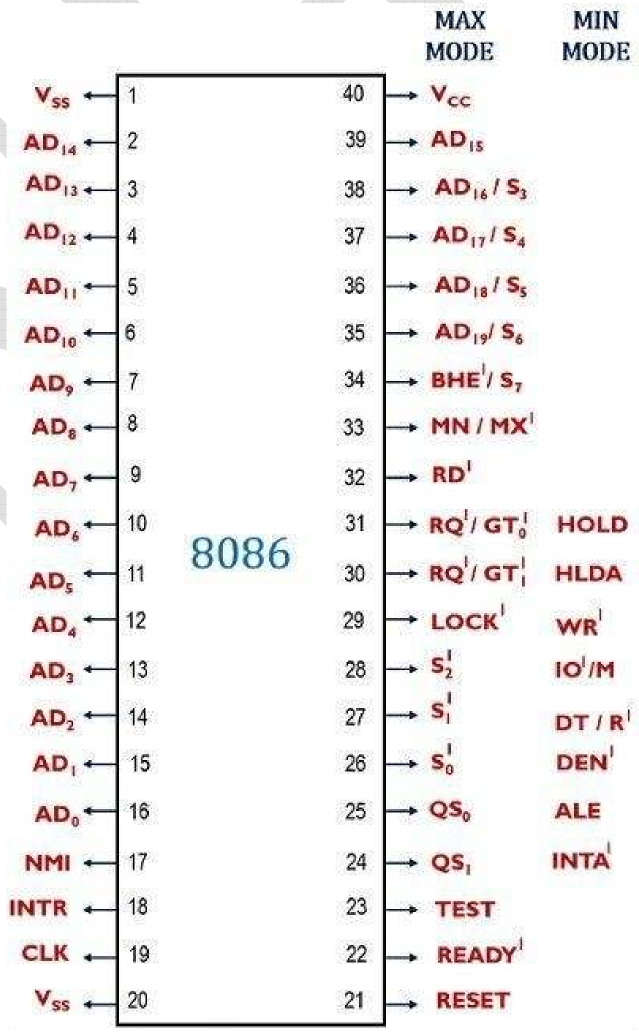
Unit II- 8086-16 bit Microprocessor

Definition: 8086 is a 16-bit microprocessor and was designed in 1978 by Intel. Unlike, 8085, an 8086 microprocessor has **20-bit address bus**. Thus, is able to access 2^{20} i.e., 1 MB address in the memory.

As we know that a microprocessor performs arithmetic and logic operations. And an 8086 microprocessor is able to perform these operations with 16-bit data in one cycle. Hence is a **16-bit microprocessor**. Thus the size of the data bus is 16-bit as it can carry 16-bit data at a time. The architecture of 8086 microprocessor, is very much different from that of 8085 microprocessor.

Pin Diagram and Description of 8086 Microprocessor

The 8086 microprocessor also contains **40 pins** dual in line. The 8086 to have better performance, operates in 2 modes that are minimum and maximum mode. The minimum mode is a single processor configuration while the maximum mode is a multiple processor configuration. Due to this reason, in the 40 pin IC of 8086 microprocessor, 8 pins i.e., pin numbered from 24 to 32 are assigned different configurations separately according to the two modes. Here, in the figure below, it is clear that from pin number 24 to 32, we have shown the different configuration for minimum and maximum mode. However, excluding these 8 pins, the rest 32 pins are the same for both minimum as well as maximum mode.



Pin description of 8086 Microprocessor

V_{CC} – ***Pin number 40*** – At this pin, the external power supply of **+ 5V** is provided to the processor.

V_{SS} – ***Pin number 1 and 20*** – These two pins acts as the ground. This pin directs the extra current of the microprocessor to ground.

AD_0 – AD_{15} – ***Pin number 2 to 16 and 39*** – These are the multiplexed address and data bus.

We know that the 8086 microprocessor has 20-bit address bus and 16-bit data bus. So, the 16 lines of the address and data bus are multiplexed together so as to reduce the number of lines inside the IC.

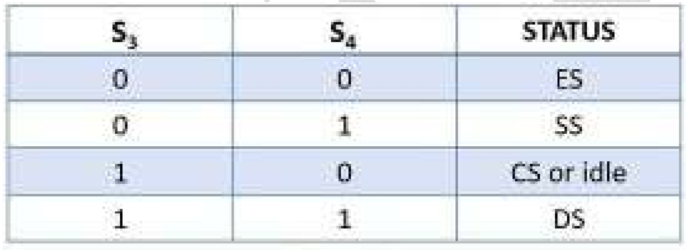
We are aware of the fact that at a time either address or data will be transmitted by the bus. So, at a particular time only either the address or the data bus will be enabled from the multiplexed buses.

A_{16}/S_3, A_{17}/S_4, A_{18}/S_5 and $A_{19}S_6$ – ***Pin number 35 to 38*** – Out of 20 address bits, 4 are present in the multiplexed form with the status signals. In the case of memory operations, these pins act as an address bus and contain the memory address of any particular instruction or data.

However, from I/O operations these pins are low that shows the status of the processor.

Basically, the signal at S_3 and S_4 show that which segment is currently accessed by the microprocessor among the four segments present in it.

The table below will show the encoding of S_3 and S_4:



S_3	S_4	STATUS
0	0	ES
0	1	SS
1	0	CS or idle
1	1	DS

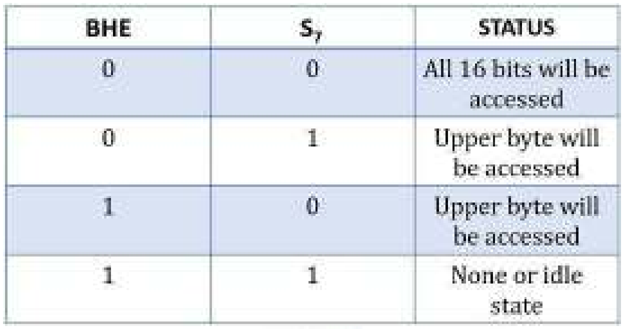
Also, S_5, when enabled, shows the presence of an interrupts in the microprocessor. So, basically, it serves as an **interrupt flag**.

The signal at S_6 shows the status of the bus master for the current operation. More simply we can say, whether the 8086 is the bus master or any other proficient device is acting as the bus master.

When 0 is present as the signal at this pin then it indicates the 8086 is holding the access of the bus otherwise it is high i.e., 1.

BHE' / S_7 – ***Pin number 34*** – BHE is an acronym for Bus High Enable. The combination of the BHE signal and S_7 status informs about the existence of the data on the bus. Also, different combinations show whether the bus is containing overall 16 bit, upper byte or lower byte of the data.

The table below represents the status for the signal at this pin:



BHE	S_7	STATUS
0	0	All 16 bits will be accessed
0	1	Upper byte will be accessed
1	0	Upper byte will be accessed
1	1	None or idle state

MN/MX' – ***Pin number 33*** –The status at this particular pin shows whether the processor is operating in the minimum mode or maximum mode.

A signal 0 at this pin informs that the 8086 is operating in maximum mode i.e., multiple processors. While signal 1 shows the operation under minimum mode i.e., single processor.

RD' – ***Pin number 32*** – An active low signal at this pin shows that the microprocessor is performing read operation with either memory or I/O devices.

CLK – ***Pin number 19*** – A signal at this pin provides the timing to the internal operations that are being executed inside the microprocessor.

NMI – ***Pin number 17*** – NMI is Non-maskable interrupt. These are basically uncontrollable interrupts generated inside the processor. When an NMI occurs, then an interrupt service routine is generated by the interrupt vector table.

TEST – ***Pin number 23*** – This pin basically shows the wait instruction. Whenever a low signal at this pin occurs then the processing inside the processor continues. As against, in case of the high signal, the processor has to wait for the disabling of this pin.

INTR – ***Pin number 18*** – INTR stands for an interrupt request. The processor after each clock cycle samples the INTR and if the signal at this pin is found to be high then the processor controls that interrupt internally.

READY – ***Pin number 22*** – This signal is used by the peripherals and memory devices in order to show the readiness for the next operation.

RESET – ***Pin number 21*** – Whenever this pin is enabled then it resets the processor and other devices connected to the system by immediately terminating the recent task.

Pins in Minimum mode

INTA' – ***Pin number 24*** – It is an interrupt acknowledge pin. Whenever an INTR signal is generated, then the microprocessor generates INTA signal, as a response to that interrupt.

ALE – ***Pin number 25*** – ALE is an abbreviation for address latch enable. Whenever an address is present in the multiplexed address and data bus, then the microprocessor enables this pin.

This is done to inform the peripherals and memory devices about fetching of the data or instruction at that memory location.

DEN' – ***Pin number 26*** – DEN is used for data enable. This is an active low pin that means whenever a 0 is present at this pin then the transceiver gets enabled and it separates the data from the multiplexed address and data bus.

DT/R' – ***Pin number 27*** – This pin is used to show whether the data is getting transmitted or is received. A high signal at this pin provides the information regarding the transmission of data. While a low indicates reception of data.

M/IO' – ***Pin number 28*** – This pin indicates whether the processor is performing an operation with memory or I/O devices. Whenever a high is present at this pin then it shows the operation is carried out through the memory. While a low signal shows operation through I/O devices.

WR' – ***Pin number 29*** – An active low signal at this pin indicates that the processor is performing write operation from either memory or I/O devices.

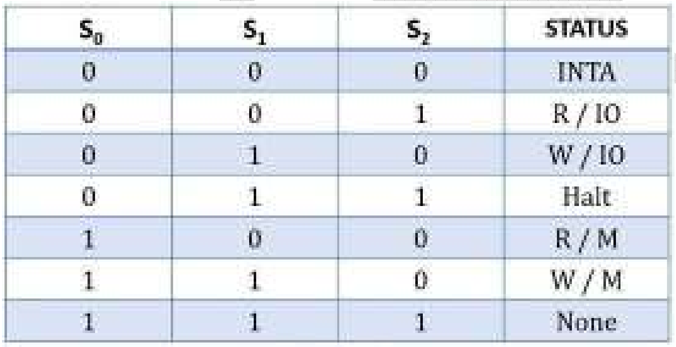
HOLD – ***Pin number 31*** – When an external device enables this pin then the processor stops accessing the buses immediately after the recent task gets over.

HLDA – ***Pin number 30*** – This pin is used as a response pin for the hold request. Once request for accessing the buses is produced by an external entity. Then the microprocessor acknowledges the device that its request will be considered once it gets over by the current operation.

Pins in Maximum mode

S_0', S_1' and S_2' – ***Pin number 26 to 28*** – These are basically 3 status pins and are active low. This means that if the status at all the 3 pins is 0 then it shows that multiple interrupts are to be handled in maximum mode.

The table below is representing the status of the processor in different combinations:



S_0	S_1	S_2	STATUS
0	0	0	INTA
0	0	1	R / IO
0	1	0	W / IO
0	1	1	Halt
1	0	0	R / M
1	1	0	W / M
1	1	1	None

QS_0 and QS_1 – ***Pin number 24 and 25*** – These two pins indicate the status of the 6-byte pre-fetch queue present in the architecture of 8086.

QS_0	QS_1	STATUS
0	0	No operation
0	1	First byte from queue
1	0	Empty queue
1	1	Subsequent byte from queue

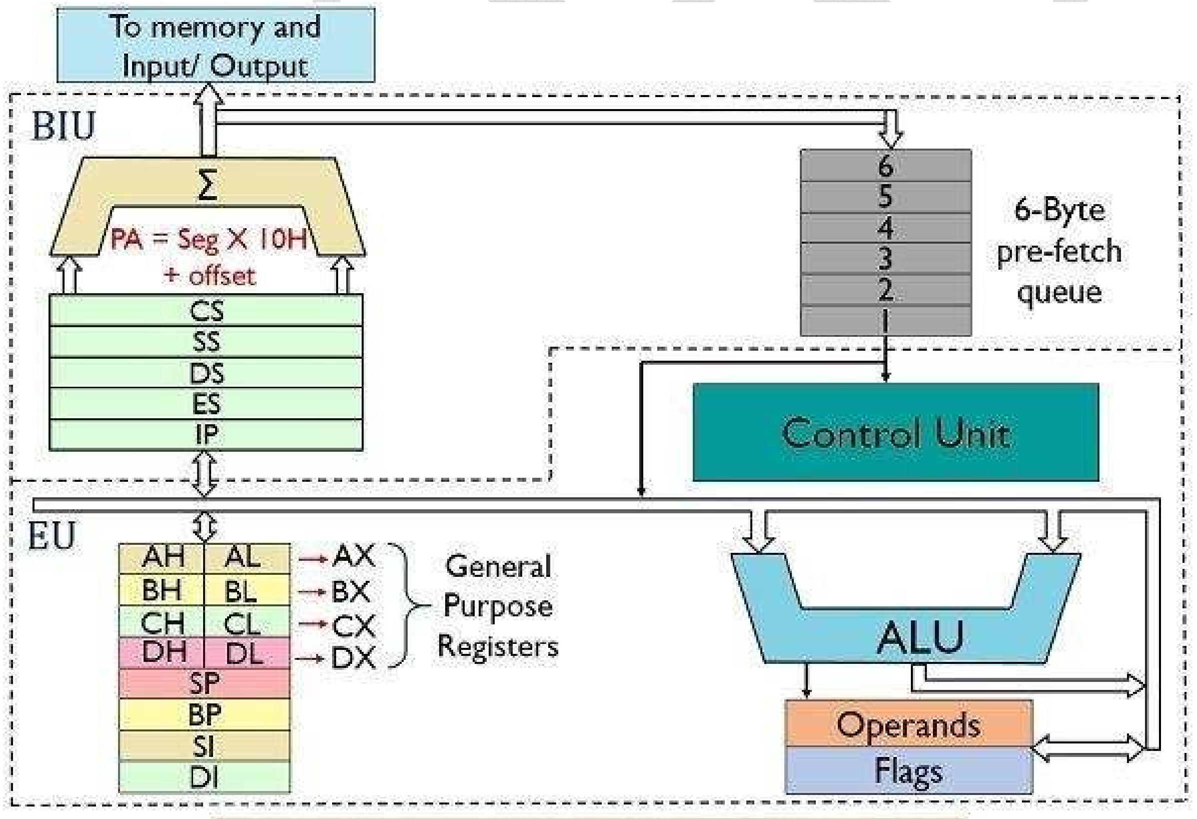
LOCK’ – ***Pin number 29*** –This pin is involved in maximum mode operation. So, basically, when a single processor is accessing the buses and peripherals then it locks the resources being used by it. So, that no other entity can access it until the recent processor frees it.

RQ’/ GT_0‘ and RQ’/ GT_1‘ – ***Pin number 30 and 31*** – Due to the involvement of multiple processors, these pins indicate the request and grant permission for accessing the buses, memory and peripherals.

Block Diagram of 8086 Microprocessor

The architecture of 8086 microprocessor is composed of 2 major units, the BIU i.e., Bus Interface Unit and EU i.e., Execution Unit.

The figure below shows the block diagram of the architectural representation of the 8086 microprocessor:



Block Diagram of 8086 Microprocessor

Bus Interface Unit (BIU)

The Bus Interface Unit (BIU) manages the data, address and control buses.

The BIU functions in such a way that it:

- Fetches the sequenced instruction from the memory,
- Finds the physical address of that location in the memory where the instruction is stored and
- Manages the 6-byte pre-fetch queue where the pipelined instructions are stored.

An 8086 microprocessor exhibits a property of pipelining the instructions in a queue while performing decoding and execution of the previous instruction. This saves the processor time of operation by a large amount. This pipelining is done in a **6-byte queue**.

Also, the BIU contains **4 segment registers**. Each segment register is of 16-bit. The segments are present in the memory and these registers hold the address of all the segments. These registers are as follows:

1.Code segment register: It is a 16-bit register and holds the address of the instruction or program stored in the code segment of the memory.

Also, the IP in the block diagram is the instruction pointer which is a default register that is used by the processor in order to get the desired instruction. The **IP contains the offset address** of the next byte that is to be taken from the code segment.

2. Stack segment register: The stack segment register provides the starting address of stack segment in the memory. Like in stack pointer, PUSH and POP operations are used in this segment to give and take the data to/from it.

3. Data segment register: It holds the address of the data segment. The data segment stores the data in the memory whose address is present in this 16-bit register.

4. Extra segment register: Here the starting address of the extra segment is present. This register basically contains the address of the string data.

It is to be noteworthy that the physical address of the instruction is achieved by combining the segment address with that of the offset address.

6-byte pre-fetch queue: This queue is used in 8086 in order to perform pipelining. As at the time of decoding and execution of the instruction in EU, the BIU fetches the sequential upcoming instructions and stores it in this queue.

The size of this queue is 6-byte. This means at maximum a 6-byte instruction can be stored in this queue. The queue exhibits **FIFO** behaviour first in first out.

Execution Unit (EU)

The Execution Unit (EU) performs the decoding and execution of the instructions that are being fetched from the desired memory location.

Control Unit:

Like the timing and control unit in 8085 microprocessor, the control unit in 8086 microprocessor produces control signal after decoding the opcode to inform the general purpose register to release the value stored in it. And it also signals the ALU to perform the desired operation.

ALU:

The arithmetic and logic unit carries out the logical tasks according to the signal generated by the CU. The result of the operation is stored in the desired register.

Flag:

Like in 8085, here also the flag register holds the status of the result generated by the ALU. It has several flags that show the different conditions of the result.

Operand:

It is a temporary register and is used by the processor to hold the temporary values at the time of operation.

The reason behind two separate sections for BIU and EU in the architecture of 8086 is to perform fetching and decoding-executing simultaneously.

Working of 8086 Microprocessor

In the previous section, we have discussed the operation of various sections of the BIU and EU. Now in this section, we will have a look at the overall processing cycle of 8086 microprocessor. So, basically, when an instruction is to be fetched from the memory, then firstly its physical address must be calculated and this is done at the BIU.

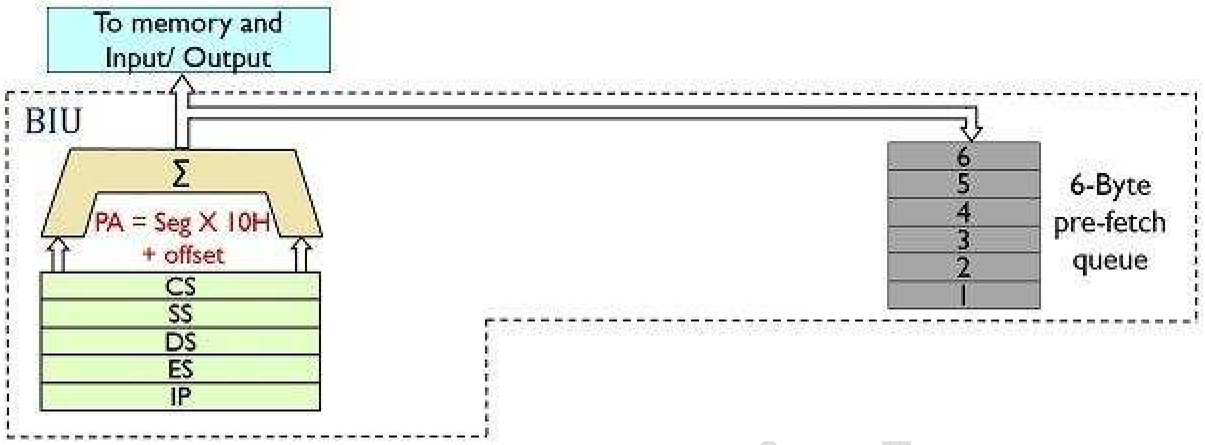
Physical address generation :

The physical address of an instruction is given as:

PA = Segment address X 10 + Offset

For example: Suppose the segment address is 2000 H and the offset address is 4356 H. So, the generated physical address is **24356 H**.

Here, the code segment register provides the base address of the code segment which is combined with the offset address.

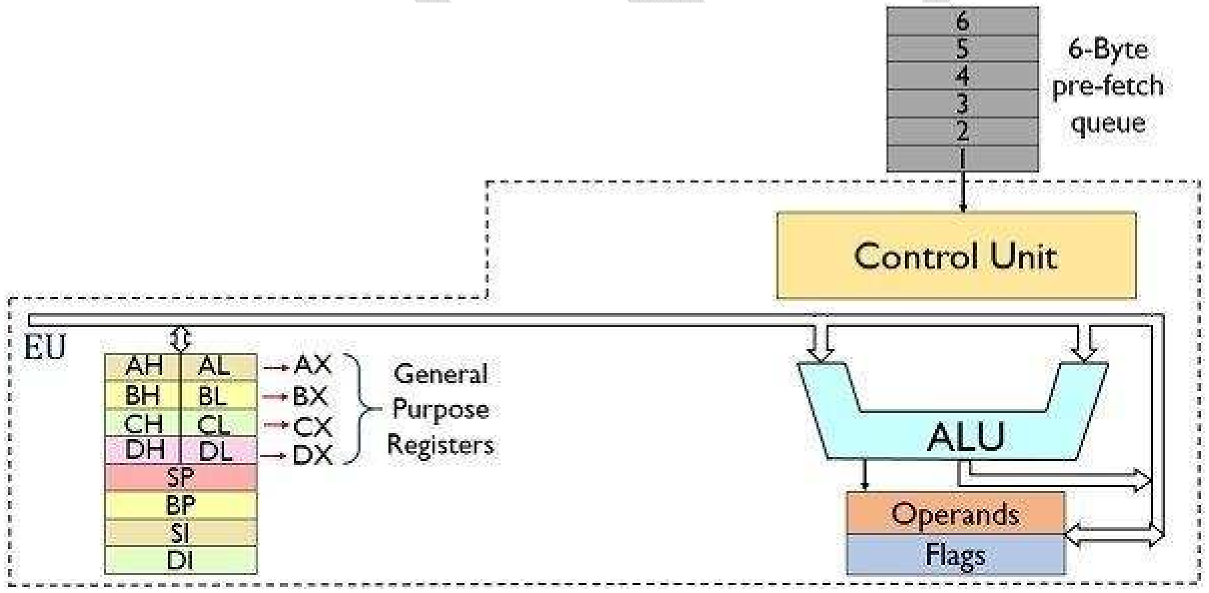


The code segment contains the instructions. Each time an instruction is fetched the offset address inside the code segment gets incremented. So, once the physical address of an instruction is calculated by the BIU of the processor, it sends the memory location by the address bus to the memory.

Further, the desired instruction at that memory location which is present in the form of the opcode is fetched by the microprocessor through the data bus.

Suppose the instruction is **ADD BL, CL**. But, inside the memory, it will be in the form of an opcode. So, this opcode is sent to the control unit.

The control unit decodes the opcode and generates control signals that inform the BL and CL register to release the value stored in it. Also, it signals the ALU to perform the ADD operation on that particular data.



It is noteworthy that in any instruction, like ADD BL, CL. BL denotes the destination of the result of the add operation.

This clearly shows that whatever, the operation is performed its result must be stored in the first register i.e., BL for this particular example.

Let us take another example: Consider an instruction, **ADD CL, 05H**.

This means that the operand which is 05H is to be added with the data present in the CL register and is stored in that particular register i.e., CL.

In such condition, the operand is not provided to the control unit as only the opcode is required to be decoded by the CU. Hence the operand is directly provided to the ALU.

Also, the status of this result is stored in the flag register. So, whenever, ALU carries out an operation, it simultaneously generates the result as well as its status.

It is to be noteworthy that in BIU, pipelining fails whenever there is branching in the instruction. This is because generally instructions are present in a sequential manner. But, sometimes the instructions are required to be executed unsequentially.

However, in the queue, the instructions are stored sequentially. So, in case there exist a need for any random instruction to be decoded. The opcode stored in the queue will become invalid and must be cleared at that particular time.

Memory Segmentation in 8086 Microprocessor

Segmentation is the process in which the main memory of the computer is logically divided into different segments and each segment has its own base address. It is basically used to enhance the speed of execution of the computer system, so that the processor is able to fetch and execute the data from the memory easily and fast.

Need for Segmentation –

The Bus Interface Unit (BIU) contains four 16 bit special purpose registers (mentioned below) called as Segment Registers.

- **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.

The number of address lines in 8086 is 20, 8086 BIU will send 20bit address, so as to access one of the 1MB memory locations. The four segment registers actually contain the upper 16 bits of the starting addresses of the four memory segments of 64 KB each with which the 8086 is working at that instant of time. A segment is a logical unit of memory that may be up to 64 kilobytes long. Each segment is made up of contiguous memory locations. It is an independent, separately addressable unit. Starting address will always be changing. It will not be fixed.

Note that the 8086 does not work the whole 1MB memory at any given time. However, it works only with four 64KB segments within the whole 1MB memory.

Types Of Segmentation –

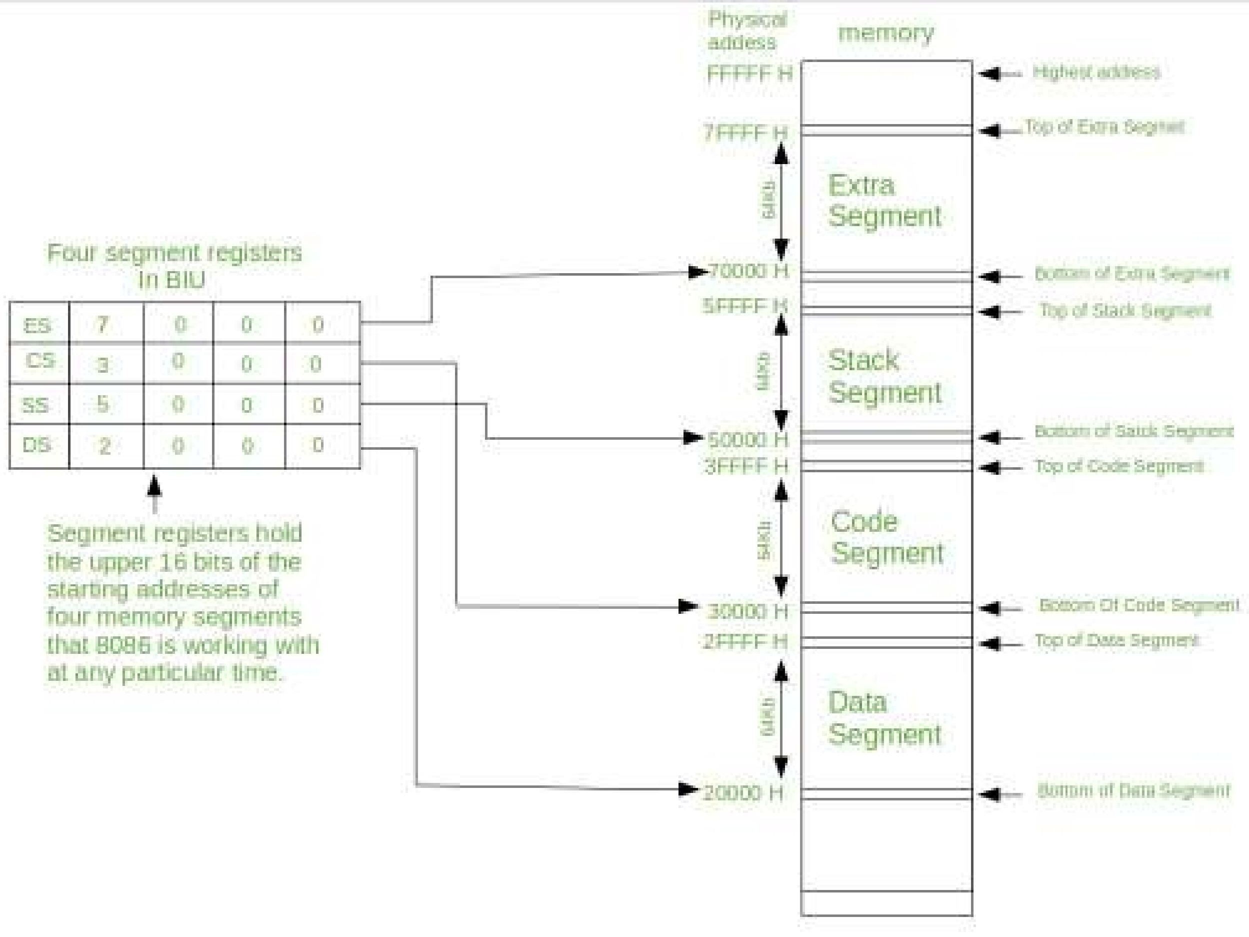
1. **Overlapping Segment –** A segment starts at a particular address and its maximum size can go up to 64kilobytes. But if another segment starts along with this 64kilobytes location of the first segment, then the two are said to be *Overlapping Segment*.
2. **Non-Overlapped Segment –** A segment starts at a particular address and its maximum size can go up to 64kilobytes. But if another segment starts before this 64kilobytes location of the first segment, then the two segments are said to be *Non-Overlapped Segment*.

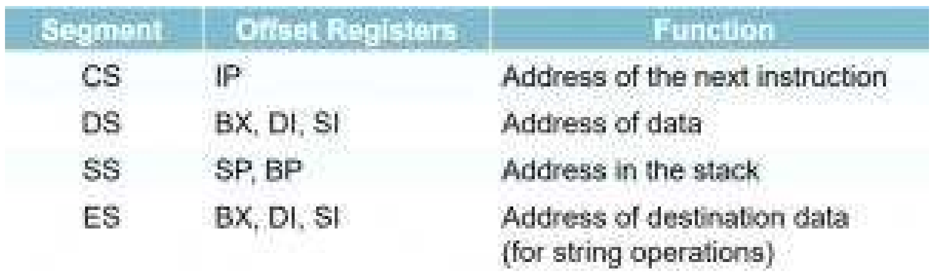
Rules of Segmentation

Segmentation process follows some rules as follows:

- The starting address of a segment should be such that it can be evenly divided by 16.
- Minimum size of a segment can be 16 bytes and the maximum can be 64 kB.

Below is the one way of positioning four 64 kilobyte segments within the 1M byte memory space of an 8086.





Segment	Offset Registers	Function
CS	IP	Address of the next instruction
DS	BX, DI, SI	Address of data
SS	SP, BP	Address in the stack
ES	BX, DI, SI	Address of destination data (for string operations)

Advantages of the Segmentation

The main advantages of segmentation are as follows:

- It provides a powerful memory management mechanism.
- Data related or stack related operations can be performed in different segments.
- Code related operation can be done in separate code segments.
- It allows to processes to easily share data.
- It allows to extend the address ability of the processor, i.e. segmentation allows the use of 16 bit registers to give an addressing capability of 1 Megabytes. Without segmentation, it would require 20 bit registers.
- It is possible to enhance the memory size of code data or stack segments beyond 64 KB by allotting more than one segment for each area.