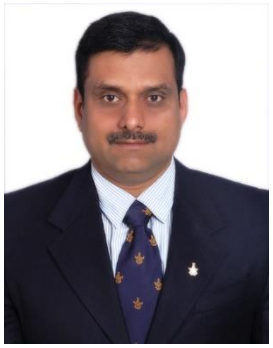
Microprocessors and Interfaces: 2021-22
Lecture 7
8086 Instructions Set : Part-1

By Dr. Sanjay Vidhyadharan



Types of Instructions

- Instructions with two operands (source and destination)
  -(R R, R M, R Idata, M Idata, but not M M)
- Instructions with one operand (source or destination)
  -(R , M, but not Idata)
- Instructions without any operand

Types of Instructions

- Data Transfer Instructions
- Arithmetic Instructions
- Logical Instructions
- Branch and Program control Instructions

Data Transfer Instructions

1. Data Transfer Instructions

- MOV destination, source
- XCHG destination, source
- XLAT
- PUSH source
- POP destination
- IN Reg, Port address
- OUT Port address , Reg
- LEA 16 bit register, memory
- LDS 16 bit register, memory
- LES 16 bit register, memory
- LAHF
- SAHF
- PUSHF
- POPF

1. Data Transfer Instructions

- General Purpose Data Transfer

  (MOV, XCHG, XLAT, PUSH, POP)
- Input / Output Data Transfer

  (IN, OUT)
- Address Object Data Transfer

  (LEA, LDS, LES)
- Flag Transfer Data Transfer

  (LAHF, SAHF, PUSHF, POPF)

1. Data Transfer Instructions

- **MOV DST, SRC**

  - Copies the content of source to destination
  - No Flags Affected
  - Size of source and destination must be the same
  - Source can be register, memory, or immediate data
  - Destination can be register or memory location

Different MOV options

R ← M

M ← R

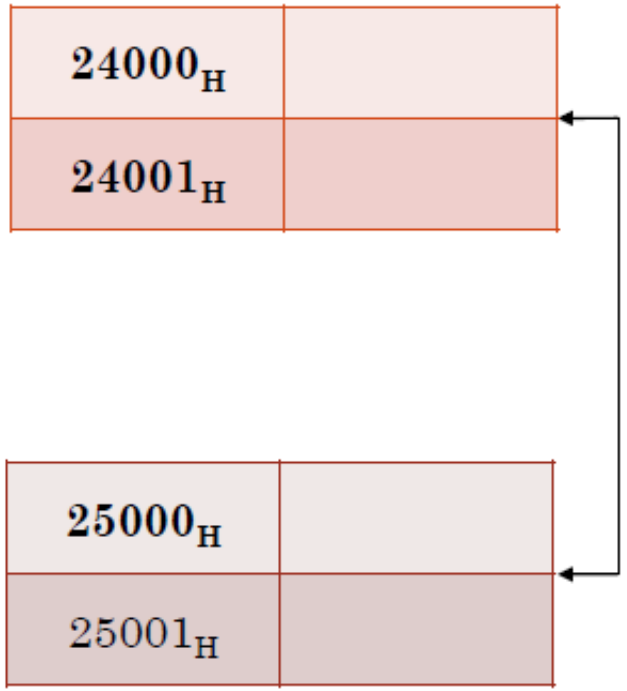
R ← R

M ← I

R ← I

MOV Instructions

Example 1: Swap the word at memory location 24000_H with 25000_H



MOV Instructions

MOV AX, 2000_H

MOV DS, AX

MOV SI, 4000_H

MOV DI, 5000_H

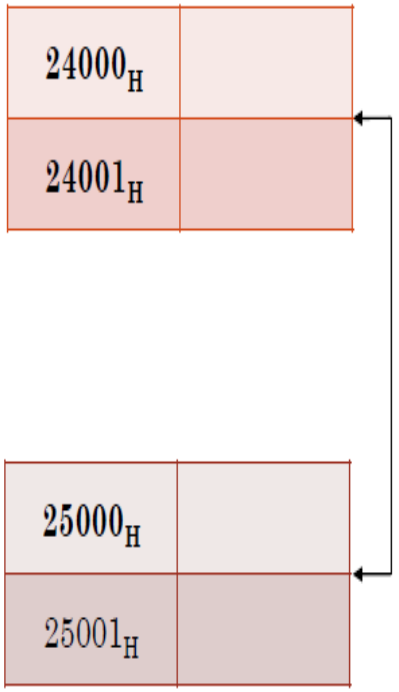
MOV BX, [SI]

MOV DX, [DI]

MOV [SI], DX

MOV [DI], BX

- Initialise Segment Register
- Initialise Offset Registers
- Transfer data from reg to mem temporarily
- Store back the data in mem



XCHG Instructions

- XCHG : (exchange)switches the contents of the source and destination operands (byte or word).
- Can not exchange the contents of two memory locations directly.
- Memory location can be specified as the source or destination.
- Segment registers can not be used.

XCHG reg , mem

Machine code format

Example

XCHG AX,BX

Before execution AX = 0001H, BX = 0002H

After execution AX = 0002H, BX = 0001H

XLAT

- XLAT (translate) replaces a byte in the AL register with a byte from a 256 byte, user-coded translation table.

- Register BX is assumed to be pointed to the beginning of the table (i.e. beginning location of the table)

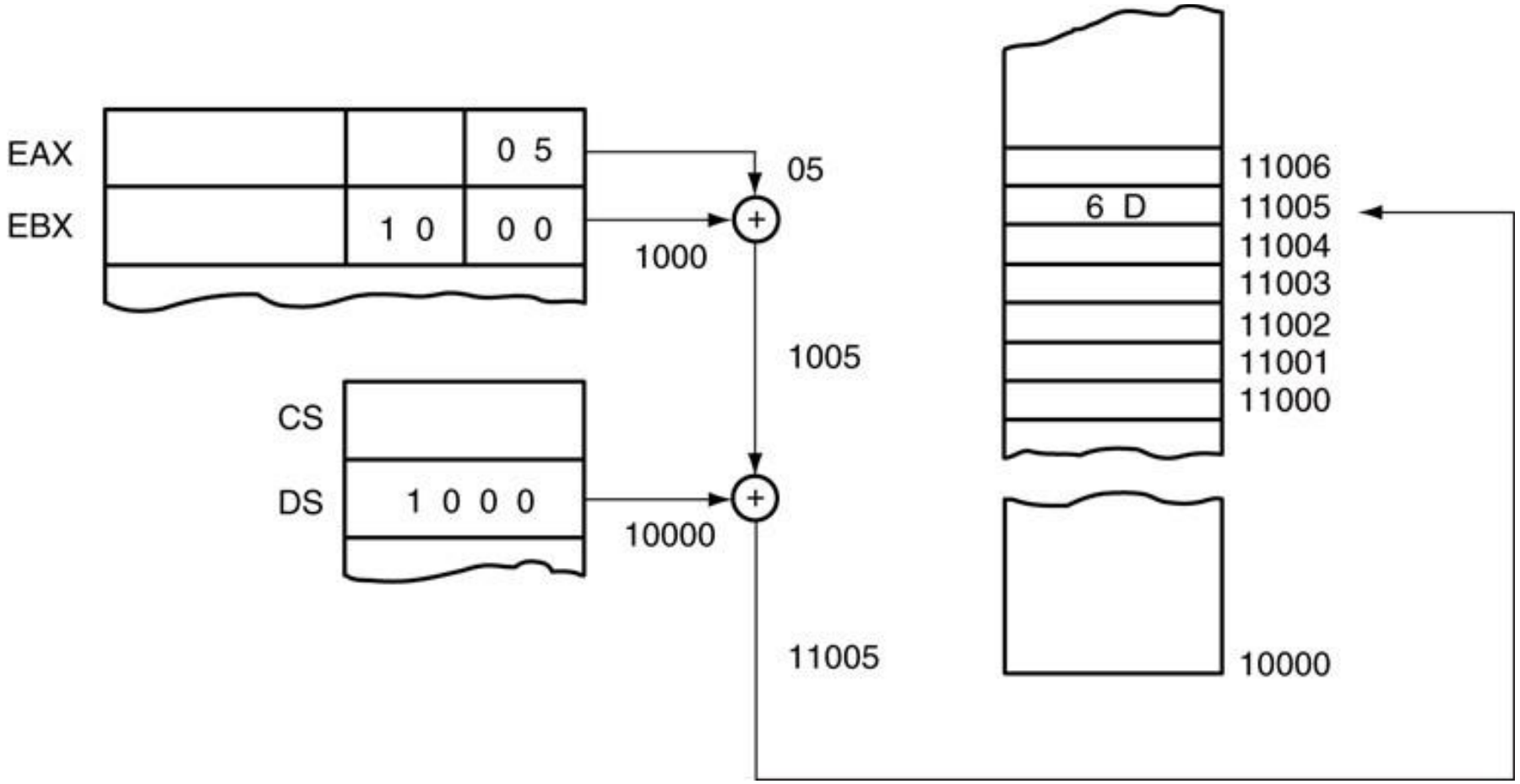
- Byte AL is used index in to the table (AL = 0).
- AL is replaced with byte at location [BX]+AL.

XLAT

11010111

Machine code format

XLAT


EAX
EBX
0 5
1 0
0 0
05
1000
1005
CS
DS
1 0 0 0
10000
11005
6 D
11006
11005
11004
11003
11002
11001
11000
10000

Example(contd)….

DS = 2000
Assume BX is pointing beginning location of the table (i.e. BX = 0000 H).

```
MOV AL, 05
XLAT
```

After execution AL is copied with byte located by BX + AL = 0000 H + 05 H = 0005 H.
PA = DS*10 H+0005 H=20005 H
(i.e. content at location 20005 H is copied into AL = 6D H)

```
MOV AL, 07
XLAT
```

After execution what is the value of AL = ?

20000 H	00	3FH
20001 H	01	06 H
20002 H	02	5B H
20003 H	03	4F H
20004 H	04	66 H
20005 H	05	6D H
20006 H	06	7D H
20007 H	07	27 H
20008 H	08	7F H
20009 H	09	6F H

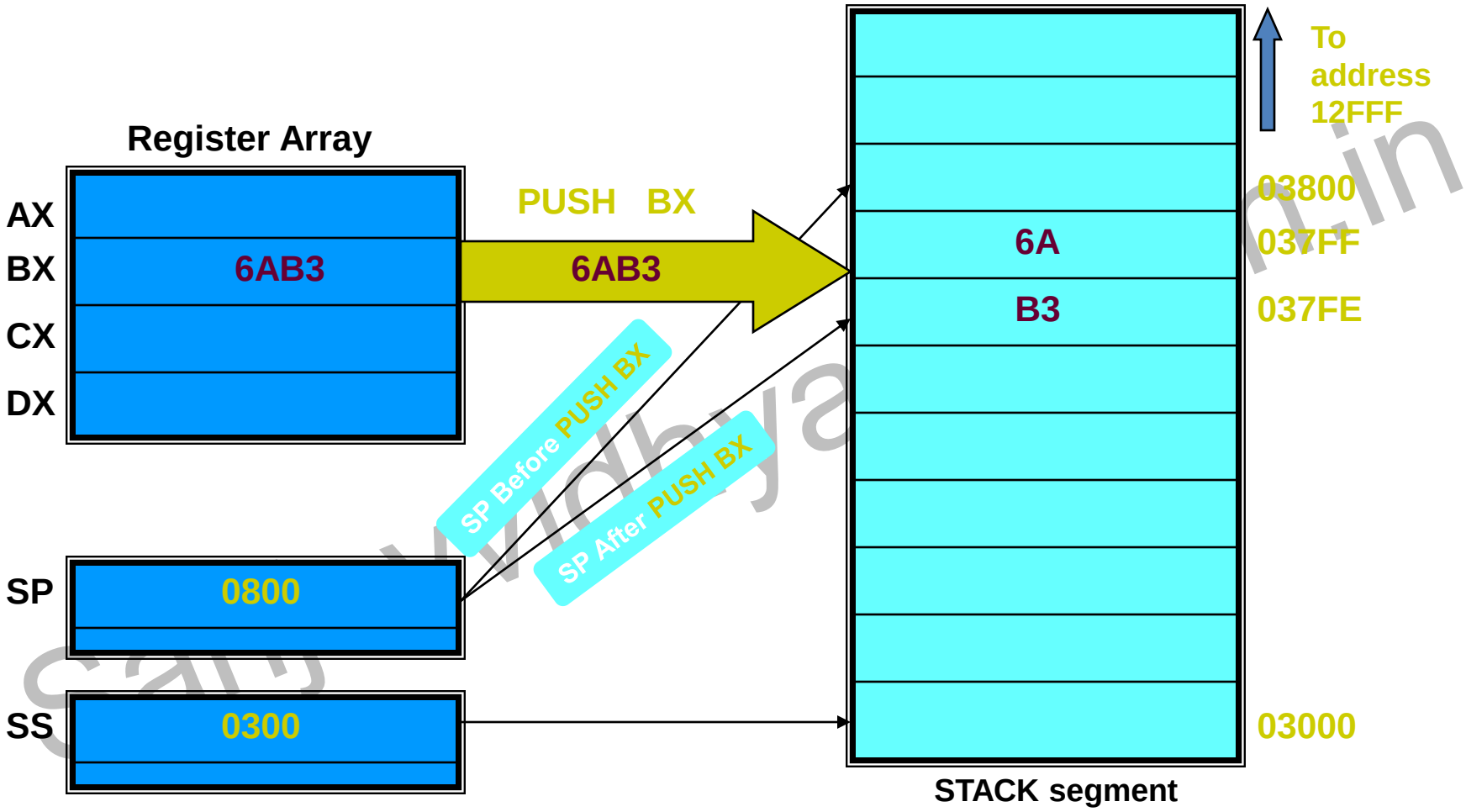
PUSH

- It is used for storing data into temporary memory (stack).
- Pushes the contents of the specified register / memory location on to the stack.
- The stack pointer is decremented by 2 , after each execution.
- The source of the word can be a general purpose register, a segment register, or memory.
- The SS and SP must be initialized before this instruction.
- No flags are affected.

PUSH

- Store data into LIFO stack memory
- 2/4 bytes involved
- Whenever 16-bit data pushed into stack
- Contents of SP register decremented by 2
- MSB moves into memory [SP-1]
- LSB moves into memory [SP-2]

PUSH



PUSH

Push data from

- Registers/Segment Register
- Memory
- Flag Register

PUSH

- Always transfers 2 bytes of data to the stack;
  - 80386 and above transfer 2 or 4 bytes

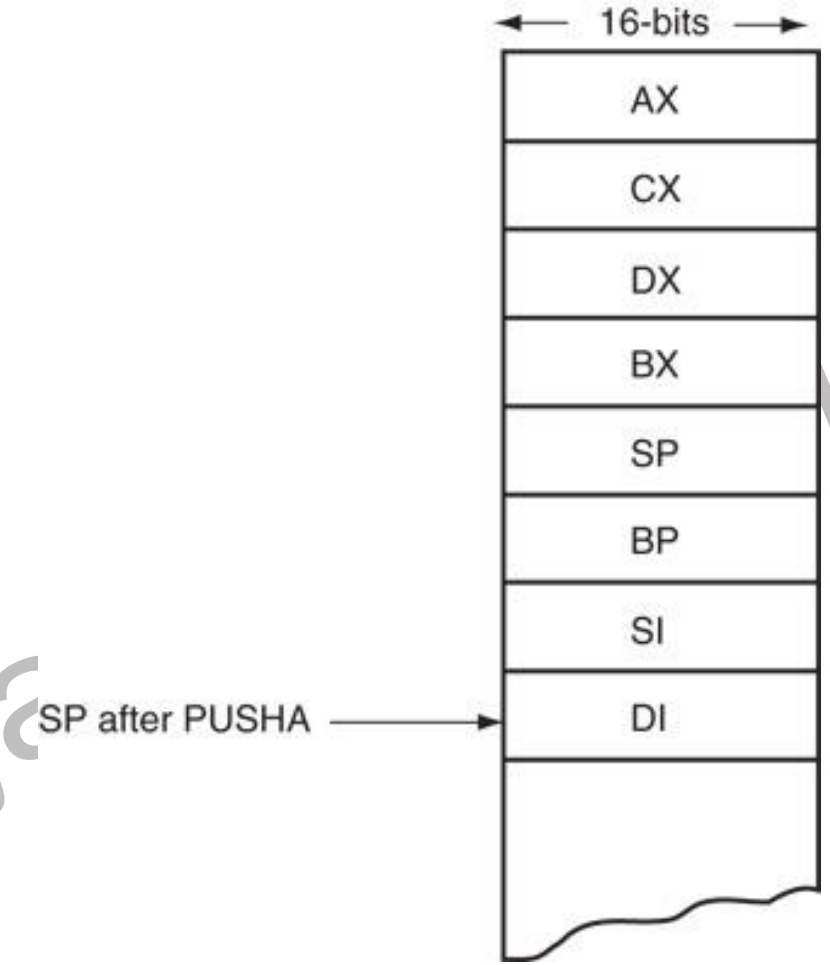
- PUSHA instruction copies contents of the internal register set, except the segment registers, to the stack.

- PUSHA (**push all**) instruction copies the registers to the stack in the following order: AX, CX, DX, BX, SP, BP, SI, and DI.

PUSH

- PUSHA instruction pushes all the internal 16-bit registers onto the stack.
- requires 16 bytes of stack memory space to store all eight 16-bit registers
- After all registers are pushed, the contents of the SP register are decremented by 16.
- PUSHAD instruction places 32-bit register set on the stack in 80386 - Core2.
  - PUSHAD requires 32 bytes of stack storage

PUSHA instruction, showing the location and order of stack data.



PUSH Instruction

- PUSHF (**push flags**) instruction copies the contents of the flag register to the stack.

- PUSHAD instructions push the contents of the 32-bit register set in 80386 - Pentium 4.

- PUSHA instructions do not function in the 64-bit mode of operation for the Pentium 4

PUSH

Example-PUSH operation

PUSH AX
PUSH EBX
PUSH DS
PUSH WORD PTR[BX]
PUSHF
PUSHFD
PUSHA
PUSHAD
PUSH 16-imm
PUSHD 32-imm

Directives BYTE PTR, WORD PTR, DWORD PTR

•mov [SI], al ; Store a byte-size value in memory location pointed by SI suggests that an 8-bit quantity should be moved because AL is an 8-bit register.

•When instruction has no reference to operand size,
mov [SI], 5 ; Error: operand must have the size specified

•To get around this instance, we must use a pointer directive, such as
mov BYTE PTR [ESI], 5 ; Store 8-bit value
mov WORD PTR [ESI], 5 ; Store 16-bit value
mov DWORD PTR [ESI], 5 ; Store 32-bit value

POP

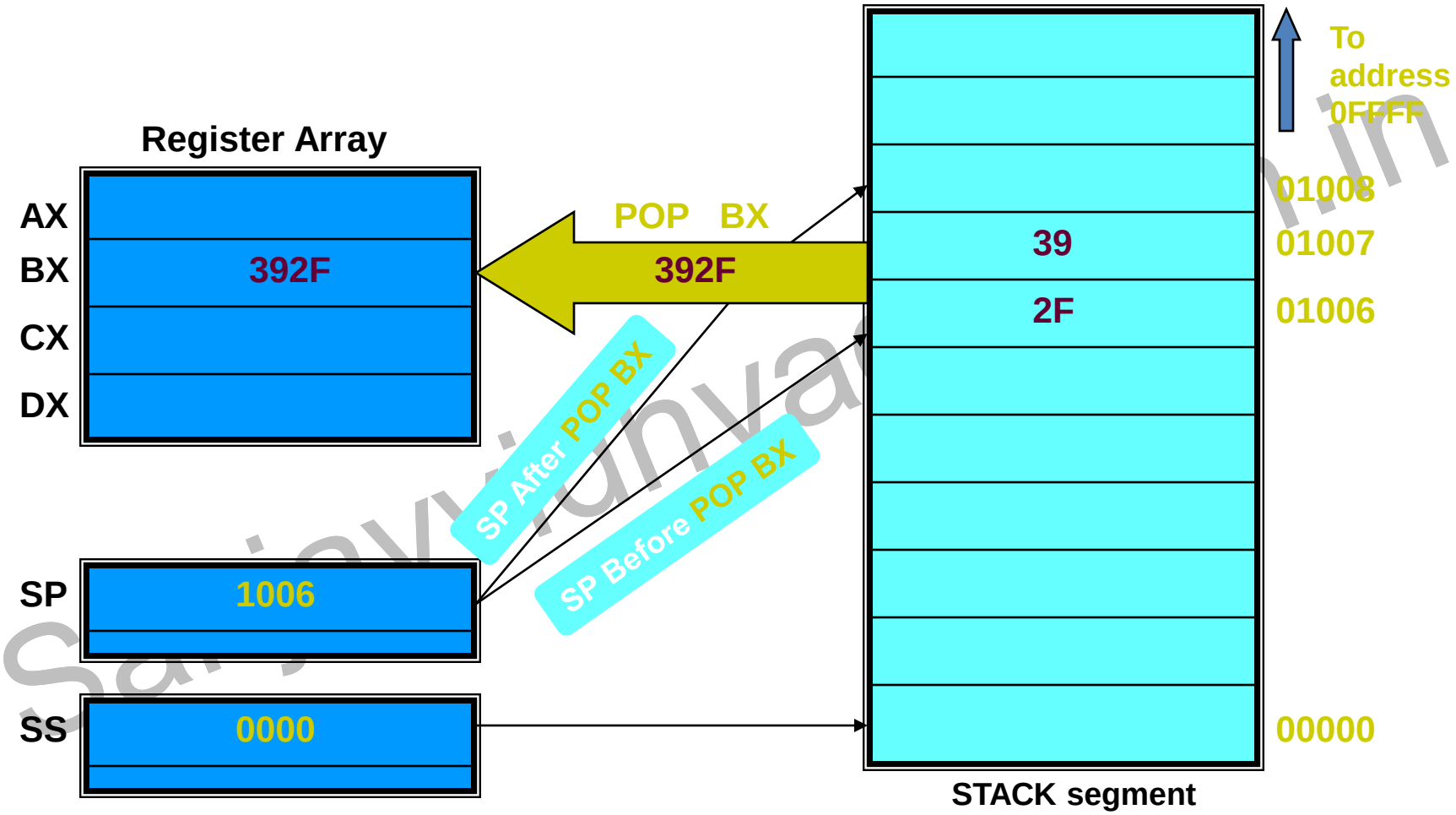
- Performs the inverse operation of PUSH
- The POP instruction removes a word from the top of the stack to the specified 16-bit register or memory location.
- The stack pointer is incremented by 2 after the execution.
- Ex: POP CX

  - copies a word from the top of stack to the CX register and increment SP by 2.
- not available as an immediate POP

POP

- POPF (pop flags) removes a 16-bit number from the stack and places it in the flag register;
  - POPFD removes a 32-bit number from the stack and places it into the extended flag register
- POPA (pop all) removes 16 bytes of data from the stack and places them into the following registers, in the order shown: DI, SI, BP, SP, BX, DX, CX, and AX.
  - reverse order from placement on the stack by PUSHA instruction, causing the same data to return to the same registers

POP



PUSH F

- Pushes flag register on to the stack; First the upper byte and then the lower byte is pushed on to it. The SP (stack pointer) is decremented by 2 for each PUSH operation.

- The FLAGS themselves are not affected.

PUSHF

10011100

Machine code format

POPF

- Transfers a word from the current top of the stack to the FLAG register and increments the SP by 2.
- All the flags will be affected.
- PUSHF and POPF are used when there is a subprogram.

POPF

1 0 0 1 0 0 0 0

Machine code format

Thank You