**Segmentation**

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.

**Segmented Memory Model**

The CPU has special registers (segment registers) to hold the base address (starting point) of each segment. Every program has its own memory segments (CS, DS, SS, and ES), and each segment is of 64kb, so 64 \* 4 = 256kb in memory allocation:

**CSR (Code Segment Register):** Stores starting address of code segment of the current program.

**DSR (Data Segment Register):** stores starting address of data segment of the current program.

**SSR (Stack Segment Register):** stores starting address of stack segment of the current program.

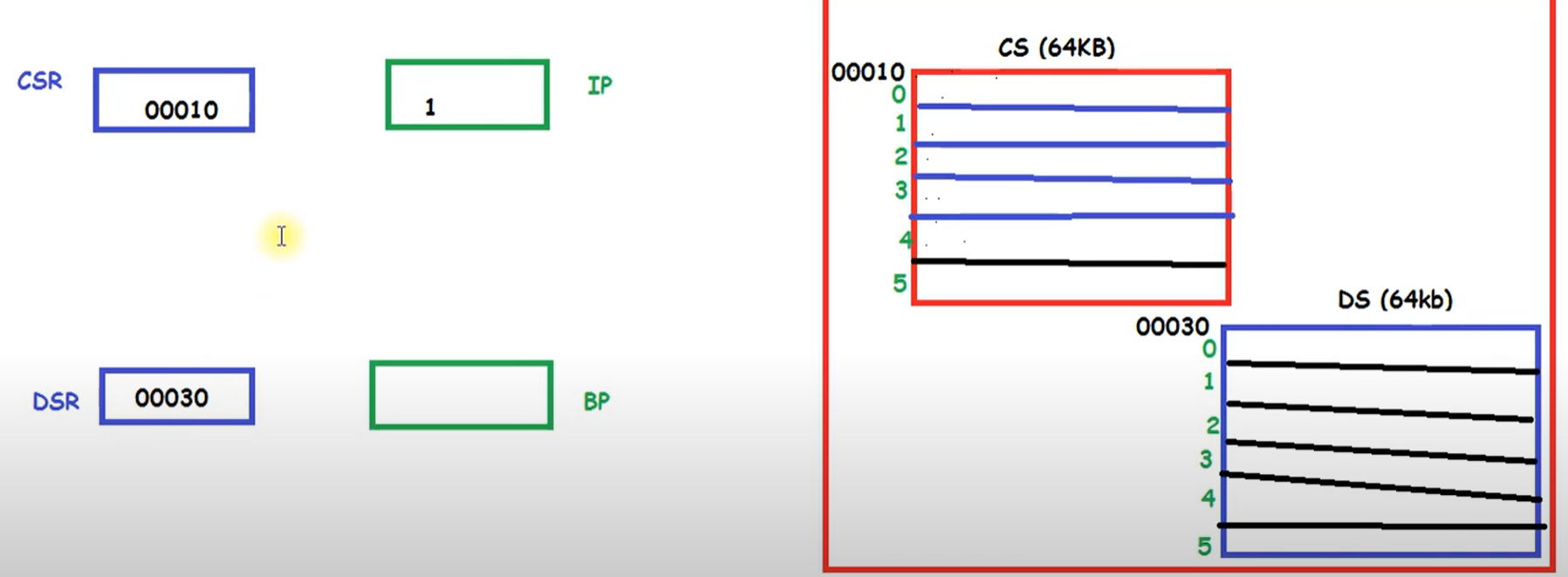
**ESR (Extra Segment Register):** stores starting address of extra segment of the current program.

**Offset Registers**

Every memory segment has its own offset register. Offset is simply the distance between the starting address of a memory segment and the desired location in the segment.

In the x86 segmented memory model, each **segment register** corresponds to a particular memory segment, and each segment has associated **offset registers**. The offset registers define the location within the segment, creating a full **physical address** when combined with the segment register.





CSR is pointing to the starting address of the Code Segment of the currently executing code, and the Instruction Pointer is working as an offset register.

A screenshot of a computer

Description automatically generated

**Clarification:**

* **Each segment does not have its own dedicated offset register** (e.g., there is no unique offset register tied to each segment like CS, DS, SS, etc.). Instead, **general-purpose registers** (such as SP, BP, SI, DI, etc.) and **specific registers** (such as IP for CS) act as offset registers depending on the operation and the segment being accessed.

**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*.

Why a segment start cannot start from the physical address 55,555?

**Segmentation Alignment**: The x86 architecture uses a segmentation model where segment bases are aligned to certain boundaries. For instance, in real mode and protected mode, segment bases are often aligned to 16-byte boundaries. This is because the segment registers in x86 architecture are shifted left by 4 bits, effectively giving segment bases that are multiples of 16.

**Segment Register Limits**: In real mode, segment registers are 16-bit and can represent values from 0x0000 to 0xFFFF. When multiplied by 16 (shifting left by 4 bits), this gives segment bases that are multiples of 16, i.e., 0x0000, 0x0010, 0x0020, ..., up to 0xFFFF0.

**Physical Address Calculation**: In real mode, the physical address is calculated as segment base \* 16 + offset. If you use a segment base that doesn’t align with the 16-byte boundary, the resulting address calculation would not be consistent with the segment boundaries.

**Segment Overlap**: Segment registers define the base address of a segment. If a segment base address is not aligned correctly, it might cause overlap or conflict with other segments or lead to invalid memory access.

**Example**

For physical address 0xD9AB:

* To align a segment base correctly, you would need to ensure it is a multiple of 16.
* 0xD9AB divided by 16 gives 0xD9A with a remainder. Therefore, it is not aligned to a 16-byte boundary.