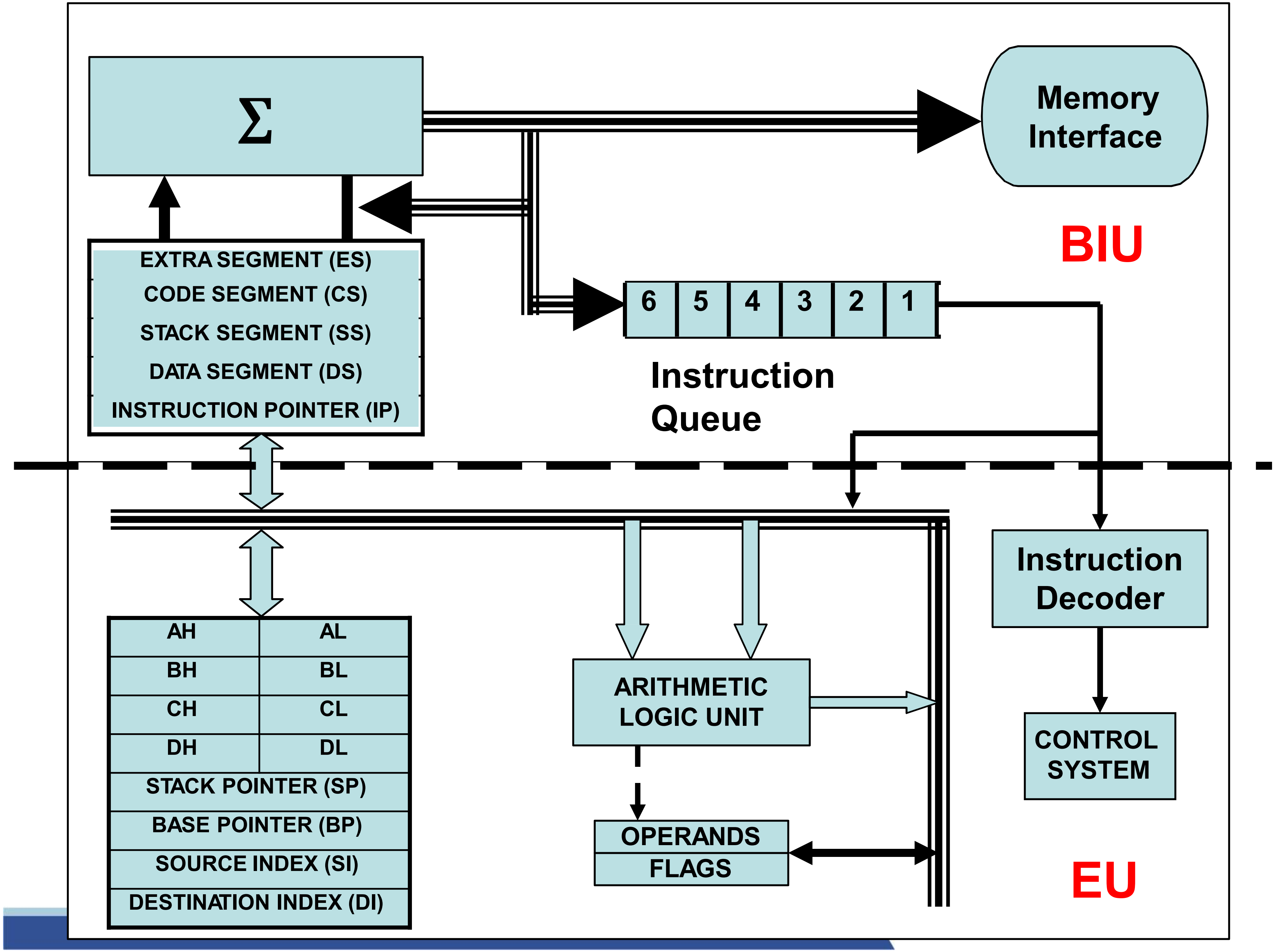
8086 Architecture


Σ
Memory Interface
BIU
EXTRA SEGMENT (ES)
CODE SEGMENT (CS)
STACK SEGMENT (SS)
DATA SEGMENT (DS)
INSTRUCTION POINTER (IP)
6
5
4
3
2
1
Instruction Queue
Instruction Decoder
AH
AL
BH
BL
CH
CL
DH
DL
STACK POINTER (SP)
BASE POINTER (BP)
SOURCE INDEX (SI)
DESTINATION INDEX (DI)
ARITHMETIC LOGIC UNIT
OPERANDS
FLAGS
CONTROL SYSTEM
EU

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
- AX 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

General Purpose Registers

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 (Compatible with 8085, except OF)
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	

Flag Register

Auxiliary Carry Flag

This is set, if there is a carry from the lowest nibble, i.e, bit three during addition, or borrow for the lowest nibble, i.e, bit three, during subtraction.

Carry Flag

This flag is set, when there is a carry out of MSB in case of addition or a borrow in case of

Sign Flag

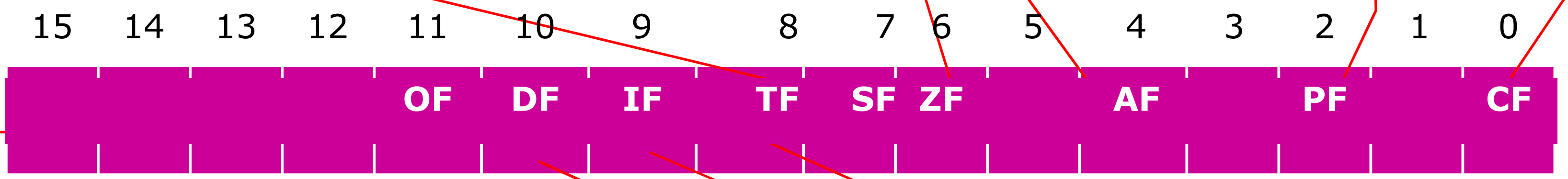
This flag is set, when the result of any computation is negative

Zero Flag

This flag is set, if the result of the computation or comparison performed by an instruction is zero

Parity Flag

This flag is set to 1, if the lower byte of the result contains even number of 1's ; for odd number of 1's set to zero.



Over flow Flag

This flag is set, if an overflow occurs, i.e, if the result of a signed operation is large enough to accommodate in a destination register. The result is of more than 7-bits in size in case of 8-bit signed operation and more than 15-bits in size in case of 16-bit sign operations, then the overflow will be set.

Tarp Flag

If this flag is set, the processor enters the single step execution mode by generating internal interrupts after the execution of each instruction

Direction Flag

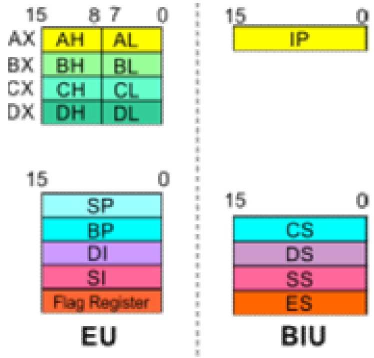
This is used by string manipulation instructions. If this flag bit is '0', the string is processed beginning from the lowest address to the highest address, i.e., auto incrementing mode. Otherwise, the string is processed from the highest address towards the lowest address, i.e., auto incrementing mode.

Interrupt Flag

Causes the 8086 to recognize external mask interrupts; clearing IF disables these interrupts.

Registers, Flag

8086 registers categorized into 4 groups



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				OF	DF	IF	TF	SF	ZF		AF		PF		CF

Sl.No.	Type	Register width	Name of register
1	General purpose register	16 bit	AX, BX, CX, DX
		8 bit	AL, AH, BL, BH, CL, CH, DL, DH
2	Pointer register	16 bit	SP, BP
3	Index register	16 bit	SI, DI
4	Instruction Pointer	16 bit	IP
5	Segment register	16 bit	CS, DS, SS, ES
6	Flag (PSW)	16 bit	Flag register

Registers and Special Functions

Register	Name of the Register	Special Function
AX	16-bit Accumulator	Stores the 16-bit results of arithmetic and logic operations
AL	8-bit Accumulator	Stores the 8-bit results of arithmetic and logic operations
BX	Base register	Used to hold base value in base addressing mode to access memory data
CX	Count Register	Used to hold the count value in SHIFT, ROTATE and LOOP instructions
DX	Data Register	Used to hold data for multiplication and division operations
SP	Stack Pointer	Used to hold the offset address of top stack memory
BP	Base Pointer	Used to hold the base value in base addressing using SS register to access data from stack memory
SI	Source Index	Used to hold index value of source operand (data) for string instructions
DI	Data Index	Used to hold the index value of destination operand (data) for string operations

Bus Interface Unit

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

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

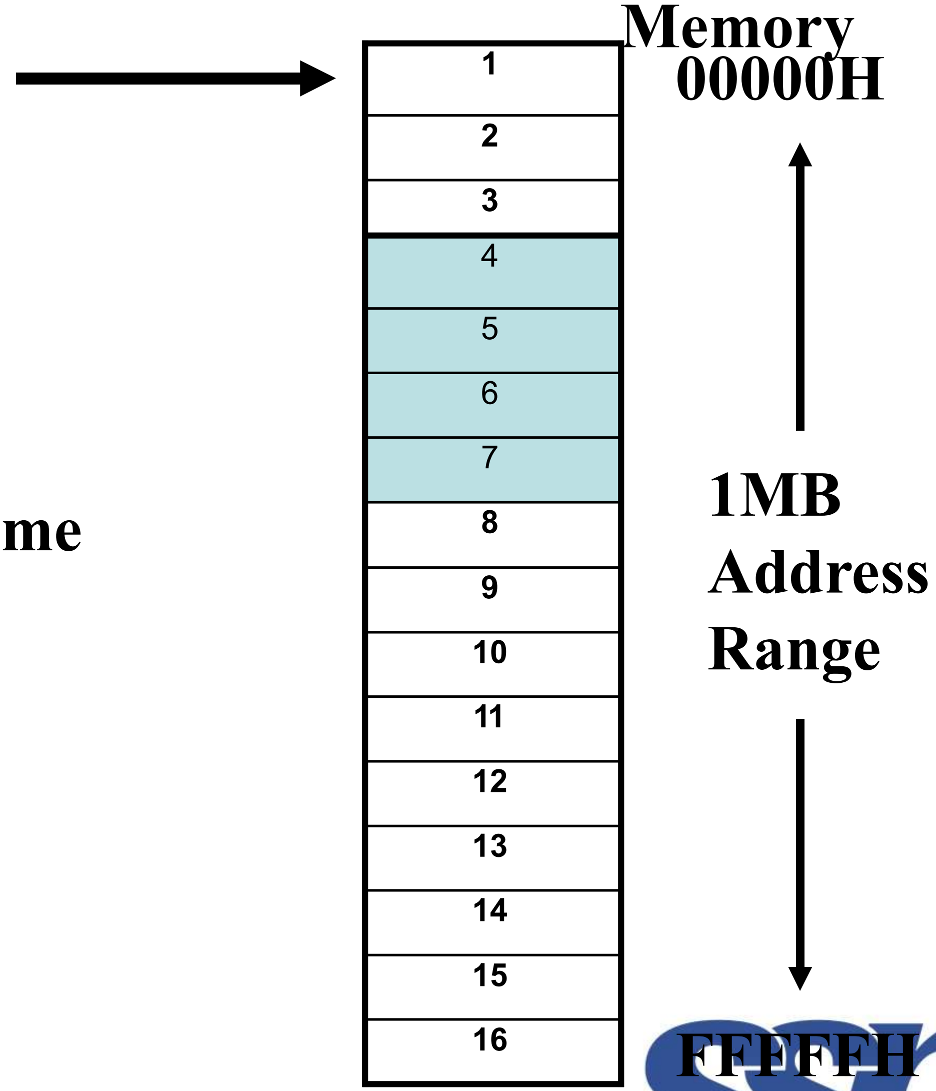
- EU of 8086 does not have to wait in between for BIU to fetch next instruction byte from memory
- So the presence of a queue in 8086 speeds up the processing
- Fetching the next instruction while the current instruction executes is called pipelining

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

64KB Memory Segment

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

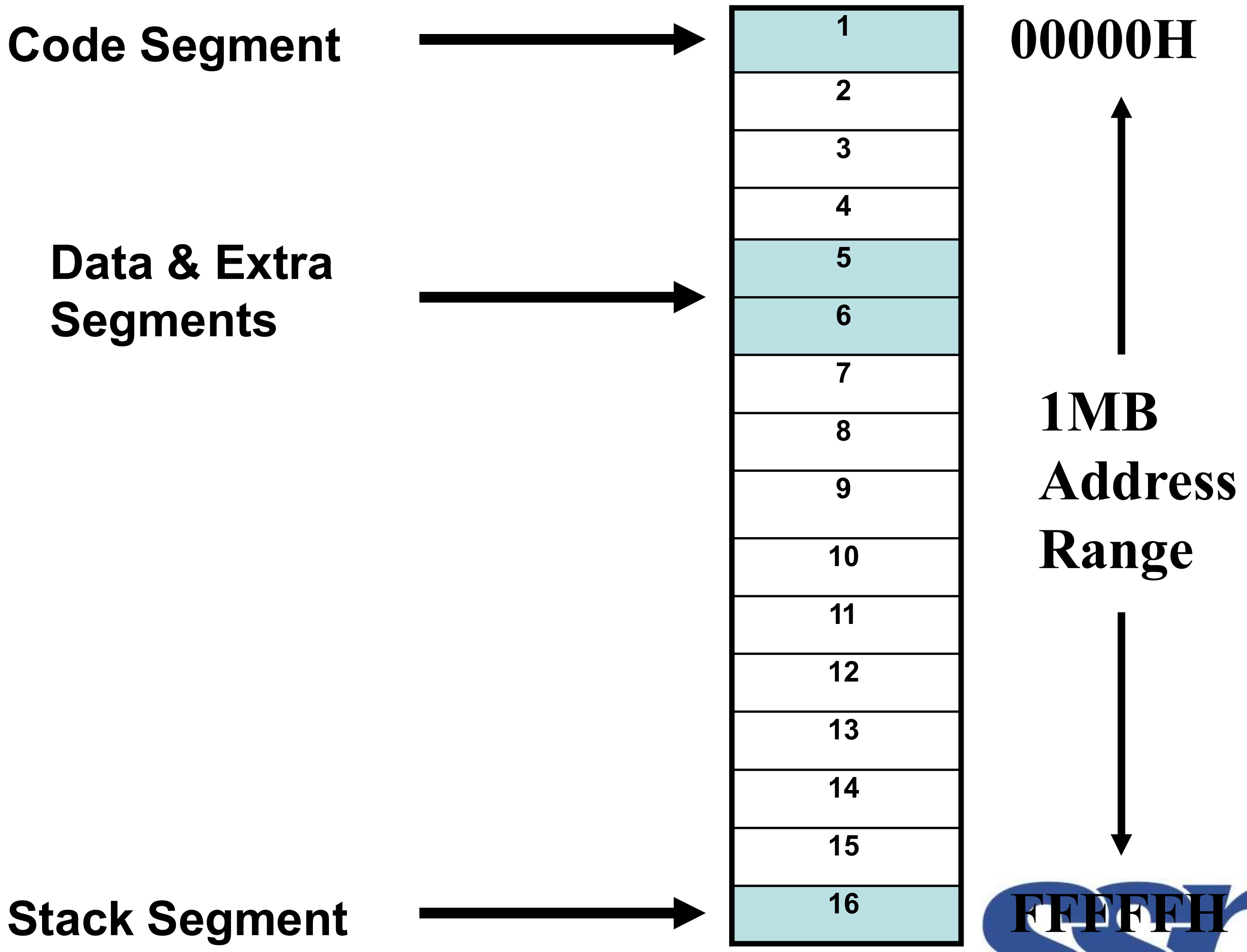
Stack Segment

- 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

Internal Memory



Code Segment

Data & Extra Segments

Stack Segment

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

00000H

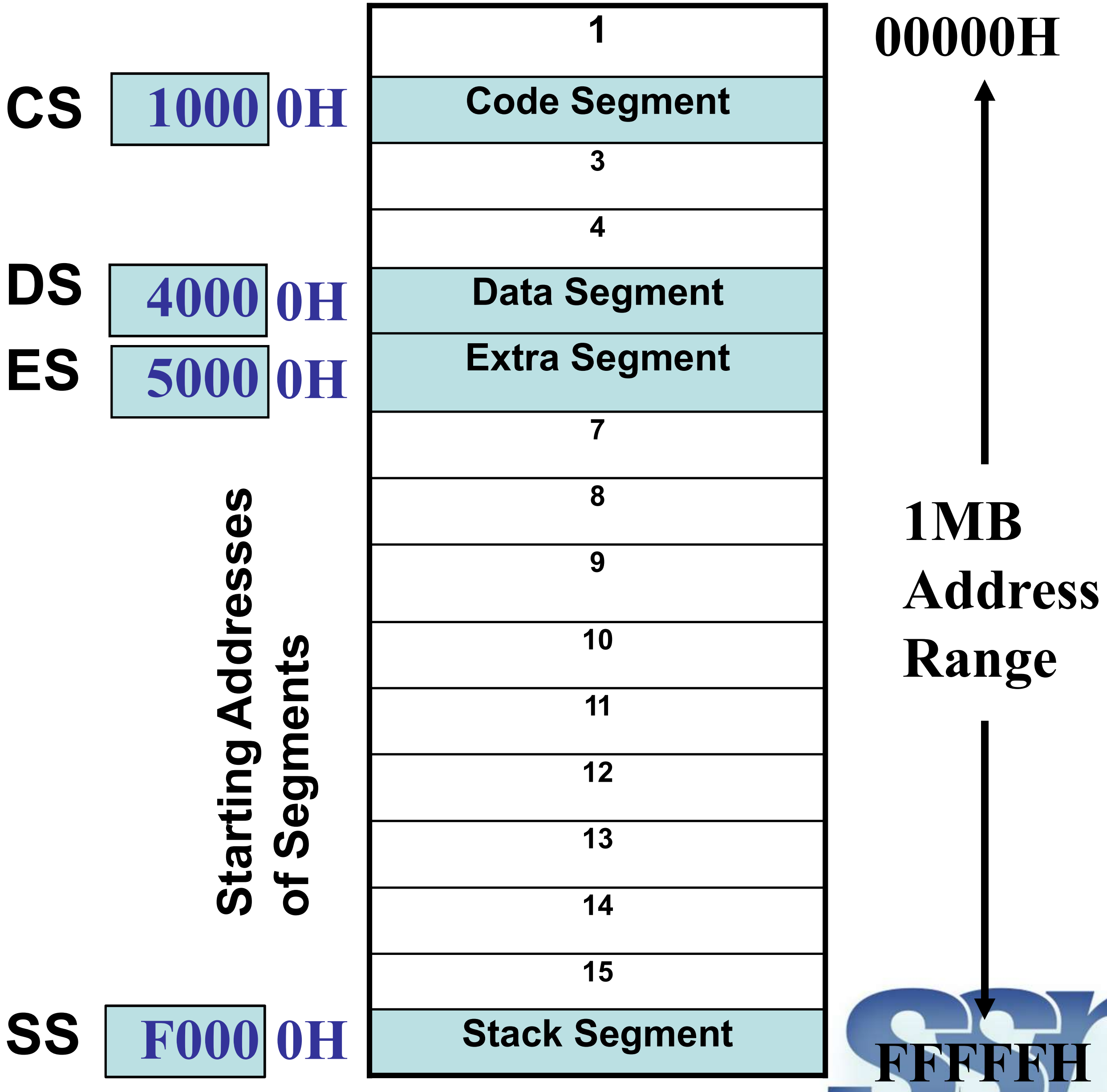
1MB Address Range

FFFFFH

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)**

Internal Memory



CS 1000 0H

DS 4000 0H

ES 5000 0H

SS F000 0H

Starting Addresses of Segments

1

Code Segment

3

4

Data Segment

Extra Segment

7

8

9

10

11

12

13

14

15

Stack Segment

00000H

1MB Address Range

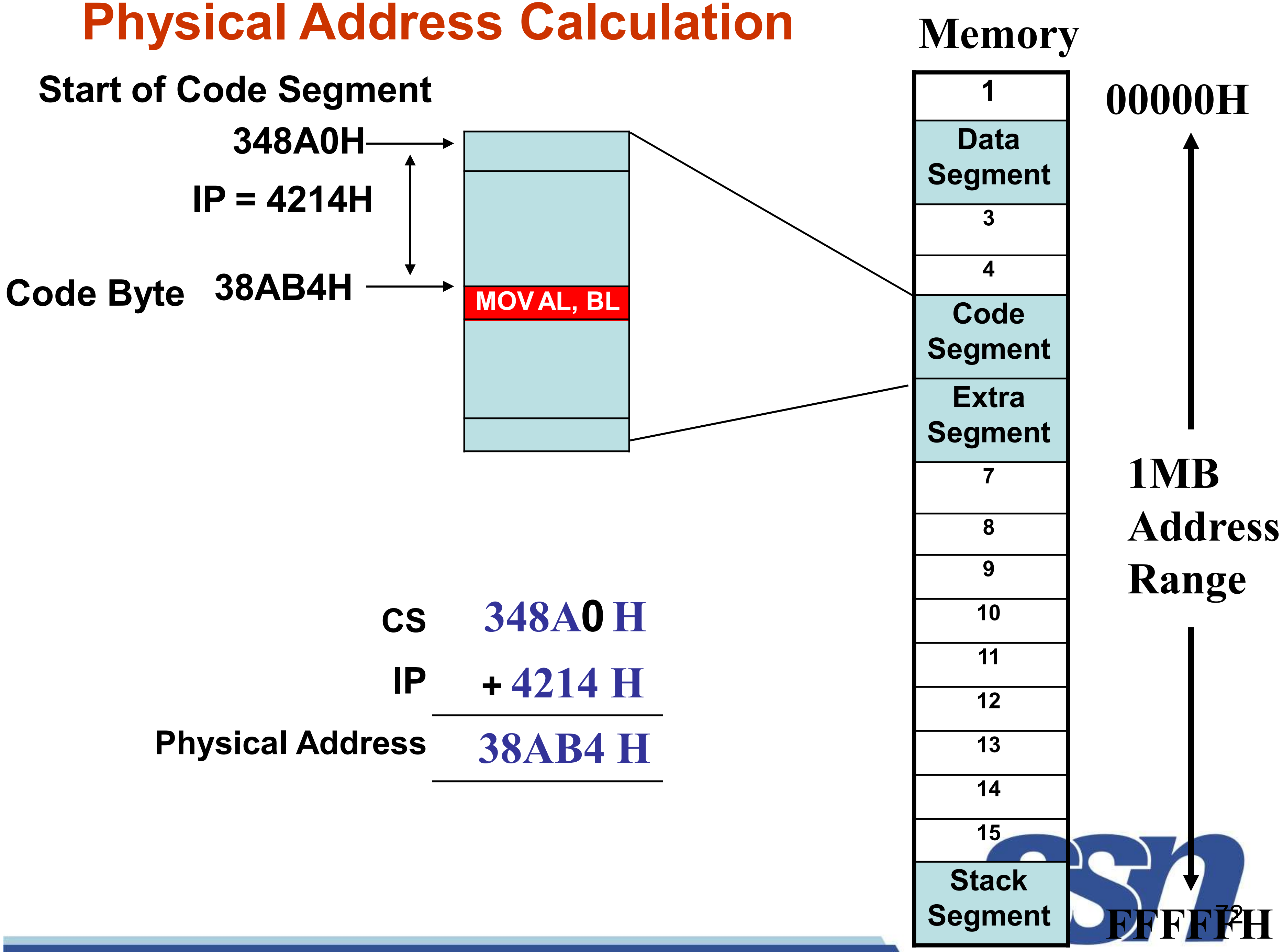
FFFFFH

- 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

Physical Address Calculation



CS 348A0 H

IP + 4214 H

Physical Address 38AB4 H

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

Thank you