2. The Bus Interface Unit (BIU):

The main components of the BIU are:

- 6-byte Instruction Queue (Q). or 6 Byte Pre-fetch Queue (FIFO).
- Segment *Registers* (CS, DS, ES, SS).
- Instruction Pointer (IP).
- Address Generation (Σ).

BIU performs the following functions:

- It generates the 20-bit physical address for memory access (Address relocation).
- It fetches instructions or operand from the memory.
- It transfers data to and from the memory and I/O.
- Maintains the 6-byte pre-fetch instruction queue (supports pipelining).

BIU takes care of all data and addresses transfers on the buses for the EU like sending addresses, fetching instructions from the memory, reading data from the memory as well as writing data to the memory. EU has no direction connection with System Buses, so this is possible just with the BIU.

For this reason, the Segment Registers are located in the BIU. While the General-purpose registers are located in the EU.

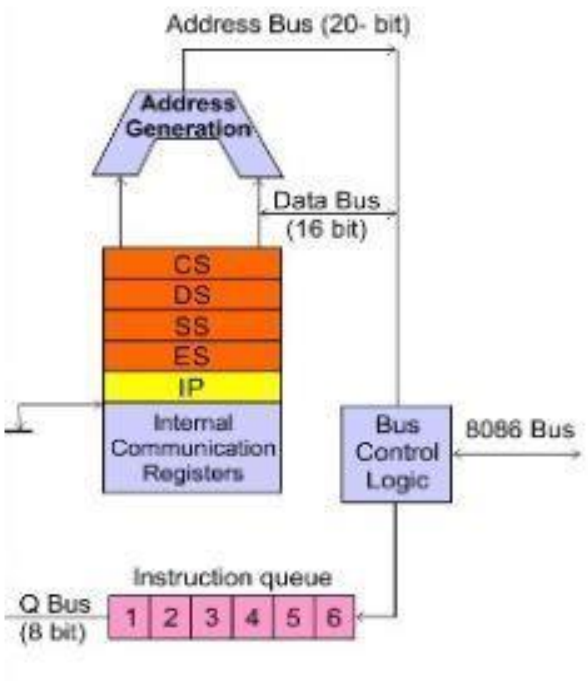

Fig. shows Bus Interface unit (BIU)

Segment Registers

The 8086 has *four* special segment registers: Code Segment *CS*, Data Segment *DS*, Extra Segment *ES*, and Stack Segment *SS*. These registers are all 16 bits wide. The Segment Registers have a very special purpose – pointing at accessible blocks of main memory.

To be able to support memory address spaces larger than the native size of the internal address register would allow, early CPUs implemented a system of segmentation whereby they would store a small set of indexes to use as offsets to certain areas.

Each Segment was placed at a specific location in memory by the software being executed and all instructions that operated on the data within those segments were performed relative to the start of that segment. This allowed a 16-bit address register, which would normally be able to access 64 KB (65536 Bytes) of memory space, in order to access 1 MB of memory space.

They deal with selecting blocks (segments) of main memory. As following below:

1. ***The CS register:*** points at the segment containing the currently executing machine instructions. The CS, also known as a text segment that *contains executable instructions.*
2. ***The DS register:*** is that segment of memory which is used to store global variables for the program. The *DS register points to* the data segment of the memory where the data is stored. *DS* is a 16-bit register containing address of 64KB segment.
3. ***The ES register:*** is exactly that: a spare segment that may be used for specifying a location in memory. The 8086 programs often *use ES register* to gain access to segments when it is difficult or impossible to modify the other segment registers.
4. ***The SS register:*** it stores the starting address of the stack. The Stack Segment is that segment of memory which is *used* to store stack data. The stack stores subroutine returns addresses, procedure parameters, and local variables.

What is the advantage of memory segmentation?

- Segmentation *helps* you to *increase* the speed of execution so that processor can able to fetch & execute the data from the memory even *faster and easier*.
- The Segmentation *allows* the memory capacity to be *1 MB, although* the actual address to be handled is of *16-bit size*. *(how) (Because)?*

Without segmentation, it would require 20-bit registers. Segmentation, is the process in which the main memory of the computer is divided into different segments and each segment has its own base address.

If the 8086 in 16 bits can only address 64K of RAM! How many address lines or address buses are required to address 1 MB memory?

It depends not on the amount of memory, but on the address space. So, you need log2(n) bits to address n bytes.

For example, you can store 256 different values in an 8-bit number, so 8 bits can address 256 bytes. 1KB = 2^{10} = 1024, so you need 10 bits to address every byte in a kilobyte. Likewise, you need 20 bits to address every byte in a *Megabyte*. So, in order to address memory larger than 64K, segment registers add additional bits to any memory addressing.

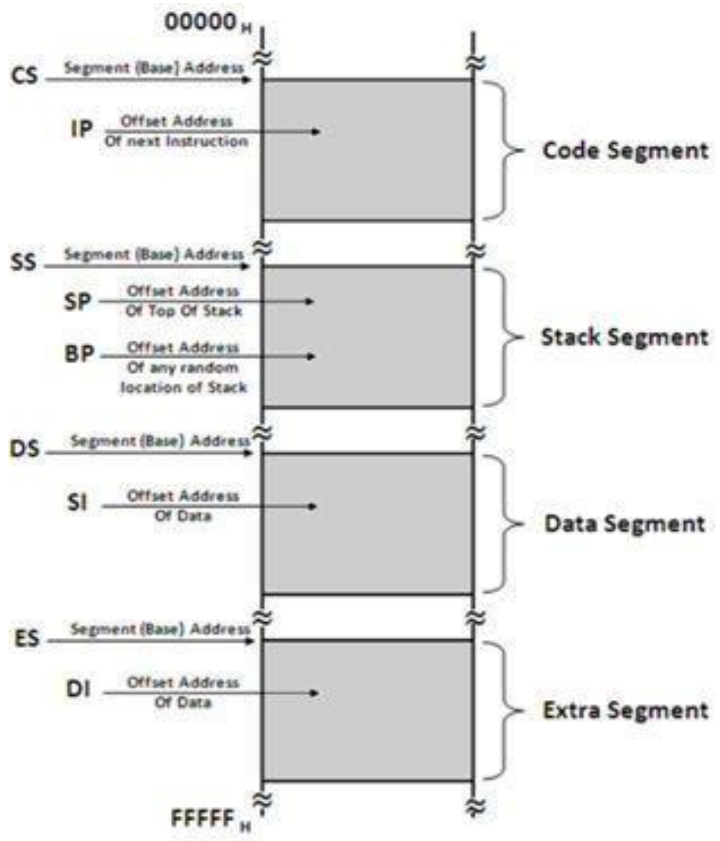

Fig.: Memory Segmentations in 8086

Table: Default segment and offset register combinations

Segment Registers	*Offset Registers*	*Purpose*
CS	*IP*	*Address of the next Instruction, also called PC*
DS	*SI, DI*	*Address of Data*
SS	*SP, BP*	*SP: top of the stack, BP: Address in Stack*
ES	*DI*	*String Destination Address, for String Instruction*