

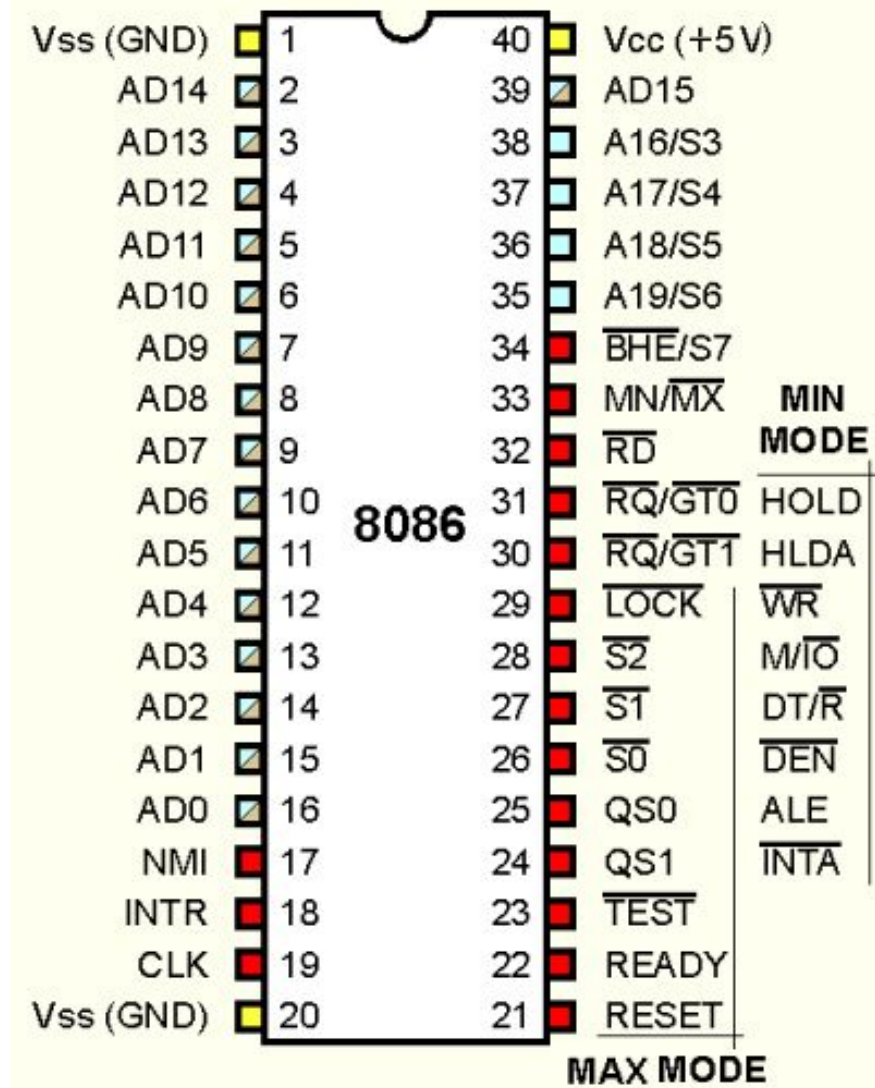
Introduction to 8086 Microprocessor

Microprocessor 8086

Features of Microprocessor 8086

- ❖ It is a 16-bit microprocessor.
- ❖ It can operate on 5,8,10 MHz
- ❖ It has 20 bit address bus.
- ❖ it has 16 bit I/O address lines.
- ❖ It supports pipelined architecture.
- ❖ can operate in two mode maximum mode and minimum mode.
- ❖ It uses memory banks
- ❖ Memory is segmented.
- ❖ 256 vectored Interrupts.

Pin Configuration Of 8086

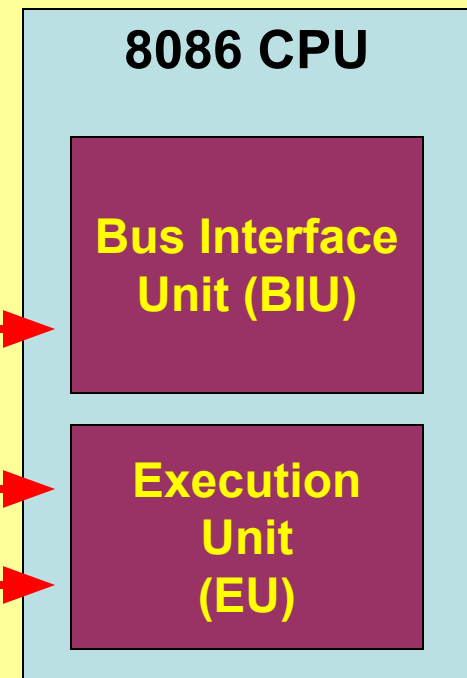


8086 Microprocessor

- Belongs to a popular microprocessor series
 - 8086, 80186, 80286, 80386, 80486, Pentium
- INTEL launched 8086 in 1978
- 8086 is a 16-bit microprocessor with
 - 16-bit Data Bus
 - 20-bit Address Bus

8086 Internal Architecture

- 8086 employs parallel processing
- 8086 CPU has two parts which operate at the same time
 - Bus Interface Unit
 - Execution Unit
- CPU functions
 1. Fetch
 2. Decode
 3. Execute



Bus Interface Unit

- Sends out addresses for memory locations
- Fetches Instructions from memory
- Reads/Writes data to memory
- Sends out addresses for I/O ports
- Reads/Writes data to Input/Output ports

Execution Unit

- Tells BIU (addresses) where to fetch instructions or data
- Decodes & Executes instructions
- Dividing the work between BIU & EU speeds up processing

Architecture Diagram of 8086

Intel 8086 Internal Architecture

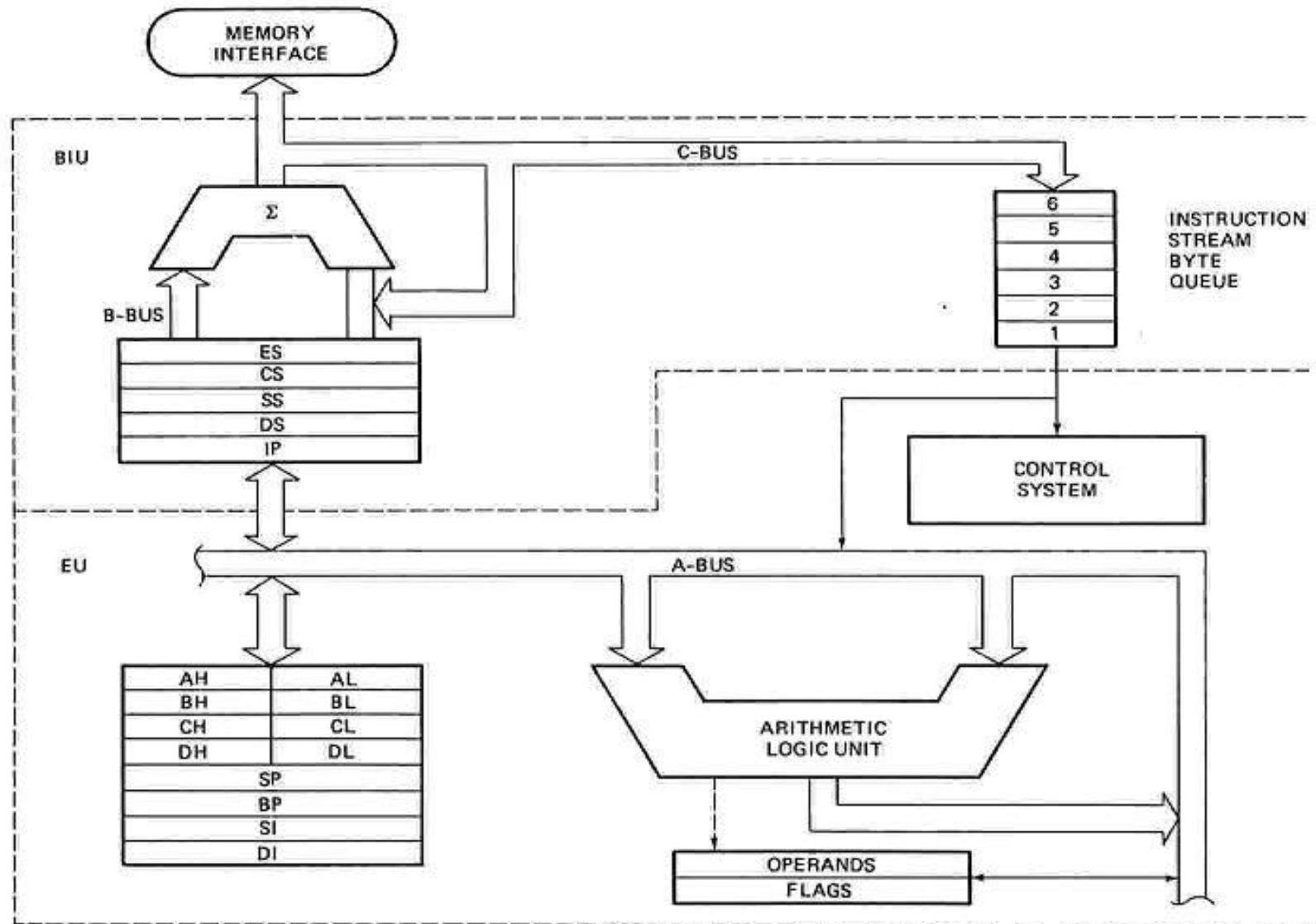


FIGURE 8086 internal block diagram. (Intel Corp.)

Execution Unit

- Main components are
 - **Instruction Decoder**
 - **Control System**
 - **Arithmetic Logic Unit**
 - **General Purpose Registers**
 - **Flag Register**
 - **Pointer & Index registers**

Instruction Decoder

- Translates instructions fetched from memory into a series of actions which EU carries out

Control System

- Generates timing and control signals to perform the internal operations of the microprocessor

Arithmetic Logic Unit

- EU has a 16-bit ALU which can ADD, SUBTRACT, AND, OR, increment, decrement, complement or shift binary numbers

General Purpose Registers

- EU has 8 general purpose registers
- Can be individually used for storing 8-bit data
- AL register is also called Accumulator
- Two registers can also be combined to form 16-bit registers
- The valid register pairs are – AX, BX, CX, DX

AH	AL
BH	BL
CH	CL
DH	DL

AH	AL	AX
BH	BL	BX
CH	CL	CX
DH	DL	DX

Flag Register

- 8086 has a 16-bit flag register
- Contains 9 active flags
- There are two types of flags in 8086
 - **Conditional** flags – six flags, set or reset by EU on the basis of results of some arithmetic operations
 - **Control** flags – three flags, used to control certain operations of the processor

Flag Register

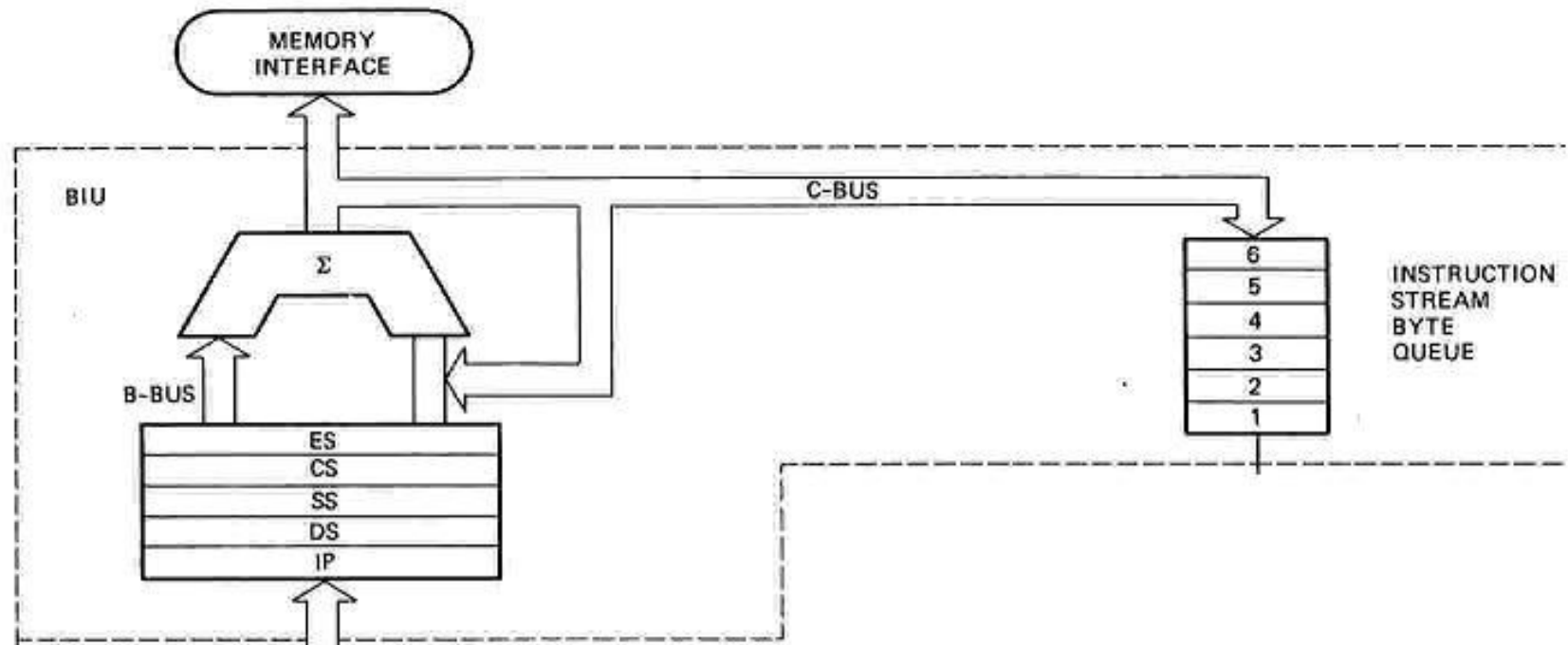
U	U	U	U	OF	DF	IF	TF	SF	ZF	U	AF	U	PF	U	CF
---	---	---	---	----	----	----	----	----	----	---	----	---	----	---	----

1.	CF	CARRY FLAG	Conditional Flags
2.	PF	PARITY FLAG	
3.	AF	AUXILIARY CARRY	
4.	ZF	ZERO FLAG	
5.	SF	SIGN FLAG	
6.	OF	OVERFLOW FLAG	
7.	TF	TRAP FLAG	Control Flags
8.	IF	INTERRUPT FLAG	
9.	DF	DIRECTION FLAG	

Bus Interface Unit

- Main Components are
 - **Instruction Queue**
 - **Segment Registers**
 - **Instruction Pointer**

Bus Interface Unit



Instruction Queue

- ❑ 8086 employs parallel processing
- ❑ When EU is busy decoding or executing current instruction, the buses of 8086 **may not be** in use.
- ❑ At that time, BIU can use buses to fetch upto six instruction bytes for the following instructions
- ❑ BIU stores these pre-fetched bytes in a **FIFO** register called Instruction **Queue**
- ❑ When EU is ready for its next instruction, it simply reads the instruction from the queue in BIU

Pipelining

- **EU of 8086 does not have to wait in between for BIU to fetch next instruction byte from memory**
- **So the presence of a queue in 8086 speeds up the processing**
- **Fetching the next instruction while the current instruction executes is called pipelining**

Memory Segmentation

- 8086 has a **20-bit** address bus
- So it can address a maximum of **1MB** of memory
- 8086 can work with only **four 64KB segments** at a time within this 1MB range
- These four memory segments are called
 - **Code** segment
 - **Stack** segment
 - **Data** segment
 - **Extra** segment

Memory

**64KB Memory
Segment**



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

00000H



**1MB
Address
Range**



FFFFFFH

**Only 4 such segments can be
addressed at a time**

Code Segment

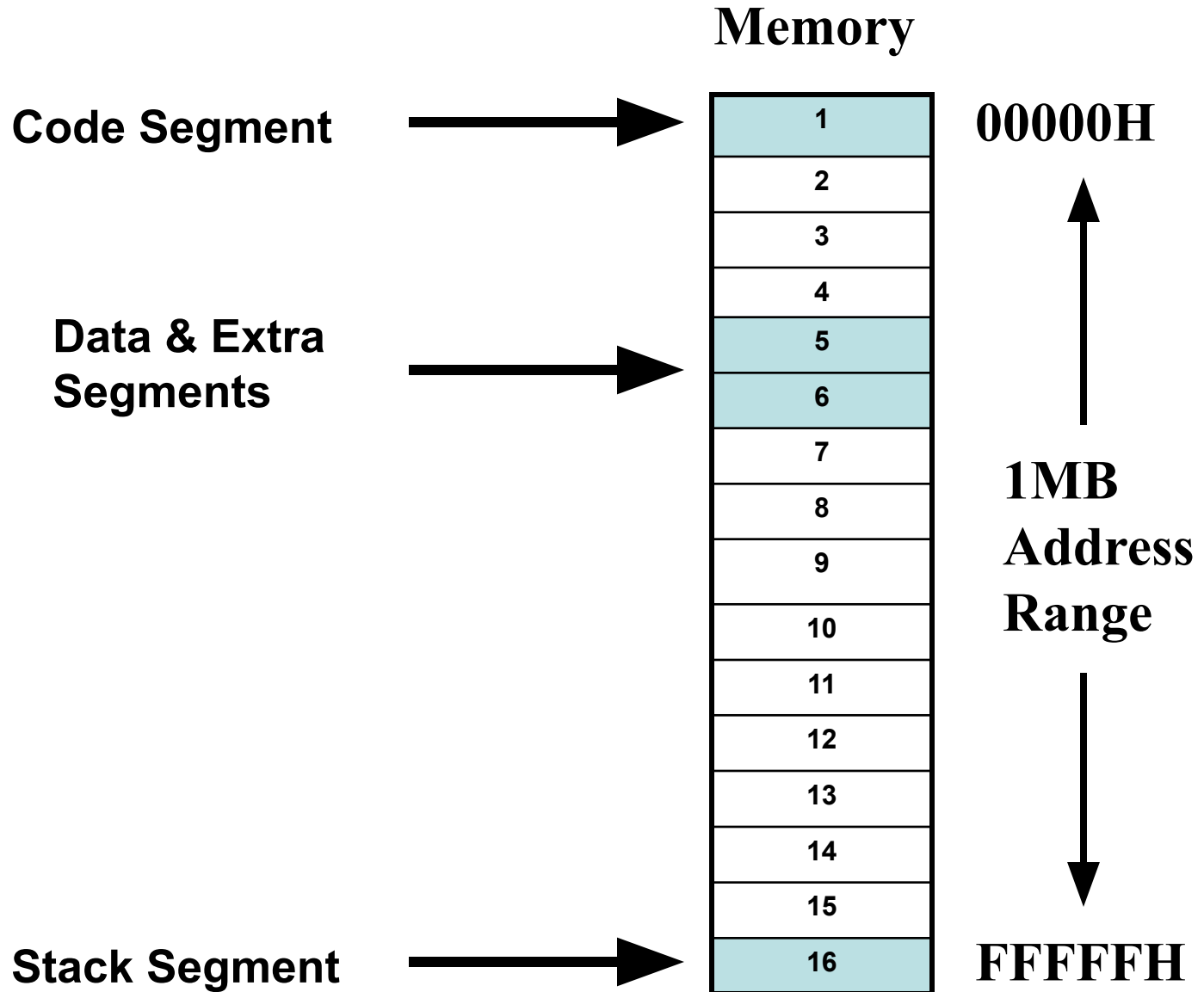
- That part of memory from where BIU is currently fetching instruction code bytes

Stack Segment

- A section of memory set aside to store addresses and data while a subprogram executes

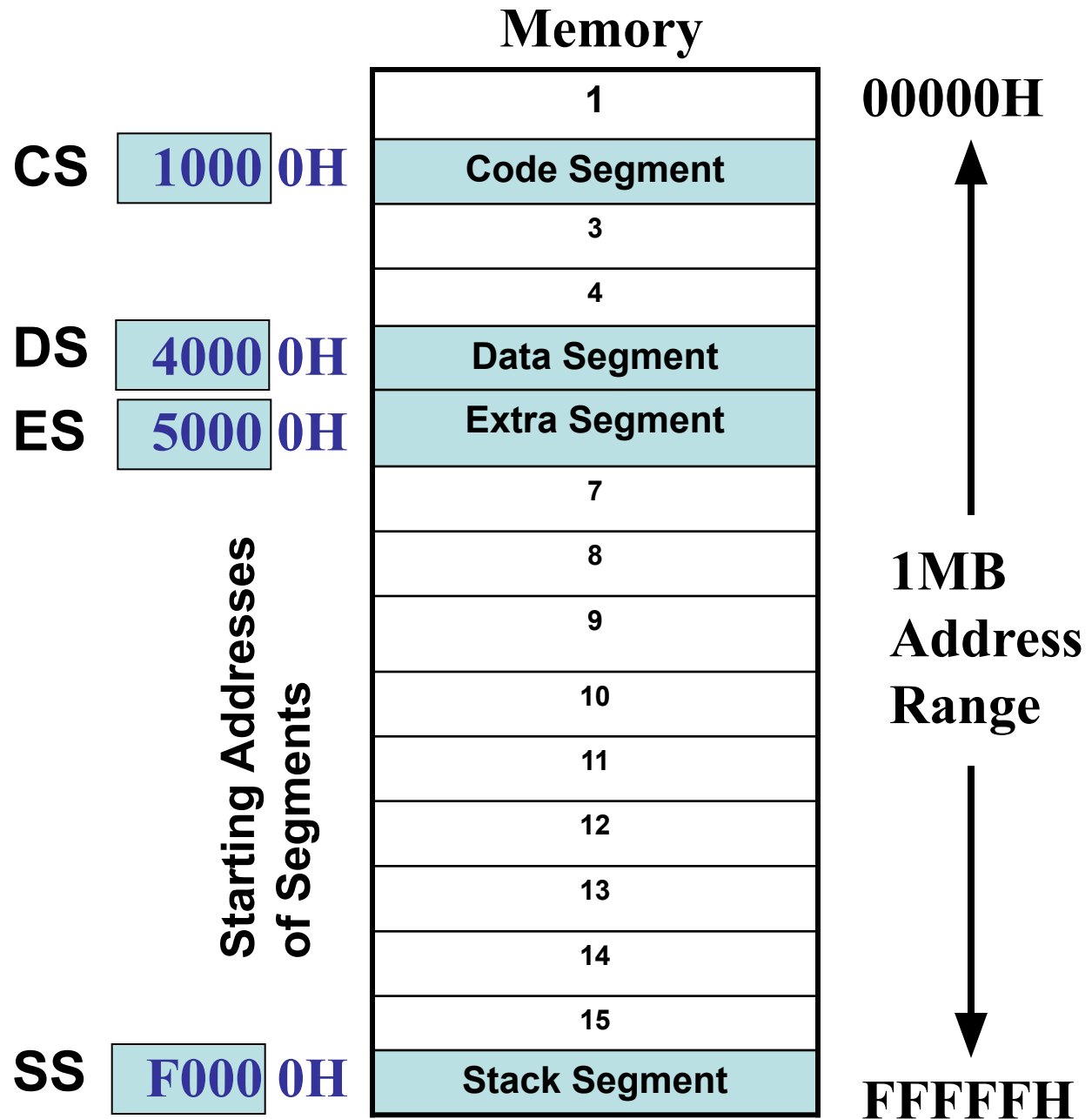
Data & Extra Segments

- Used for storing data values to be used in the program



Segment Registers

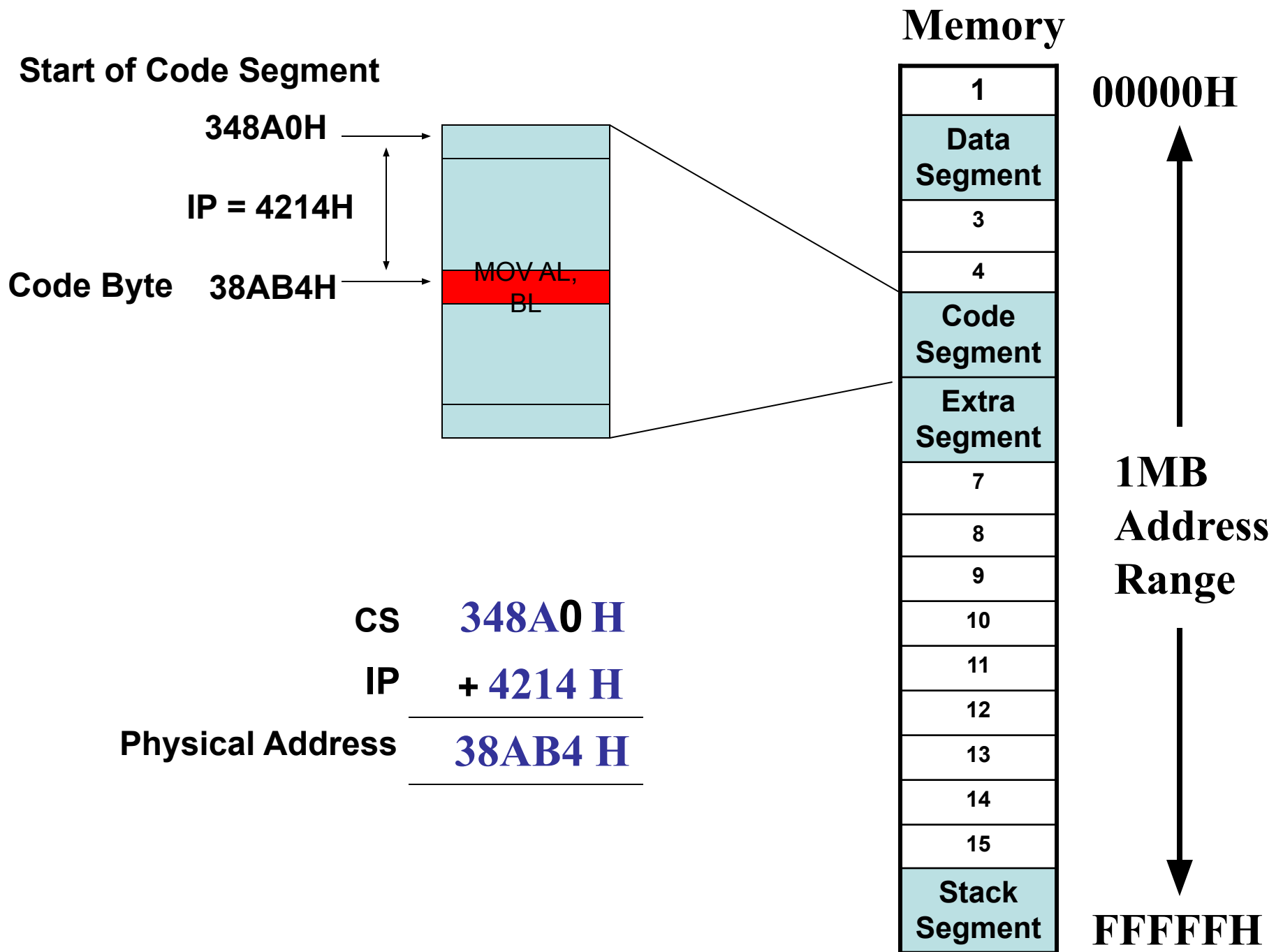
- hold the upper 16-bits of the starting address for each of the segments
- The four segment registers are
 - **CS (Code Segment register)**
 - **DS (Data Segment register)**
 - **SS (Stack Segment register)**
 - **ES (Extra Segment register)**



- Address of a segment is of 20-bits
- A segment register stores only upper 16-bits
- BIU always inserts zeros for the lowest 4-bits of the 20-bit starting address.
- E.g. if CS = 348AH, then the code segment will start at 348A0H
- A 64-KB segment can be located anywhere in the memory, but will start at an address with zeros in the lowest 4-bits

Instruction Pointer (IP) Register

- a 16-bit register
- Holds 16-bit **offset**, of the next instruction byte in the **code segment**
- BIU uses **IP** and **CS** registers to generate the **20-bit address** of the instruction to be fetched from memory



Stack Segment (SS) Register

Stack Pointer (SP) Register

- Upper 16-bits of the starting address of stack segment is stored in SS register
- It is located in BIU
- SP register holds a 16-bit offset from the start of stack segment to the top of the stack
- It is located in EU

Other Pointer & Index Registers

- Base Pointer (BP) register
- Source Index (SI) register
- Destination Index (DI) register
- Can be used for temporary storage of data
- Main use is to hold a 16-bit offset of a data word in one of the segments

Segment and Address register combination

- CS:IP
- SS:SP SS:BP
- DS:BX DS:SI
- DS:DI (for other than string operations)
- ES:DI (for string operations)