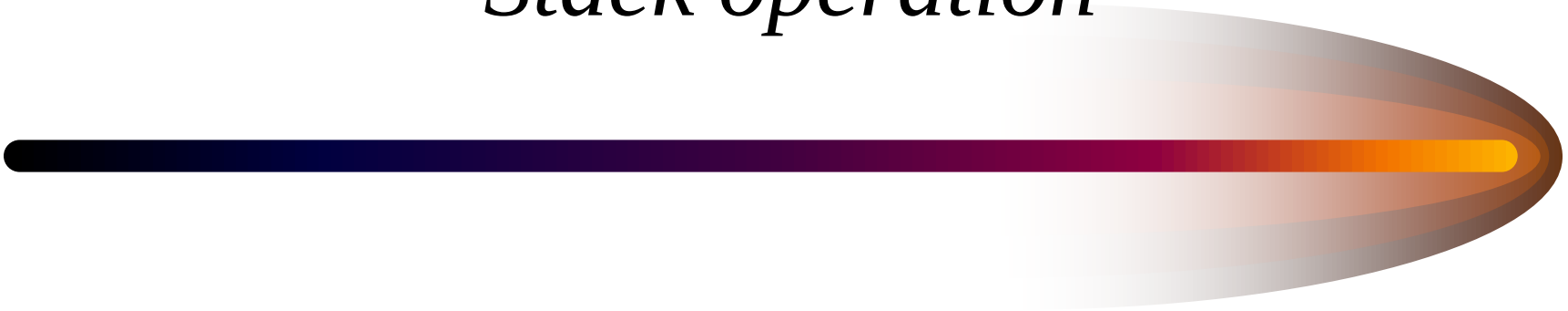
Microprocessor
Stack operation



By

P. Sridhar

Associate Professor, Dept. of ECE

MIST.

Date: 23/06/2018

Outline

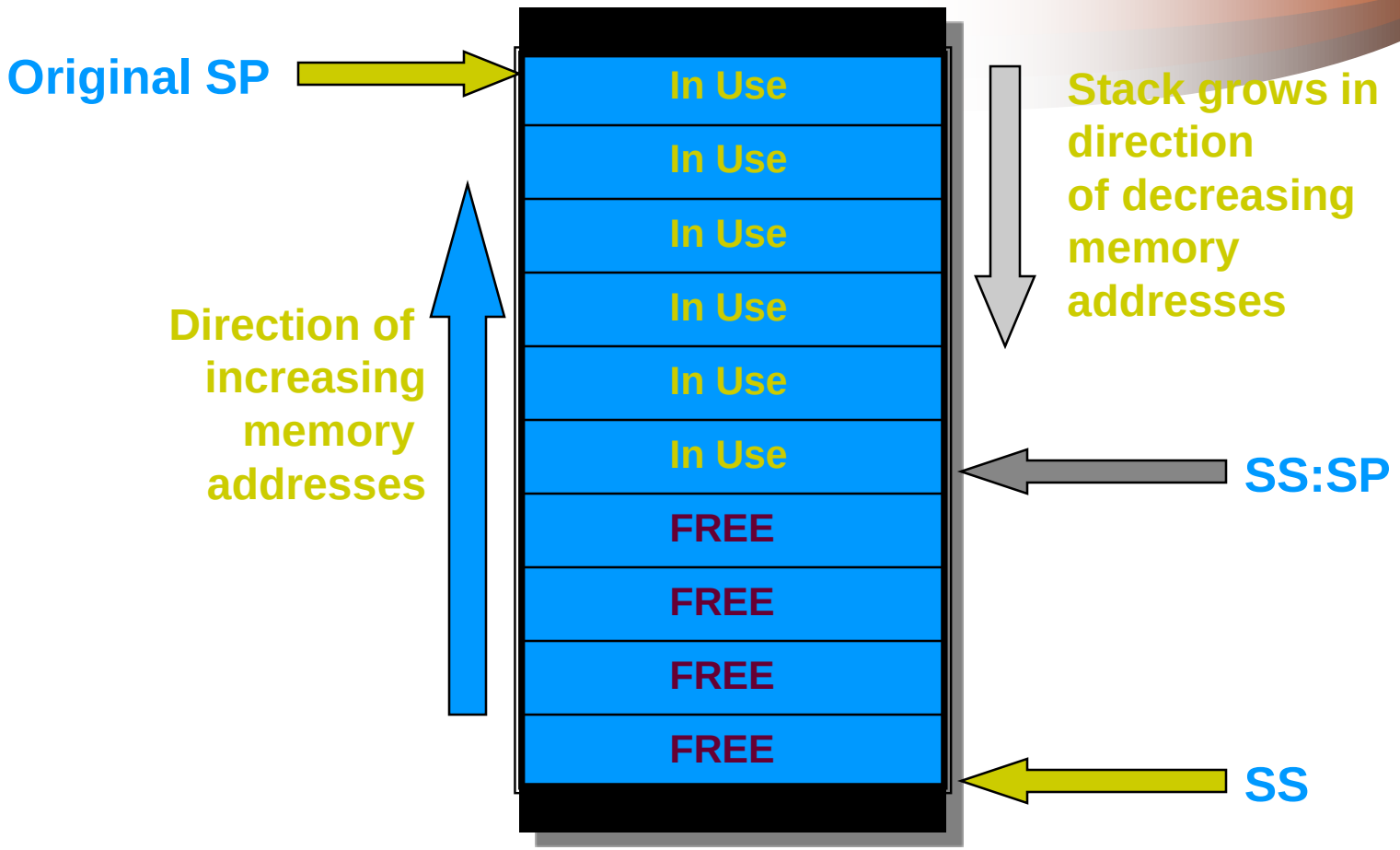
- Program Stack
- PUSH & POP instructions

Stack Key Characteristics

- Used to store temporary data during program execution
- Used to store the return address when a procedure is called
- One point of access - the top of the stack
- A program (memory)stack is always operated as Last-In-First-Out (LIFO) storage, i.e., data are retrieved in the reverse order to which they were stored

- Instructions that directly manipulate the stack
  - **PUSH** - place element on top of stack
  - **POP** - remove element from top of stack

Stack Implementation in Memory



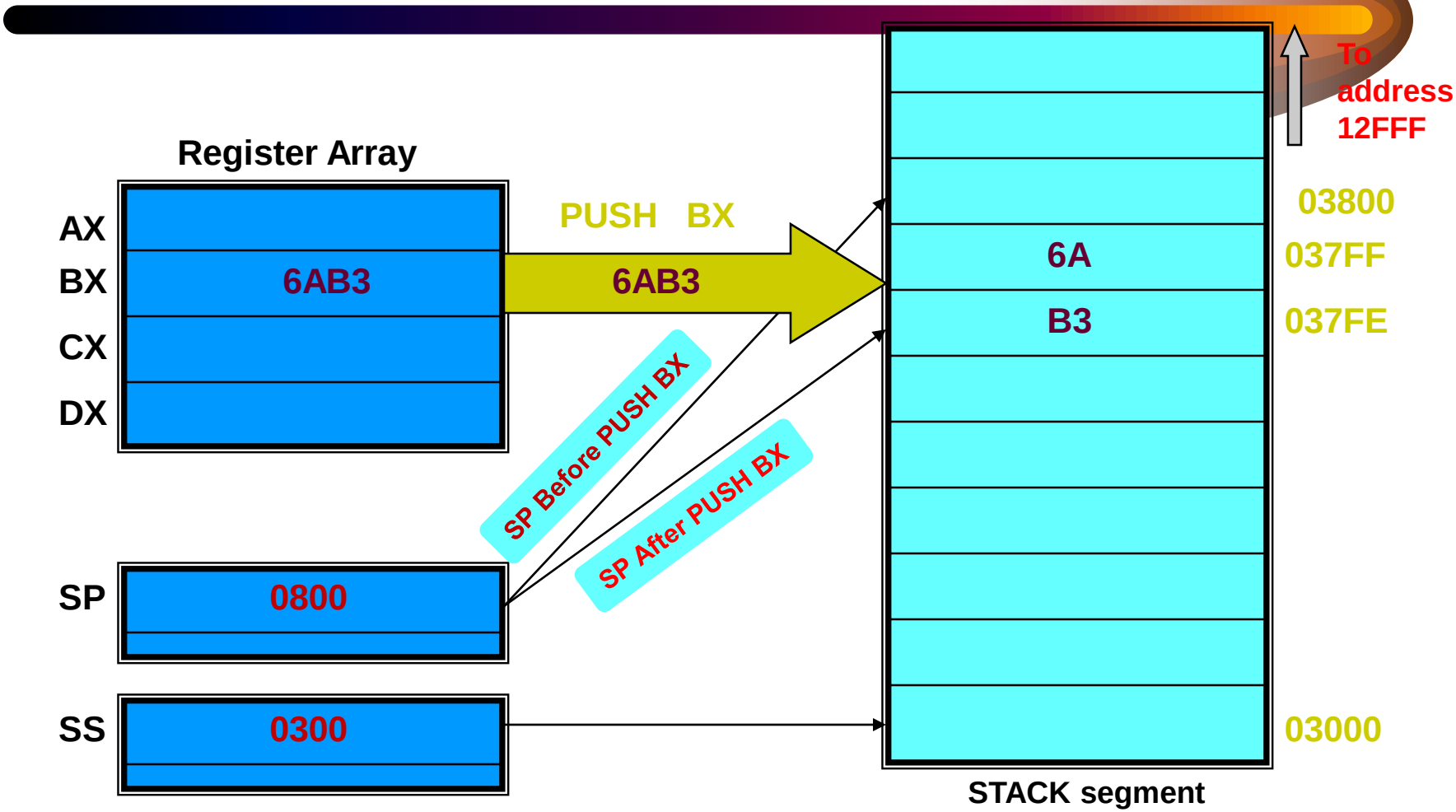
Stack Implementation in Memory (cont.)

- SS - Stack Segment
- SP (stack pointer) always points to the top of the stack
  - SP initially points to top of the stack (high memory address).
  - SP decreases as data is PUSHed

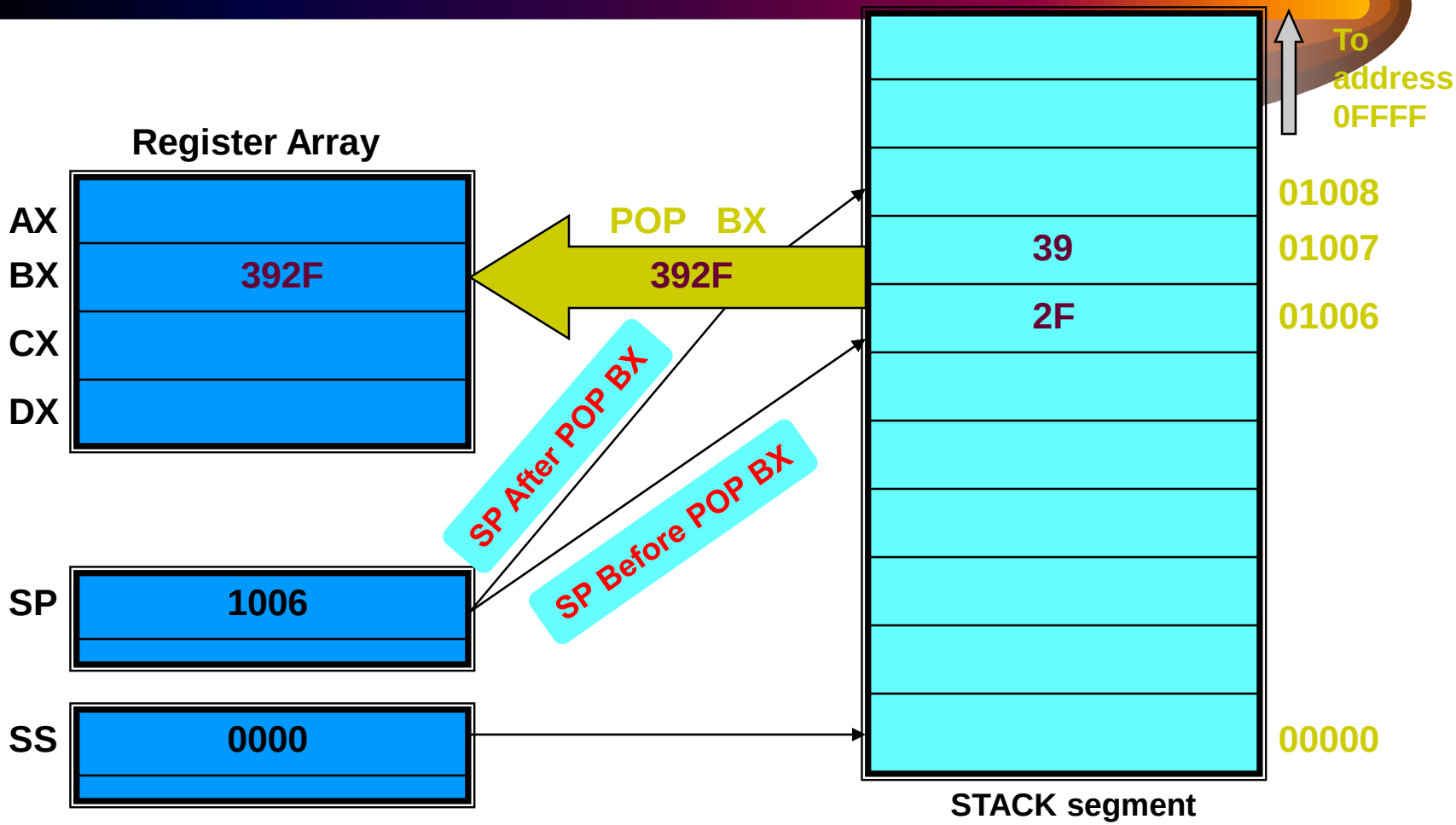
    PUSH AX ==> SUB SP, 2 ; MOV [SS:SP], AX
  - SP increases as data is POPed

  - POP AX ==> MOV AX, [SS:SP] ; ADD SP, 2

PUSH Instruction Example



POP Instruction Example



PUSH & POP (More I)

- PUSH and POP always store or retrieve words of data (never bytes) in the 8086-80286 microprocessor
- The 80386/80486 allow words or double words to be transferred to and from the stack
- The source of data for PUSH
  - any internal 16-bit/32-bit register, immediate data, any segment register, or any two bytes of memory data
- The POP places data into
  - internal register, segment register (except CS), or a memory location

PUSH & POP (More II)

- The 80286 and later microprocessors there is also PUSHA and POPA to store and retrieve, respectively the contents of internal register set (AX, CX, DX, BX, SP, BP, SI, and DI)

- Stack initialization, example:
  - assume that the stack segment resides in memory locations 10000h-1FFFFh
  - the stack segment (SS)is loaded with 1000h
  - the SP is loaded with 0000h - to start the stack at the top of the 64K…**WHY?**

The Stack Use

- To store
  - registers
  - return address information while procedures are executing
  - local variables that procedures may require

- To pass parameters to procedures

Temporary Register Storage

- PUSH and POP, for registers to preserve their value

  Example:

```
PUSH    AX              ; Place AX on the stack
PUSH    BX              ; Place BX on the stack
  ...
< modify contents of
  Registers AX & BX >
  ...
POP     BX              ; Restore original  value of BX
POP     AX              ; Restore original  value of AX
```

Thank you