 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*.
4. The register *CX* is used default counter in case of string and loop instructions.
5. The register *BX* is used as offset storage for forming physical address in case of certain addressing modes.
6. *DX* register is a general purpose register which may be used as an implicit operand or destination in case of a few instructions.

7. The EU has eight general-purpose registers, labeled AH, AL, BH, BL, CH, CL, DH and DL. These registers can be used individually for temporary storage of 8-bit data.

8. The AL register is also called the **accumulator**.

9. Certain pairs of these general-purpose registers can be used together to store 16-bit data words. The acceptable register pairs are AH and AL, BH and BL, CH and CL, and DH and DL.

10. The AH-AL pair is referred to as the AX register, the BH-BL pair is referred to as the BX register, the CH-CL pair is referred to as the CX register and the DH-DL pair is referred to as the DX register.

Segment Registers (CS, DS, ES, SS):

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 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.

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

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

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

<u>Stack Segment Register (SS):</u> 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.

Note: The advantage of this scheme is that in place of maintaining a 20-bit register for a physical address, the processor just maintains two 16-bit registers which is within the memory capacity of the machine.

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

1. 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.
   - He pointer register *SP* contains offset within the stack segment.
2. 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.
3. The register *SI* is used to store the offset of source data in data segment.
4. The register *DI* is used to store the offset of destination in data or extra segment.
5. The index registers are particularly useful for string manipulation.

Flag Register:

The 8086 flag register contents indicate the results of computation in the *ALU*. It also contains some flag bits to control the *CPU* operations.

Flag register is a 16 bit.

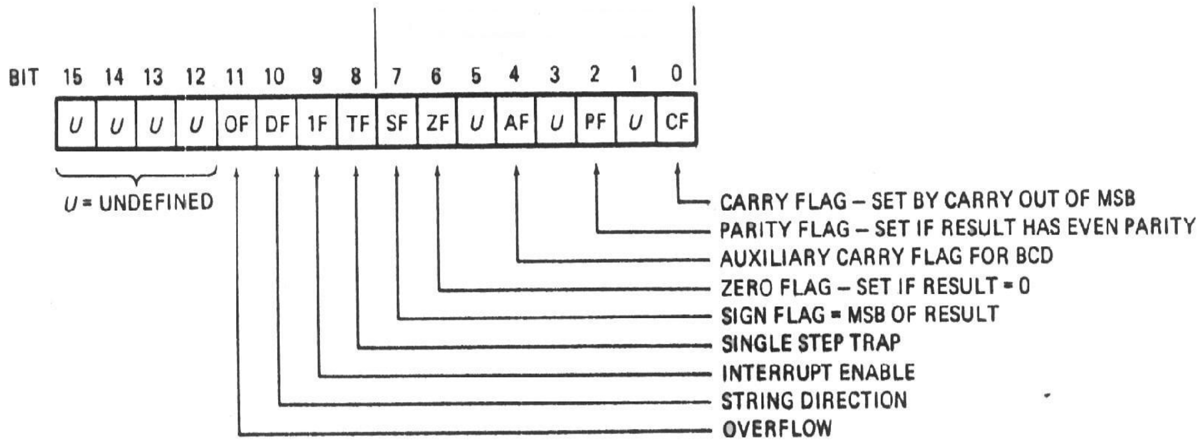
3. 8086 Flag Register and its Functions:

- 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:



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

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

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

***C*- 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.

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

***I*- Interrupt Flag:** If this flag is set, the maskable interrupt 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 incrementing mode.

***AC*-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.

***O*- 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, then the overflow will be set.

4.The 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 same 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

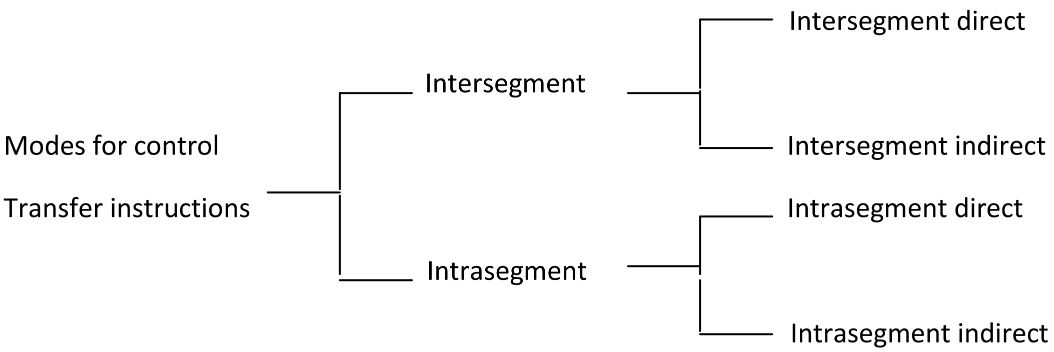
10H * DS + [BX] + [SI] + 50H

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

____

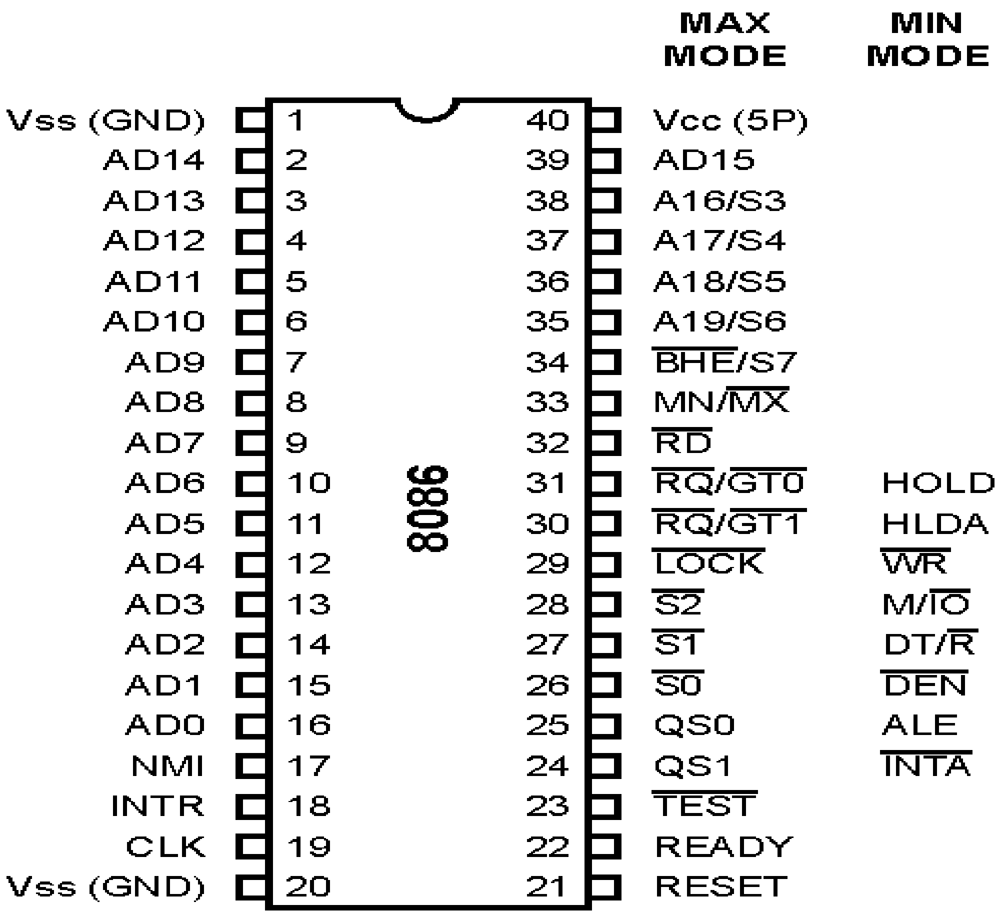
1A000H – Effective address

____

5. 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 below is **pin diagram for 8086:**



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.

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

$\overline{LOCK}$: The lock output is used to lock peripherals off the system, ie, 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 $\overline{RQ}/\overline{GT1}$

6. 8086 MINIMUM MODE SYSTEM OPERATION:

Minimum mode system comprises the following components:

1. clock generator and wait state generator

2. 2 or 3 latches(as address bus is 20 bit, 2 or 3 latches are required)

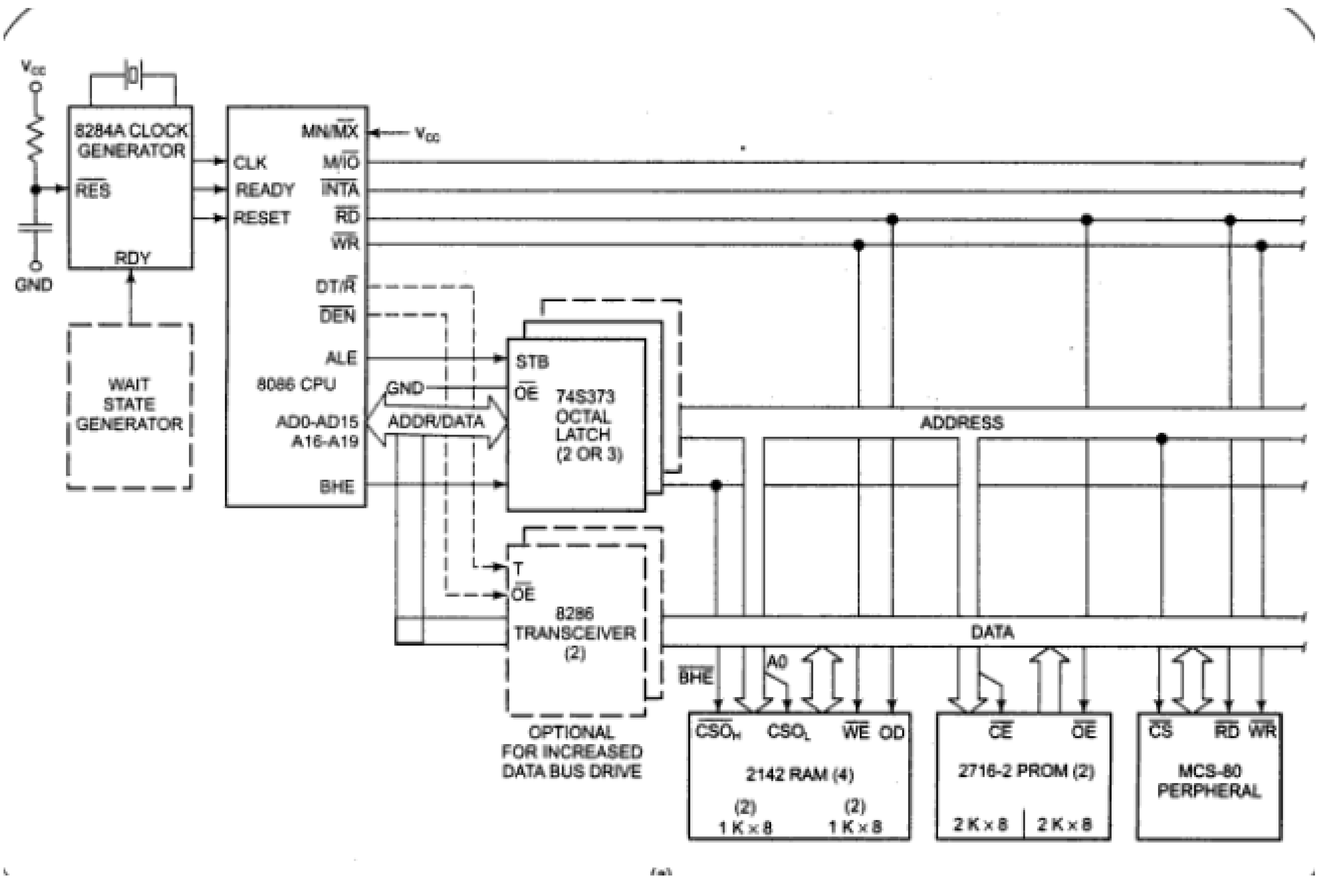
3. Transceivers(data buffers)

4. memory (RAM, EPROM for storing programs)

5. I/O

6.8086 processor

8086 minimum mode block diagram:



Explanation:

1. The control bus consists of signals like M/IO, RD and WR signals.

During a read operation from memory or port, RD signal will be asserted.

During a write operation to memory or port, WR signal will be asserted.

During memory read or write operation, M/IO signal will be high.

During port operations, M/IO signal will be low.

2.Latches: During the accessing of memory for any operation, 8086 send lower 16 bits of address on data bus. External Latches grabs this address and stores it. To strobe this address, 8086 sends out ALE signal on latches outputs. Once address is stored in latches, now the bus is used for data transfer purpose.

3.transceivers or data buffers: If we connect multiple devices (more devices) to processor on the data bus lines, 8086 cannot supply enough current drive to charge(store data) and discharge(removing data) the circuits capacitance fast enough. So we use Transceivers which are high current drive buffers to store data and discharge fast enough.

4. The Data Transmit / Receive signal (DT/R) sets the direction in which data will pass through the buffers.

When DT/R - high, then data flows from 8086 processor to ROM, RAM or ports.

When DT/R-low, then data flows from RAM, ROM or ports to 8086 processor.

5. The 8086 asserts the DEN signal to enable the three state outputs on data bus buffers at the appropriate time in an operation.

6. 8284A clock generator:It uses a crystal to produce the stable frequency clock signals which steps the 8086 through execution of constructions in an order manner. CLOCK Generator also synchronizes the RESET and READY signal with the clock so that these signals are applied to 8086 at the proper times. When the RESET input is asserted, the 8086 goes to FFFF0H address to get its next instruction.

8086 BUS ACTIVITIES DURING A READ AND WRITE MACHINE CYCLE:(TIMING DIAGRAMS)

State: one cycle of the clock is called state.

Machine cycle(made up of states): The basic microprocessor operation such as reading a byte from memory or writing a byte to a port is called machine cycle.

Instruction cycle(made up of machine cycles): The time required for microprocessor to fetch and execute an entire instruction is called Instruction cycle.

Bus cycle consists of 4 clock states or signals T1, T2, T3 and T4.

During period T1,

o The 8086 outputs the 20-bit address of the memory location to be accessed on its multiplexed address/data bus. BHE is also output along with the address during T1.

o At the same time a pulse is also produced at ALE. The trailing edge or the high level of this pulse is used to latch the address in external circuitry.

o Signal M/IO is set to logic 1 and signal DT/R is set to the 0 logic level and both are maintained throughout all four periods of the bus cycle.

• **Beginning with period T2,**

o Status bits S3 through S6 are output on the upper four address bus lines. This status information is maintained through periods T3 and T4.

o On the other hand, address/data bus lines AD0 through AD7 are put in the high-Z state during T2.

o Late in period T2, RD is switched to logic 0. This indicates to the memory subsystem that a read cycle is in progress. DEN is switched to logic 0 to enable external circuitry to allow the data to move from memory onto the microprocessor's data bus.

• **During period T3,**

o The memory must provide valid data during T3 and maintain it until after the processor terminates the read operation. The data read by the 8086 microprocessor can be carried over all 16 data bus lines.

• **During T4,**

o The 8086 switches RD to the inactive 1 logic level to terminate the read operation. DEN returns to its inactive logic level late during T4 to disable the external circuitry.

memory write cycle of the 8086:

• **During period T1,**

o The address along with BHE are output and latched with the ALE pulse.

o M/IO is set to logic 1 to indicate a memory cycle.

o However, this time DT/R is switched to logic 1. This signals external circuits that the 8086 is going to transmit data over the bus.

• **Beginning with period T2,**

o WR is switched to logic 0 telling the memory subsystem that a write operation is to follow.

o The 8086 puts the data on the bus late in T2 and maintains the data valid through T4. Data will be carried over all 16 data bus lines.

o DEN enables the external circuitry to provide a path for data from the processor to the memory.

7. Timing Diagrams for read and write cycles of 8086:

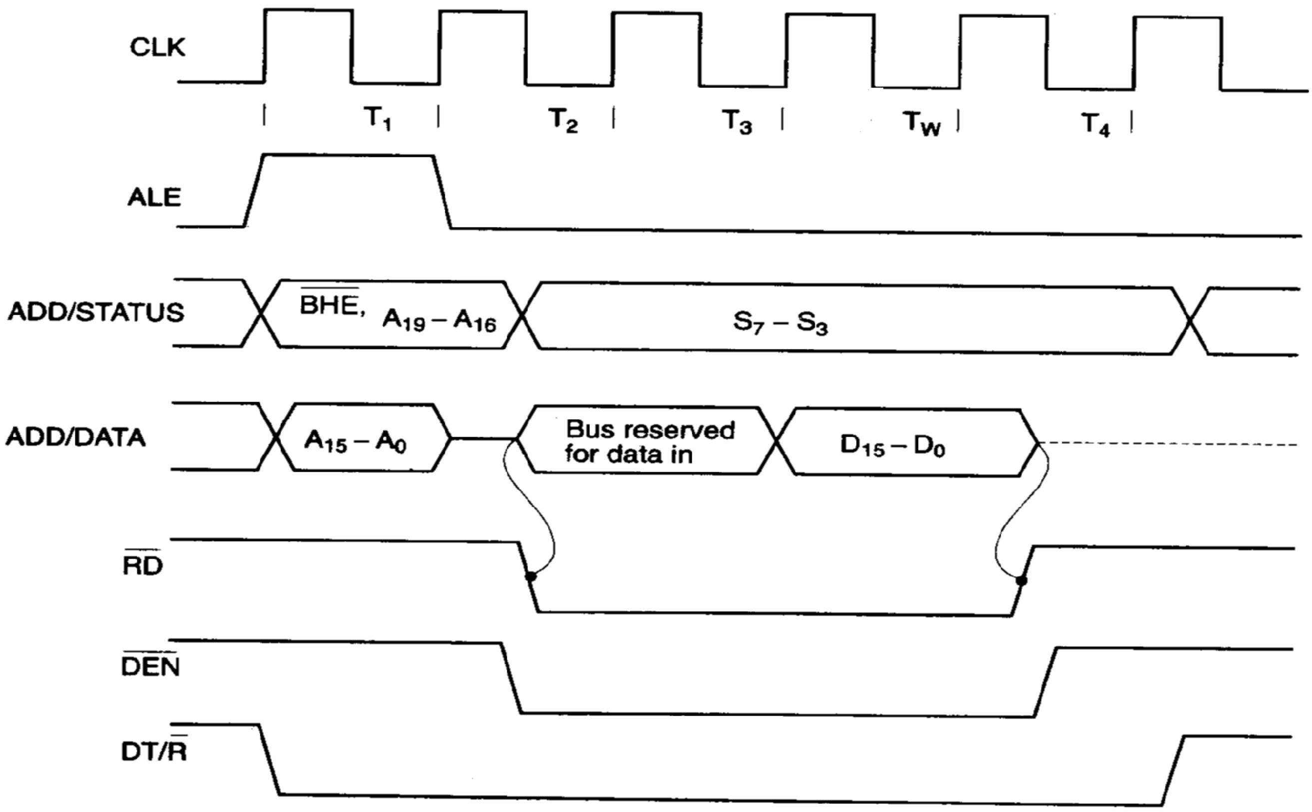


Fig. 1.9(a) *Read Cycle Timing Diagram for Minimum Mode*

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/$\overline{IO}$ 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.

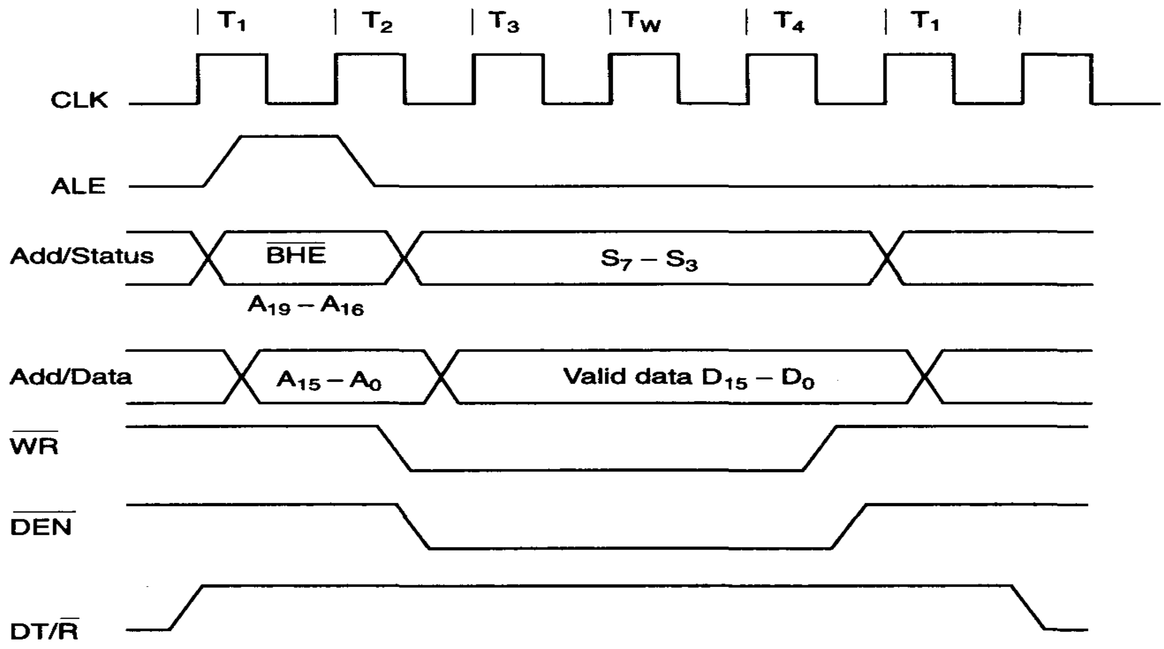


Fig. 1.9(b) ***Write Cycle Timing Diagram for Minimum Operation***

A write cycle also begins with the assertion of ALE and the emission of the address. The $M/\overline{IO}$ 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 $\overline{WR}$ becomes active at the beginning of T_2 (unlike $\overline{RD}$ is somewhat delayed in T_2 to provide time for floating).

The $\overline{BHE}$ 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 $M/\overline{IO}$, $\overline{RD}$ and $\overline{WR}$ signals indicate the types of data transfer