- The number of address lines in 8086 is 20. So the 8086 BIU will send out a 20
address in order to access one of the 1,048,576 or 1 Mb memory locations.
- But it is interesting to note that the 8086 does not work the whole 1,048,576
(1M.byte) memory at any given time. However it works with only four 6
(64k-byte) segments within the whole 1 M-byte memory.
- The four segment registers actually hold (contain) the upper 16 bits of the star
addresses of the four memory segments of 64 k byte each with which the 8086
working at that instant of time.
- A word is any two consecutive bytes in memory. Word are stored in memory with
most significant byte at the higher memory address. They bytes are sto
sequentially from byte 0000H to byte FFFFH.
- Programs view memory space as a group of segments defined by the application.
- A segment is a logical unit of memory that may be upto 64 K bytes long.
- Each segment is made up of contiguous memory locations. It is independe
separately addressable unit.
- Note that these starting addresses will always be changing. They are not fix.
- This concept can be clearly understood by referring to Fig. 11.5.2.
- Fig. 11.5.2 shows one of the possible ways to position the four 64 k byte segments
within the 1-M byte memory space of 8086. There is no restriction on the locations
these segments in the memory.

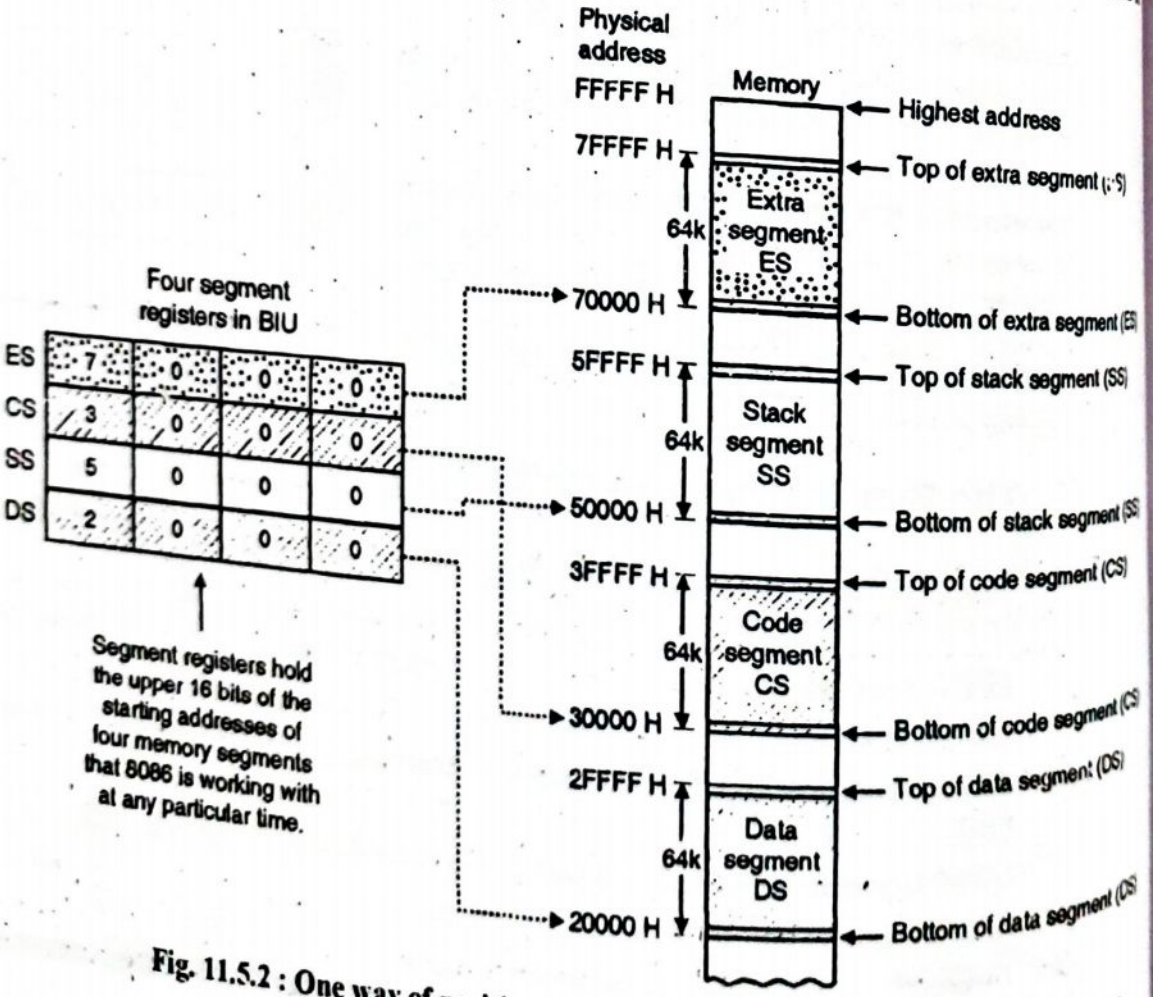

Fig. 11.5.2 : One way of positioning four 64 k byte segments within the 1 M byte memory space of an 8086

- Note that these segments can be separate from each other as shown in Fig. 11.5.2 or they can overlap.
- Note that the starting address or base address of the data segment is 20000H. The upper 16-bits of this i.e. 2000 are loaded into the data segment register (DS).
- Similarly upper 16 bits of the starting addresses of the other segments are stored into the corresponding segment registers.
- The segment overlapping usually takes place for small programs which do not need all the 64 k bytes in each segment.
- The segments can adjacent, disjoint, partially overlapping or fully overlapping.
- Fig. 11.5.3 shows another way of positioning segments within 1 MB memory space of 8086.

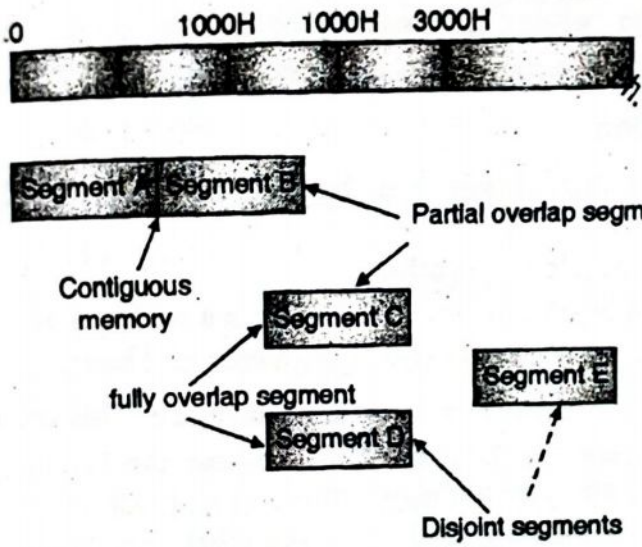

Fig. 11.5.3

- In the users program there can be many segments. But 8086 can deal with only four of them at any given time, because it has only four segment registers.
- Whenever the segment orientation is to be changed, we have to change the base addresses and load the upper 16 bits into the corresponding segment registers.
- Each time the CPU needs to generate a memory address, one of the segment registers is automatically chosen and its contents added to a logical address.
- For an instruction fetch, the code segment register is automatically added to the logical address to compute the value of the instruction address.
- For stack refrencing operations, either data or extra segment register is chosen as the base. The logical address may be made up of different types of values: it can be simply the immediate data value and a base register plus an index register.
- Generally the selection of DS or ES register is automatically done, though provisions exist to override this selection. Thus, any memory location can be addressed without changing the value of the segment base register.
- In system that use 64K or fewer bytes of memory for each memory area (code, stack, data and extra) the segment registers can be initialized to zero at the beginning of the program and then ignored, since zero plus a 16 bit offset yields a 16 bit address.

- In a system where the total amount of memory is 64 KB or less, it is possible to set all segments equal and have fully overlapping segments.
- Segment registers are very useful for large programming tasks that require isolation of program code from the data code or isolation of module data from the stack information etc.
- Segmentation makes it easily to build relocatable and re-entrant programs. In many cases the task of relocating a program (relocation means to ability for an the same program in several different areas of memory without changing addresses in the program itself. Simply requires moving the program code and then adjusting the code segment register to point to the base of the new code area.
- Since programs can be written for the 8086/8088 in which all branches and jumps are relative to the instruction pointer, it branches and jumps are relative to the instruction pointer, it does not matter what value is kept in the code segment register.
- Every application will define and use segments differently. The currently addressable segment override prefix provides a generous workspace; 64KB bytes for code, 64 Kbytes stack and 128 Kbytes of data storage.

11.5.3 Memory Segmentation

- Memory can be thought of as a vast collection of bytes. These bytes need to be organized in some efficient manner in order to be of any use.

11.5.4 Advantages of Segmentation

Q. Explain the advantages of segmentation.

The principle of segmentation discussed now has certain benefits. Some of them are as follows :

1. Segmentation provides a powerful memory management mechanism.
2. The segmented structure of 8086 memory space supports modular software design. It discourages huge monolithic programs. i.e. Among other things, It allows programmers to partition their programs into modules that operate independently of one another.
3. Segments provide a way to easily implement object-oriented programs.
4. Segments allow two processes to easily share data.
5. It allows you to extend the address ability of a processor i.e. segmentation allows the use of 16-bit registers to give an addressing capability of 1 M byte. Without segmentation, we would require 20-bit registers. In the case of the 8086, segmentation let Intel's designers extend the maximum addressable memory from 64K to one megabyte.
6. Segmentation makes it possible to separate the memory areas for stack, code and data.
7. It is possible to increase the memory size of code data or stack segments beyond 64 k bytes by allotting more than one segment for each area.
8. Segmentation makes it possible to write programs which are position independent or dynamically relocatable.

11.5.5 Code Segment (CS)

- Code segment (CS) is the part of memory from which the BIU fetches the instruction code bytes.
- The upper 16 bits of the starting address of code segment are held by the code segment (CS) register.
- But the memory address for the base of code segment is 20 bit long. So what about the lower 4-bits ? The answer is that the BIU always inserts zeros for the lowest 4-bits.
- So if the CS register contents are 3428 H then after adding zeros, the physical starting address of code segment becomes 34280 H.
- Therefore a segment will always start at an address with zeros in the lowest 4 bits.

Segment Base : The upper 16-bits of the starting address of a segment, stored in the segment register is called as the segment base.

11.5.6 Stack

- In Fig. 11.5.3 we have defined the stack segment from the memory location 50000 H to 5FFFF H.
- The stack is a section of memory which is reserved to store the addresses and data while executing a subroutine program.
- The stack segment (SS) register holds the upper 16 bits of the starting address of the stack area.

11.5.7 Instruction Pointer (IP) Register

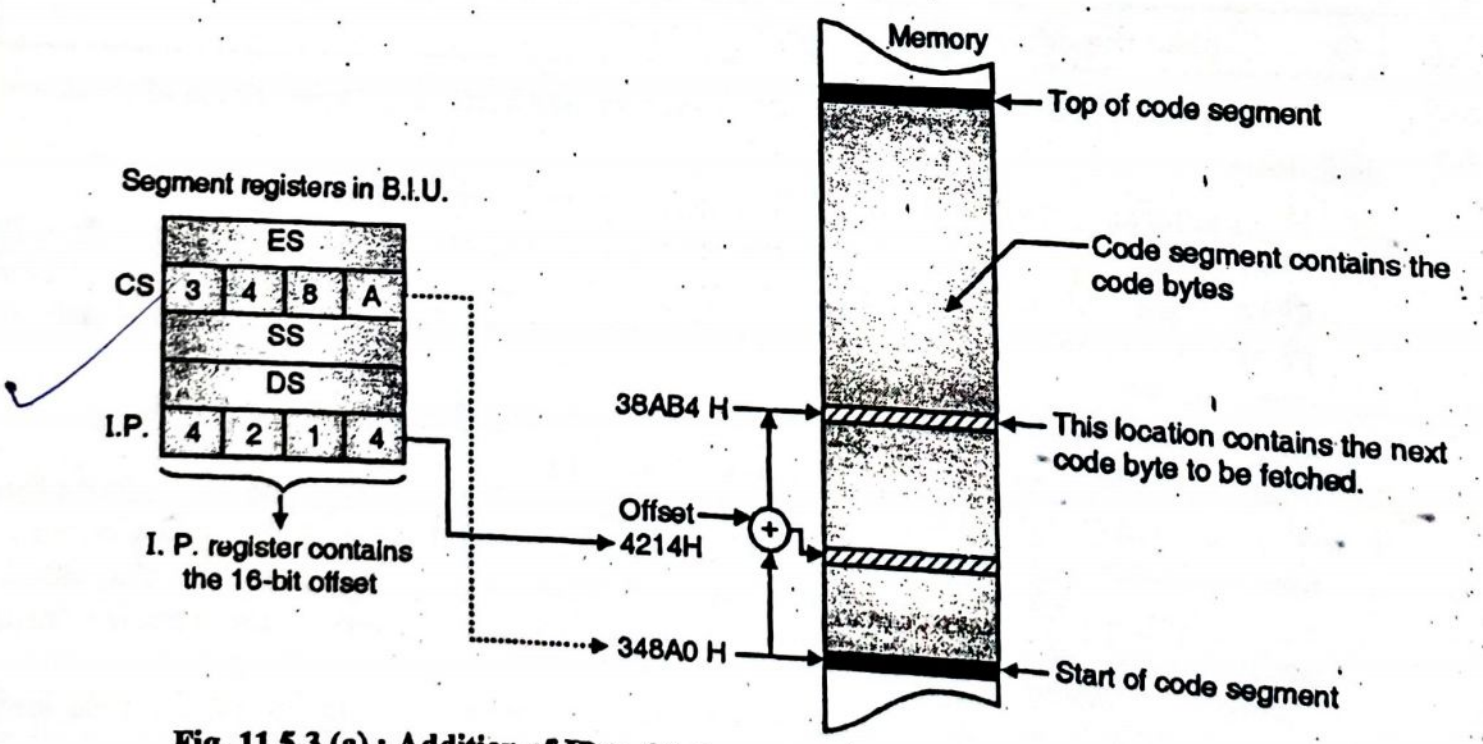

Fig. 11.5.3 (a) : Addition of IP to CS to produce the physical address of the code byte

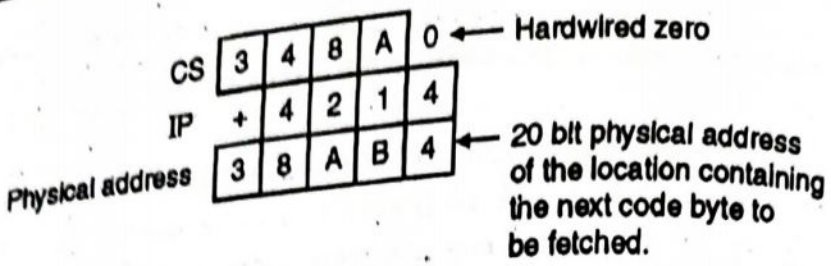

Fig. 11.5.3 (b) : Computation

- Another special purpose register in the BIU is the instruction pointer (IP) register.
- As discussed earlier the code segment (CS) register holds the upper 16 bits of the 20 bit starting address of the code segment. And code segment is the segment from which the BIU is currently fetching the instruction code bytes.
- The instruction pointer (IP) register holds the 16 bit address or **offset** of the next code byte within the code segment. This is illustrated in Fig. 11.5.3(a).

Rules of segmentation

1. If non-overlapping, there can be 16 segment in all, each of 64KB (since 1MB / 64KB = 16)
2. Segments can overlap each other
3. A segment can begin at any location that is a multiple of 10H
4. At any given time a maximum of 4 segments and hence 256KB can be accessed
5. The memory of 8086 is a wrap around memory. This means that once the address generated crosses the 1MB limit, it accesses the location at the top 00000H. For e.g. if CS = FFFF and IP = 0010. Physical address = CS*10H + IP = 100000H. The address lines in 8086 are only 20, so the MSB '1' is discarded and the location being accessed is 00000H.
6. In an instruction only the offset address in mentioned. The segment register is fixed for each of the pointer registers. Default segment register assignments are as follows :

Pointer Register	Default Segment Register
BX	DS
SI	DS
DI	DS (ES in case of string operations)
SP	SS
BP	SS
IP	CS

7. Using segment override prefix, one can change the above default segment register assignments

A Point to note is that the four segments need not point to different segments but can be same when

$$ES = CS = DS = SS$$

or can be partially overlapped as in figure 2.9 where the stack segment and extra segment are overlapped from the address 2C C C 0 H to 2 F F F F H.

As earlier discussed the addresses within a segment can range from 0 0 0 0 H to F F F F H. The address within the segment is called offset or logical address.

eg., CS	2 0 2 0 0	base address
IP	F 2 D 2	offset address
	2 F 4 d 2	physical address

Now the physical address is the address that is actually sent on the address bus (that is it is 20 bits wide). The offset or logical address is the offset from the location zero of a segment. This segment is the location pointed by the base address or the segment register.

Now consider the following example:

For extra segment

C 2 2 2 0 H	base address ES
2 3 2 3 H	offset BP
C 4 5 4 3 H	physical address

for Stack segment

B F 3 D 0 H	base address SS
5 1 7 3 H	Offset SP
C 4 5 4 3 H	physical address

Now here we observed that two different logical address point to the same memory address. This is due to the overlapping of two segments and should be avoided as here the program data and stack data values can be caused to be overwritten on each other resulting in wrong results.

2.2.3. ADVANTAGE OF MEMORY SEGMENTATION

Though you may at this find memory segmentation to be slightly complex but as you advance through this last section you shall find out about its advantages. However, here we shall discuss the advantages only superficially. The first advantage that memory segmentation has is that only 16 bit registers are required both to the store segment base address as well as offset address. This makes the internal circuitry easier to build as it removes the requirement for 20 bits register in case the linear addressing method is used.

The second advantage is relocatability. For example in a normal computer system time sharing process is used. Time sharing involves allocating a fixed amount of time for programs of different users. However, memory segmentation is a boon for time shared systems as all user program can be loaded into memory. The 8086 can then be made to work on a time sharing basis on each of these program by just reloading the segment address of each of these programs in the segment registers. Thus segmentation allows the programmer to separately use each of the program loaded in memory by just reloading the 8086 segment registers.

11.5.8 Physical Address Generation

Q. How does 8086 convert a logical address to physical address ?

- Let us now understand the generation of 20 bit physical address of the location in the code segment which contains the next code byte.
- The sequence of operation is as follows.

Physical Address Generation

Step 1 : The CS register contains the upper 16 bits of the starting address of the code segment.

∴ CS Register : | 3 | 4 | 8 | A | Segment base

↓

Step 2 : The BIU will automatically insert zeros for the lowest four bits of the segment base address to get the 20 bit physical address for the starting of code segment.

∴ Starting address of code segment : | 3 | 4 | 8 | A | 0 ← BIU adds this zero

↓

Step 3 : The I.P. register contains the offset or distance from this address. The offset here is 4214H.

∴ I.P. Register : | 4 | 2 | 1 | 4 |

↓

Step 4 : Add the starting address of code segment (20 bit) to the offset to get the physical address of the location containing the next code byte as follows :

Starting address of code segment →	3	4	8	A	0	← Hard wired 0
Offset in the I.P. Register → +		4	2	1	4	
Physical address of the location containing the next code byte →	3	8	A	B	4	

- This 20 bit physical address is then sent out by the BIU to fetch the next code byte stored at this location.

Alternative way to represent the physical address

- An alternative way of representing a 20 bit physical address is as follows :
- | Segment base | : | Offset |
- For example the 20 bit physical address in the above example is 348A : 4214.