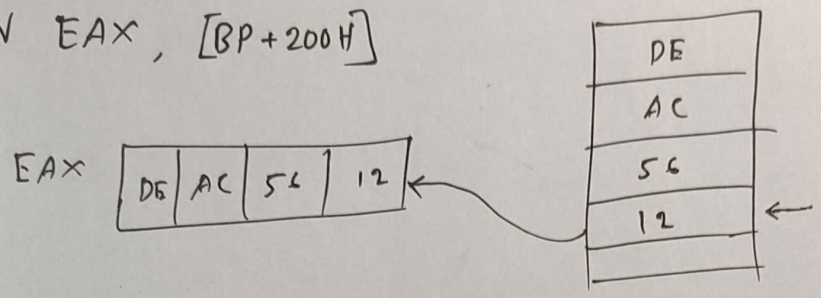
CSE 331 / L-6 / 29.02.2024 /

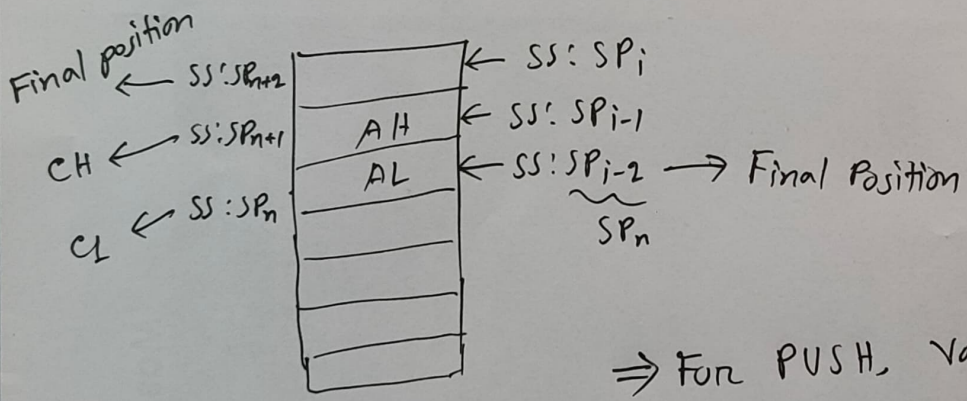
* MOV EAX, [BP+200H]



* Stack memory addressing mode:

  - followed LIFO (last in, first out)

* PUSH AX
  POP CX → must be word size, 16 bit



⇒ For PUSH, value of SP will decrease first then PUSH will execute

⇒ For POP, ~~valu~~ ~~val~~ content will pop first, then value of SP will increase by 1.

* Register mode stack operation:

⇒ PUSH BX
POP CX

PUSH CS ⇒ Higher bit will copied first then lower bit
POP CS ✗ → CS can't be ~~ov~~ override
Segment Register mode

Memory mode stack operation:

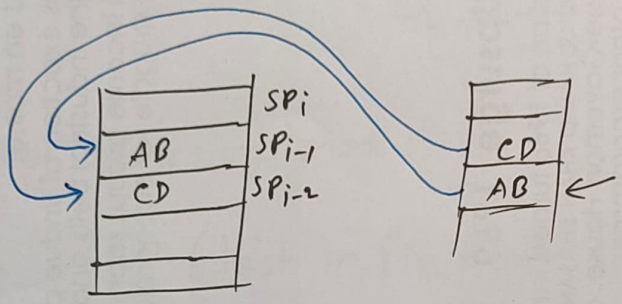
PUSH WORD PTR [BX] ⇒ memory to memory transfer

(BX → offset)

Defined the size of memory to be copied

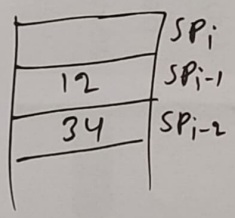
Stack stored in memory

Source is also referring memory



Immediate mode stack operation:

PUSH 1234H



POP 1234H ✗

Destination can't be a constant

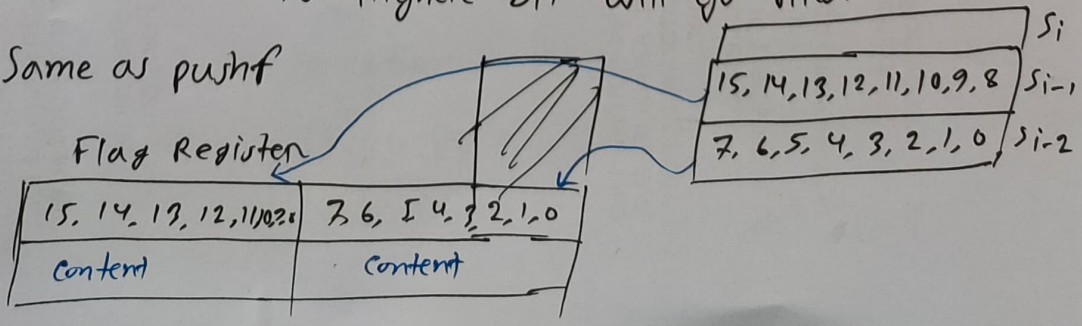
flags mode stack Operation:

Slide Page - 81, 85

PUSHF ⇒ Fully opcode, no operand

- Here, F stands for Flags register of 16 bit
- ~~First~~ Higher bit will go first

POP F ⇒ Same as pushf



- All Register mode stack operation:

  ⇒ PUSHA

  Push sequence

  ⇒ AX, CX, DX, BX, SP, BP, SI, DI

  ⇒ POPA

  pop sequence ⇒ Reverse of Push

- Becareful about the final value of SP

  Let, initial value of SP = 2345 H

  after PUSHA instruction,

  ~~value of SP will decrease by 8 times~~

  value of SP will decrease 16 times. because of 8 Register.

  ↳ Here 16 is decimal count Remember, 0-9, decimal and hexa same. but after 9 in hexa, it is A. So we need to convert 16 to hexadecimal, which is 10.

  Slide Page - 90, 92, 95, 97

  ∴ Final value of SP = (2345 - 10) H

  = 2335 H

- Base relative plus index - usefull for addressing two dimensional memory array

~~Lecture~~

Lecture Slide 4,5

Chapter - 4

Data movement instruction

- LEA ⇒ Load Effective Address
  - used to calculate the offset and store it in a location.

```
LEA AX, NUMB
MOV AX, OFFSET NUMB
```

} Both instruction do the same operation.

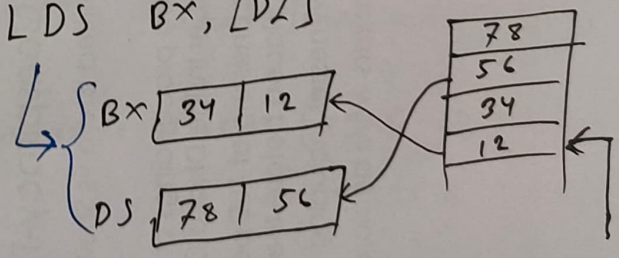
⇒ MOV BX, [DI] ⇒ copied the data stored in a memory location referred by the offset DI

⇒ LEA BX, [DI] ⇒ copied the value or content of DI Register or the entire offset value inside the third bracket.

- LDS ⇒ Load Data Segment
  - do two operation at a time

```
MOV BX, [DI]
MOV DS, [DI + 2H]
```

} LDS BX, [DI]



- LSS ⇒ Load Stack Segment
  - same as LDS

```
MOV BX, [SP]
MOV SS, [SP+2H]
```

} LSS BX, [SP]