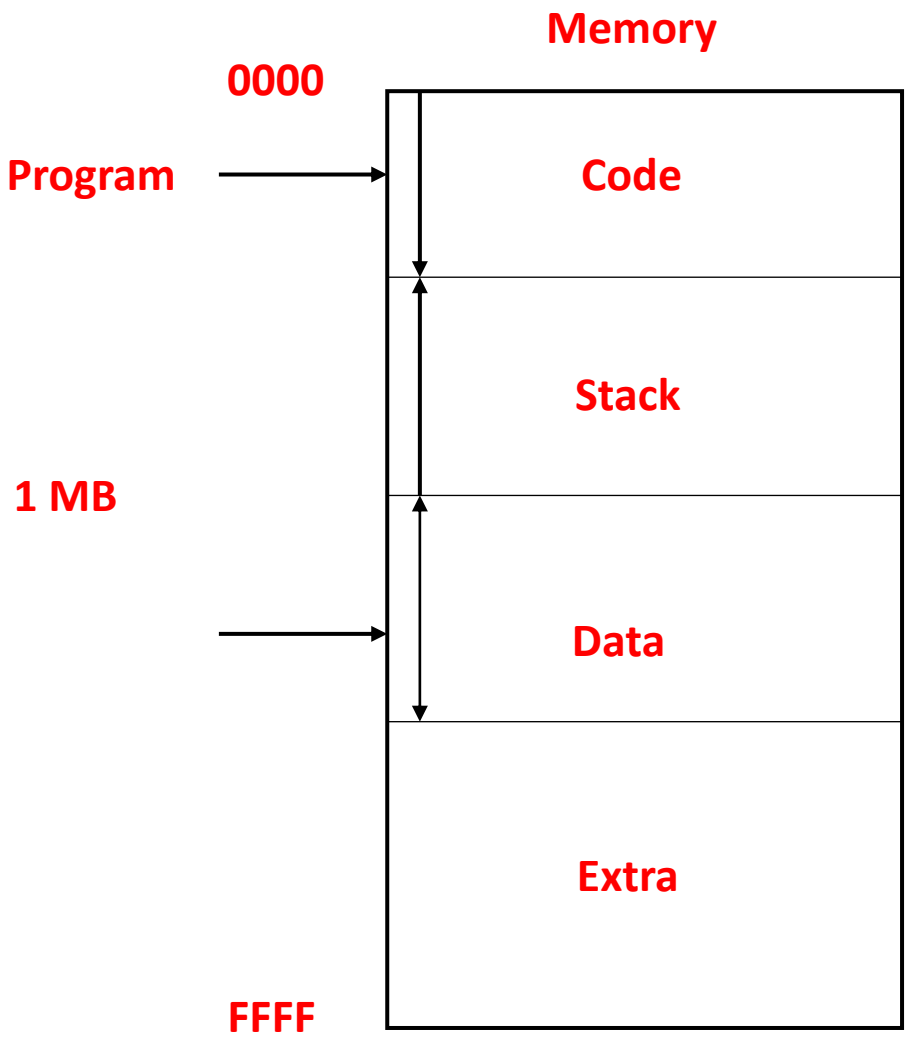
# Memory Segmentation of 8086

# Problem : Memory accessed by 8085




- Memory and Processor are the different chips.
- Memory is used to stored code, stack and data.
- Code is referred as a program which is always stored in sequence.
- Data can stored in any direction.
- Stack is used to stored the data in last in first out manner.

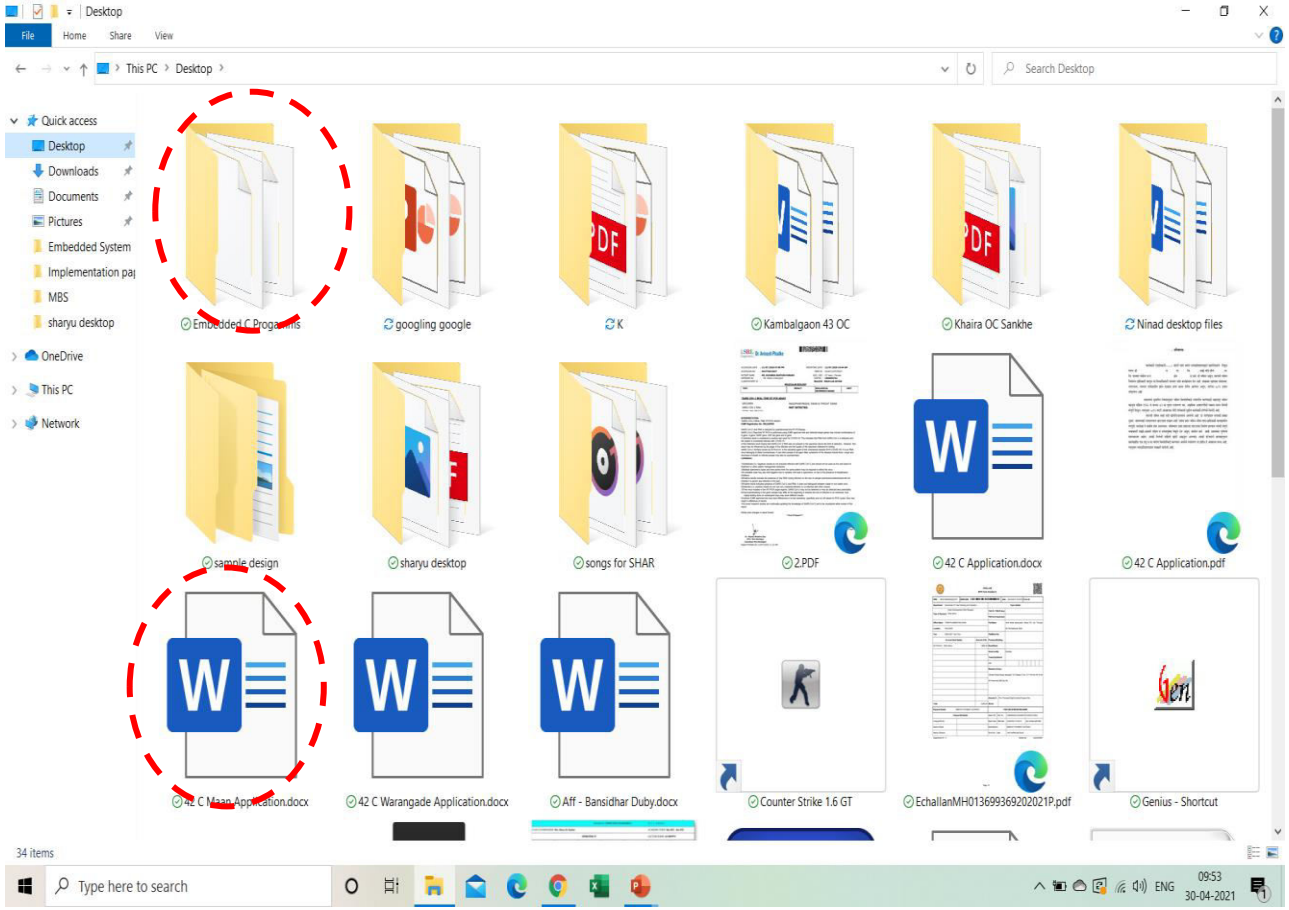
**At some point they all are going to overwrite each other.**

## Solution :

Memory is divided into different parts which called as segments to automatically prevent the overwriting .

**In 8086 memory segmentation concept was invented to avoid overwriting.**

- **Programmers are responsible to create the segments at the time of initialization when start the program.**
- **Processor is responsible to manage the segments (avoid overwriting)**




**Files and fodders are the example of segmentation.**

**There are two types of addresses :**

1. **Virtual address**
2. **Physical address**

- **File/folder name is virtual address-known to programmer**
- **Actual location in memory is physical address – unknown to programmer.**

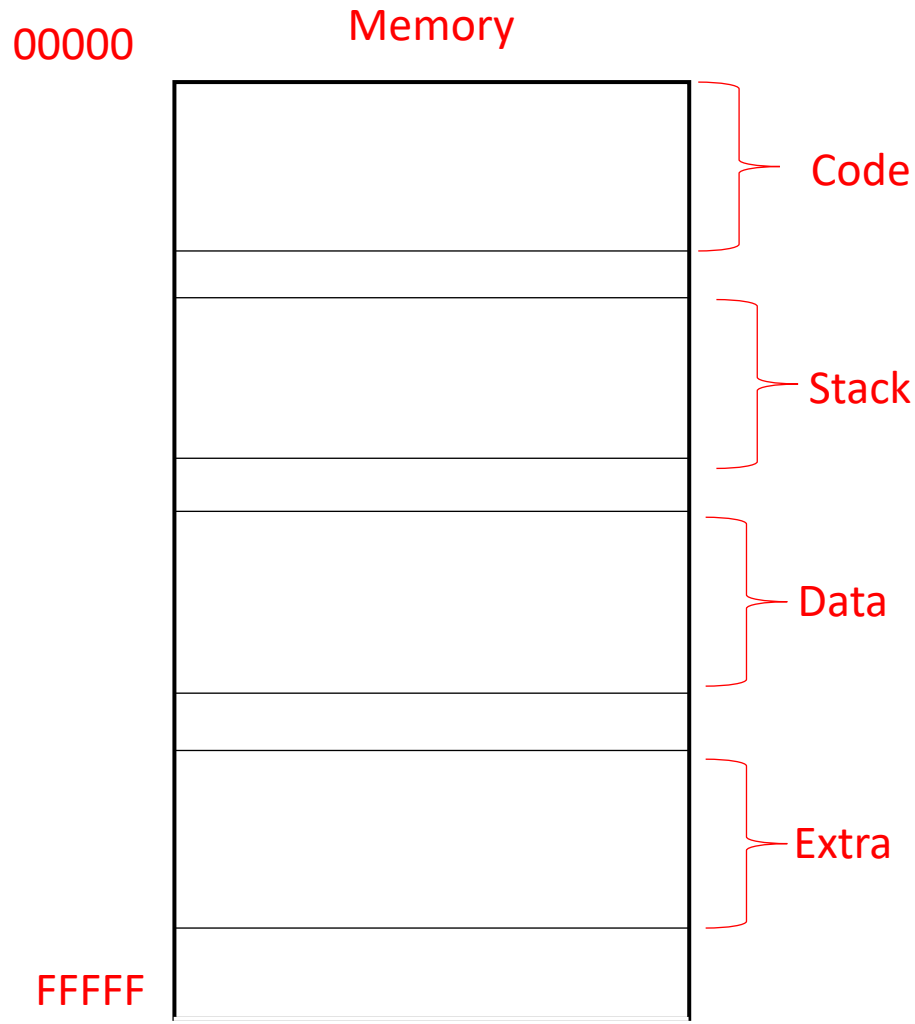
- If address bus size is 8 bit then we have $2^8$ = 256 unique address of memory locations.
- If address bus size is 16 bit then we have $2^{16}$ = 65535 = 64k memory locations.
- If we want 1MB memory hence 8086 having $2^{20}$ = 1MB i.e. 20 bit address bus.

PA = 12345

↓

0001 0010 0011 0100 0101 = 20 bit

- Individual memory location size is 8 bit.
- Hence the address bus size is 20 bit then 2 and ½ memory locations are required to store one instruction in case of 8086.
- Which means wastage of ½ byte.
- If 16 bit address bus then no wastage of memory locations.




We want **20 bit address bus** to increase the memory size
We want **16 bit address** to provide equal memory locations

To achieve this we are going to use **16 bit virtual** address which will converted into **20 bit physical** address.

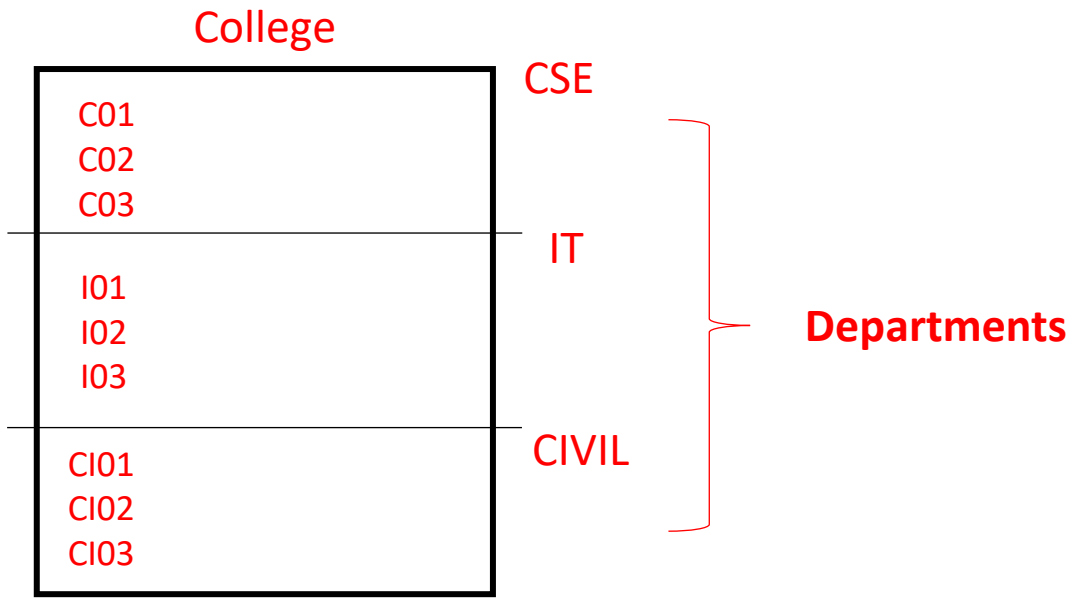
## Virtual address = segment address + offset address

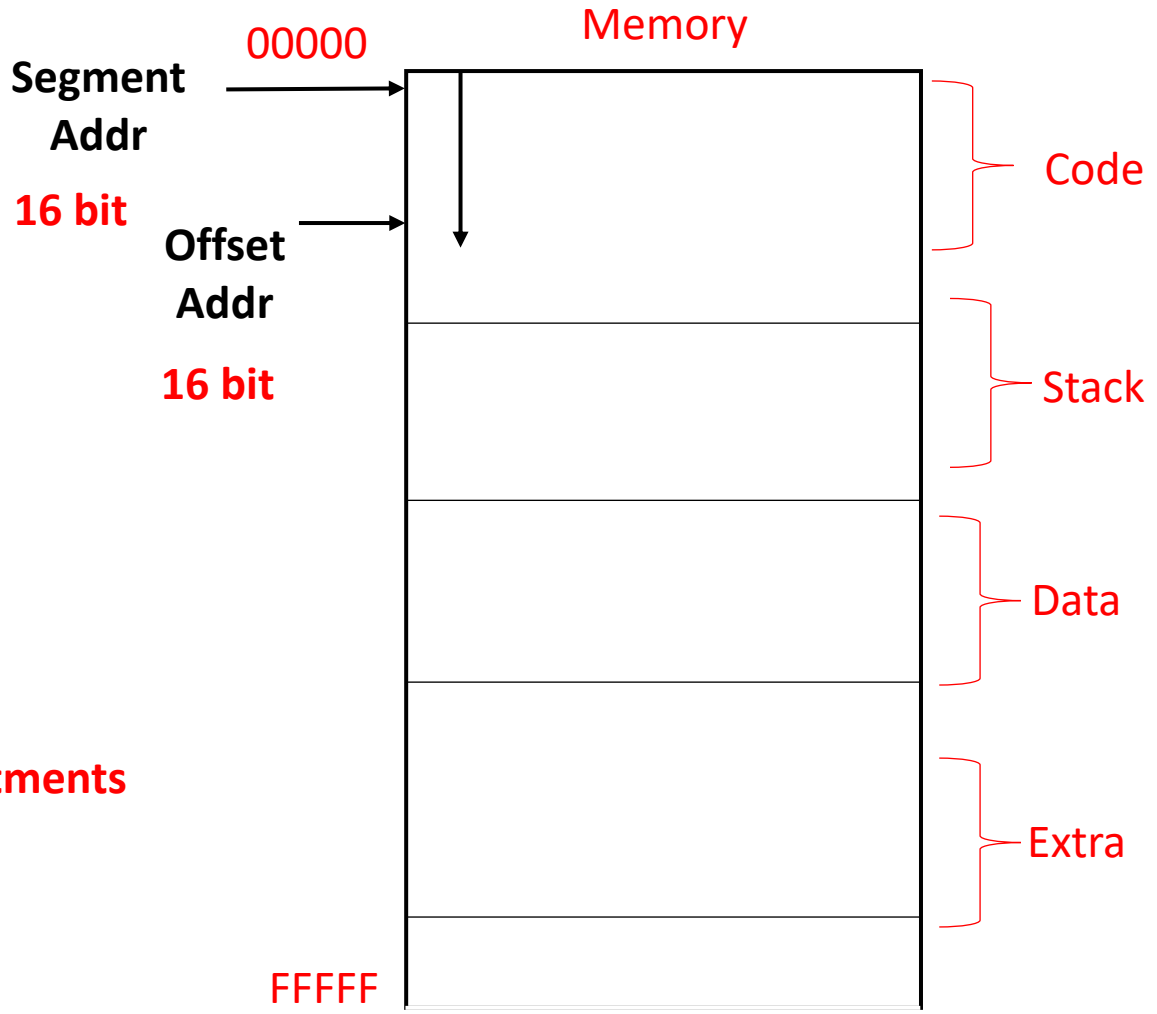
Both are 16 bit which means compatible.

**Segment Addr gives the starting address of segment**

**Offset Addr gives the location which is present at that segment.**

**Example :**




**To call particular student from college**

**Segment Addr = Department Name**
**offset Addr = roll number**

**To call C01**

**Segment Addr = C**
**offset Addr = 01**

**To call C03**

**Segment Addr = C**
**offset Addr = 03**

**No need to change segment address when location is in same segment**

There are three addresses :

1. Physical address — Not given by programmer

2. Segment address — Part of virtual address which is given only once

3. Offset address — Part of virtual address which is changed.

- Offset addresses are limited they are starting with 0000 to FFFF because they are 16 bits.

**Maximum size of segment**
**= $2^{16}$ = $2^{6}$ X $2^{10}$ = 64 X 1KB = 64 KB**

8086

| Seg reg | Off reg |
|---|---|
| CS 1000 | IP 2345 |
| SS | SP & BP |
| DS | SI |
| ES | DI |

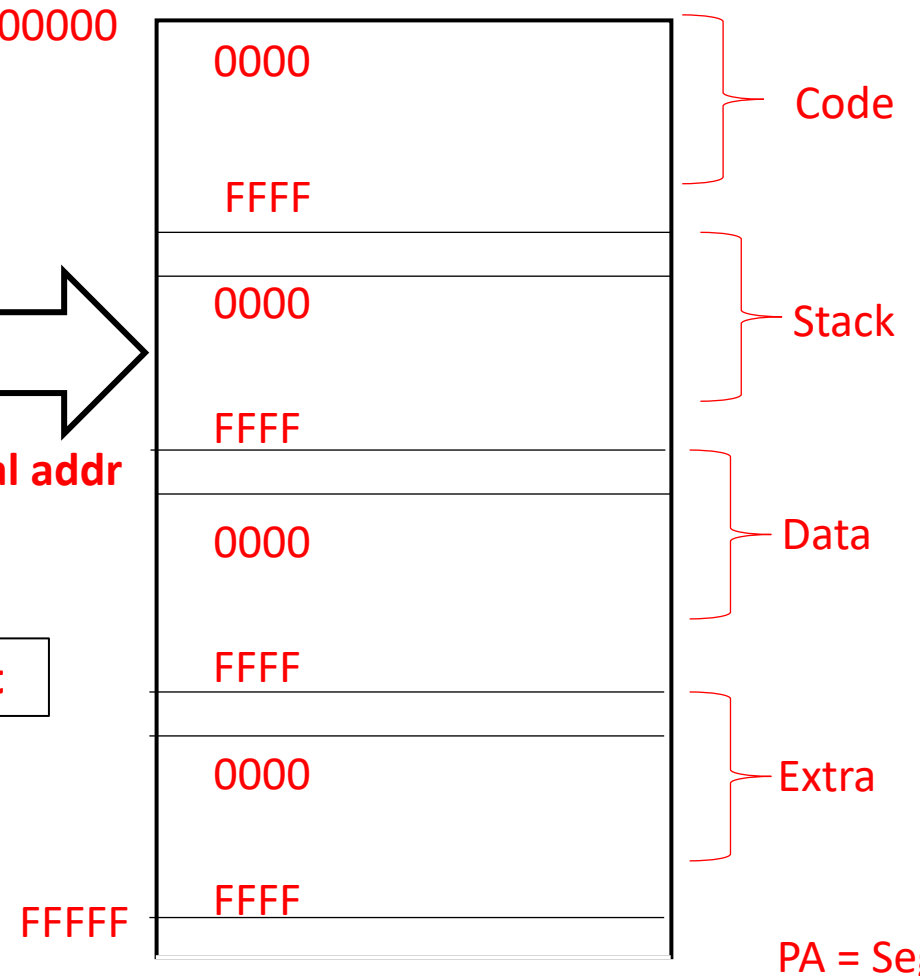
**12345 H**

20 bit address bus

**Processor gives the physical addr**

**Which is 20 bit**

**PA = Seg x 10H + offset**



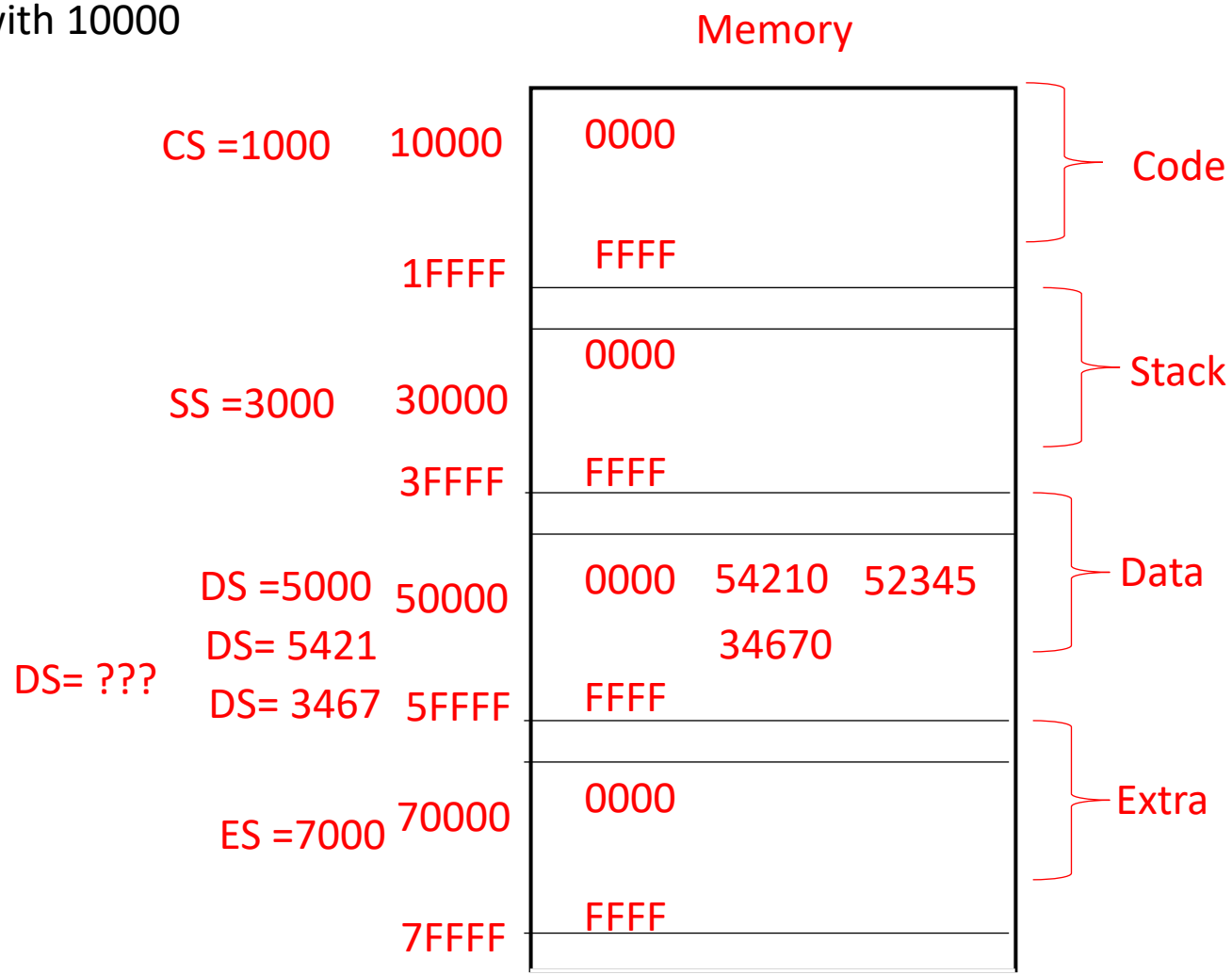

- 8086 having segment registers which are used to store the segment address and offset address of memory
- When processor wants to execute instruction from memory CS and IP is given
- Processor having 20 bit address bus

PA = Seg x 10H + offset
1000 x 10H + 2345
10000 + 2345
PA = 12345 H

If CS is 1000 then Code segment will actually start with 10000

8086

| Seg reg | Off reg |
|---|---|
| CS 1000 | IP 2345 |
| SS 3000 | SP & BP |
| DS | SI |
| ES | DI |




If SS is 3000 then Code segment will actually start with 30000

If DS is 5000 then Code segment will actually start with 50000

If ES is 7000 then Code segment will actually start with 70000

Suppose we want to start DS from any random number

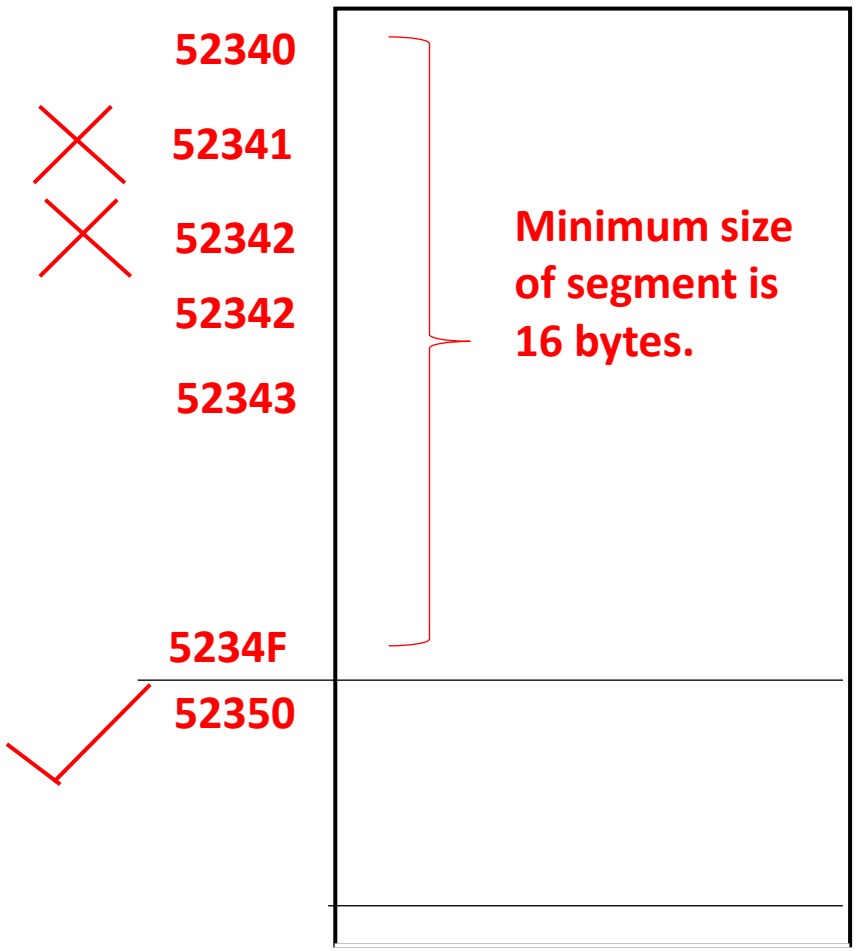
*****A segment can not begin at any location you want, it has to begin at location which is multiple of 10

**What is the minimum size of segment ????**

8086

| Seg reg | Off reg |
|---|---|
| CS | IP |
| SS | SP & BP |
| DS 5234 | SI |
| ES | DI |

- Because it is not multiple by 10
- Because it is multiple by 10




- Data segment starts with **52340** in memory
- We want only this locations after this we want next segment i.e. extra segment.

**What is the next segment address????**

**16 = 10 H**

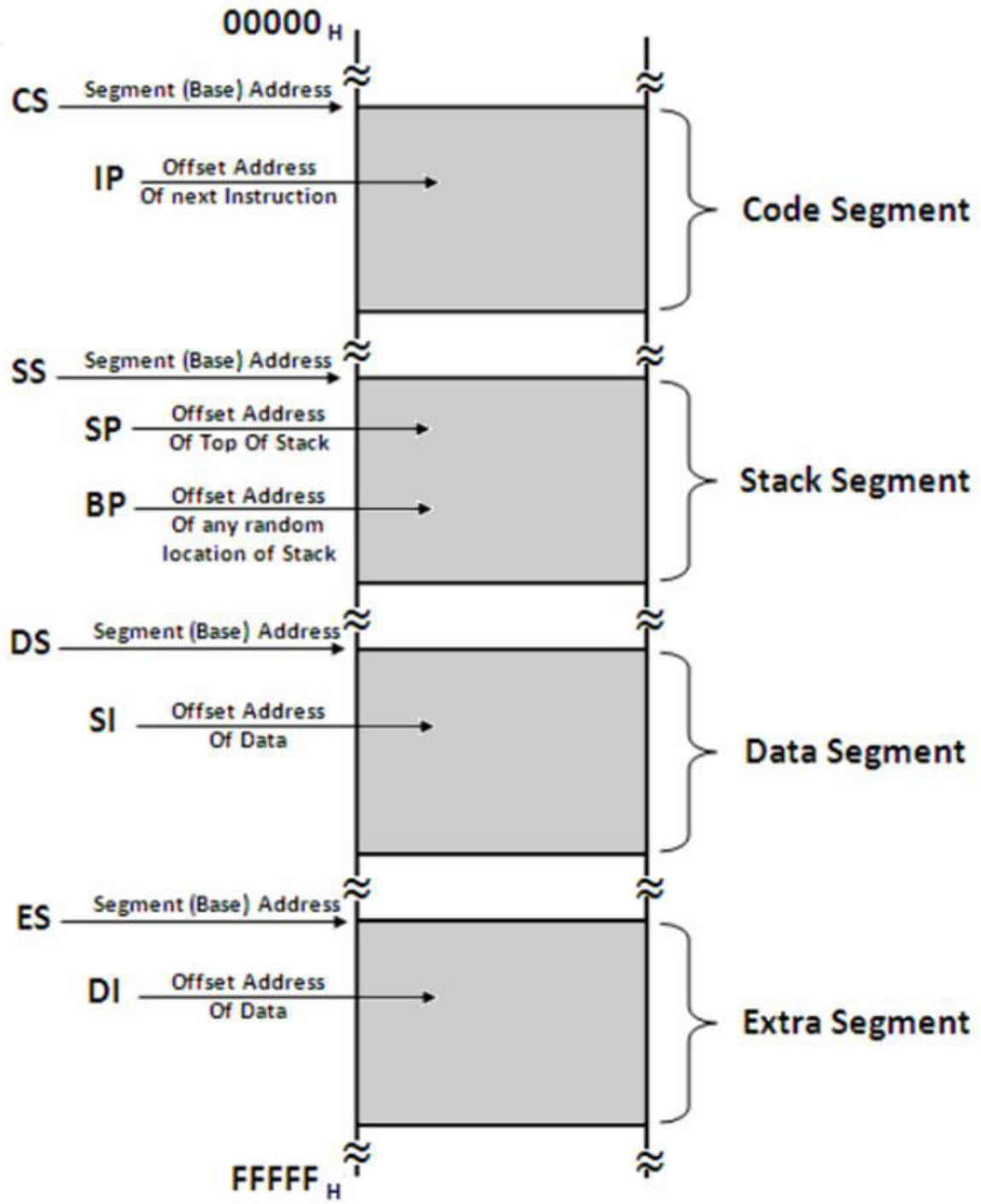
# Memory Segmentation of 8086 for exam

# MEMORY SEGMENTATION IN 8086

## NEED FOR SEGMENTATION/ CONCEPT OF SEGMENTATION

1) Segmentation means **dividing** the memory into **logically different parts called segments**.
2) 8086 has a **20-bit address bus**, hence it can access $2^{20}$ Bytes i.e. **1MB** memory.
3) But this also means that **Physical address** will now be **20 bit**.
4) It is **not possible** to work with a **20 bit address** as it is **not** a **byte compatible** number. (20 bits is two and a half bytes).
5) To avoid working with this incompatible number, we **create a virtual model** of the memory.
6) Here the memory is **divided into 4 segments**: Code, Stack Data and Extra.
7) The **max size** of a segment is **64KB** and the **minimum size** is **16 bytes**.
8) Now programmer can access each location with a **VIRTUAL ADDRESS**.
9) The Virtual Address is a **combination** of **Segment Address and Offset Address**.
10) **Segment Address indicates where the segment is located in the memory (base address)**
11) **Offset Address gives the offset of the target location within the segment.**
12) Since both, Segment Address and Offset Address are **16 bits each**, they both are **compatible numbers** and can be easily used by the programmer.
13) Moreover, **Segment Address is given only in the beginning** of the program, to initialize the segment. Thereafter, we **only give offset address**.
14) **Hence we can access 1 MB memory using only a 16 bit offset address for most part of the program. This is the advantage of segmentation.**
15) Moreover, dividing Code, stack and Data into different segments, makes the memory **more organized and prevents accidental overwrites** between them.
16) The **Maximum Size** of a segment is **64KB because offset addresses are of 16 bits**. $2^{16}$ = 64KB.
17) As max size of a segment is 64KB, programmer can create **multiple Code/Stack/Data segments** till the entire 1 MB is utilized, but **only one of each** type will be **currently active**.
18) The physical address is calculated by the microprocessor, using the formula:

    **PHYSICAL ADDRESS = SEGMENT ADDRESS X 10H + OFFSET ADDRESS**
19) Ex: if Segment Address = 1234H and Offset Address is 0005H then
    Physical Address = 1234H x 10H + 0005H = 12345H
20) This formula automatically ensures that the **minimum size of a segment is 10H bytes** (10H = 16 Bytes).



## Code Segment

This segment is used to hold the **program** to be executed.
**Instruction are fetched** from the Code Segment.
**CS** register holds the 16-bit **base** address for this segment.
**IP** register (Instruction Pointer) holds the 16-bit **offset** address.

## Data Segment

This segment is used to hold **general data**.
This segment also holds the **source** operands during **string** operations.
**DS** register holds the 16-bit **base** address for this segment.
**BX** register is used to hold the 16-bit **offset** for this segment.
**SI** register (Source Index) holds the 16-bit **offset** address during String Operations.

## Stack Segment

This segment holds the **Stack** memory, which operates in LIFO manner.
**SS** holds its **Base** address.
**SP** (Stack Pointer) holds the 16-bit **offset** address of the **Top** of the Stack.
**BP** (Base Pointer) holds the 16-bit **offset** address during **Random Access**.

## Extra Segment

This segment is used to hold **general data**
Additionally, this segment is used as the **destination** during **String Operations**.
**ES** holds the **Base** Address.
**DI** holds the **offset** address during string operations.

## Advantages of Segmentation:

1) It permits the programmer to access 1MB **using only 16-bit address**.
2) Its **divides** the **memory logically** to store Instructions, Data and Stack separately.

## Disadvantage of Segmentation:

1) Although the total memory is 16*64 KB, **at a time only 4*64 KB memory can be accessed**.