Fall 2018/19 – Lecture Notes # 4

- Pushing and Popping Operations (Stack)
- Flag Registers and bit fields

Introduction to Program Segments

- ***Addressing in Stack Segment***

- ***What is a stack, and why is it needed?***
  The stack is a section of RAM used by the CPU to store information temporarily. CPU needs this storage area since there are only limited number of registers.

- ***How stacks are accessed***
  SS (stack segment) and ***SP*** (stack pointer) must be loaded to access stack in the memory. Every register in the CPU (except segment registers and SP) can be stored in the stack and loaded from the stack.

- ***Logical Address*** **in Stack Segment** is represented by using **segment address in SS** register and **Offset Address in SP** register.

SS:SP

Introduction to Program Segments

- ***Stack: Pushing and Popping Operations***

- ***Push Instruction***

  Storing the CPU register in the stack is called a ***push***.

PUSH source; — ***Copy the content of source (16-bit register) into stack***

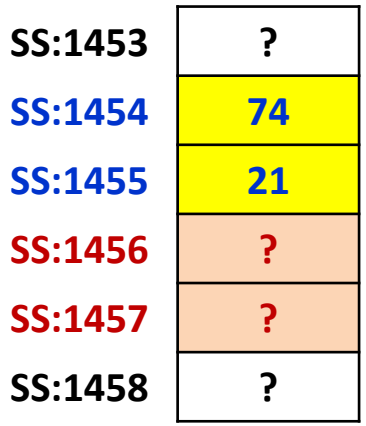
mnemonic operand

* SP register is decremented by 2

Example: Given that SP=1456H, what are the contents of AX, top of the stack and SP after the execution of the following instruction.

```
MOV  AX,2174H
PUSH AX
```

Solution: AX=2174H (stays the same), SP=1454H (decremented by 2),



Introduction to Program Segments

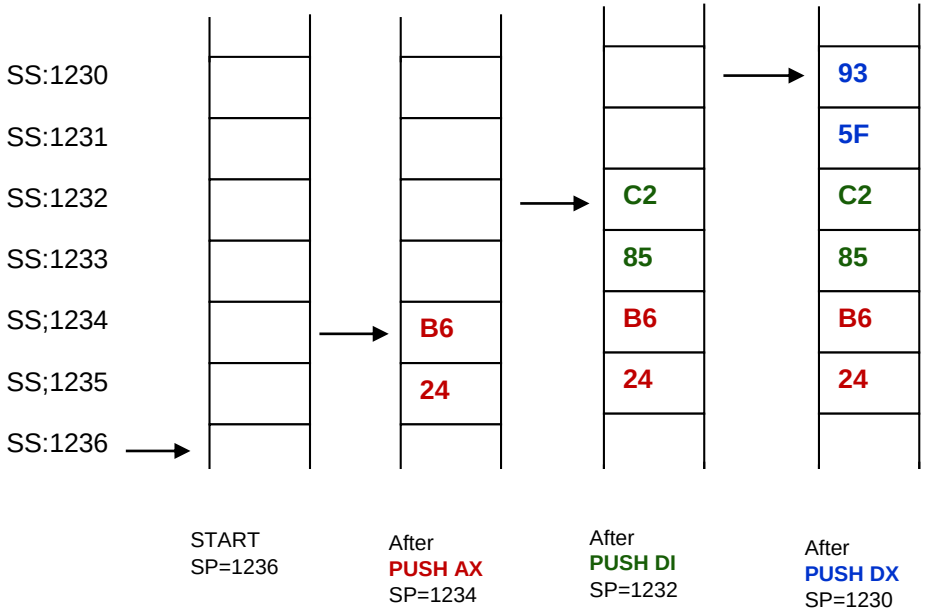
- ***Stack: Pushing and Popping Operations***

- ***Pushing onto the stack***

  Storing the CPU register in the stack is called a ***push***.

Example: SP=1236H, AX=24B6H, DI=85C2H, and DX=5F93H, show the contents of the stack as each of the following instructions is executed.

```
PUSH  AX
PUSH  DI
PUSH  DX
```



Solution:

Note that in 80x86 the lower byte of the register is stored to the lower address. **(Little Endian Convention)**

Introduction to Program Segments

- ***Stack: Pushing and Popping Operations***

- ***Pop Instruction***

  Loading the contents of the stack into the CPU register is called a ***pop***.

POP destination;

mnemonic *operand*

Copy the top of the stack into destination (16-bit register)

* SP register is incremented by 2

Example: Assume that SP=134AH and the illustration on the right shows the content of the top of the stack. What will be the content of AX and SP after of after the execution of the following instruction.

POP AX

Solution: AX=FC76H and SP=134CH (incremented by 2),

SS:1349	?
SS:134A	76
SS:134B	FC
SS:134C	?
SS:134E	?
SS:134F	?

Stack Segment

Introduction to Program Segments

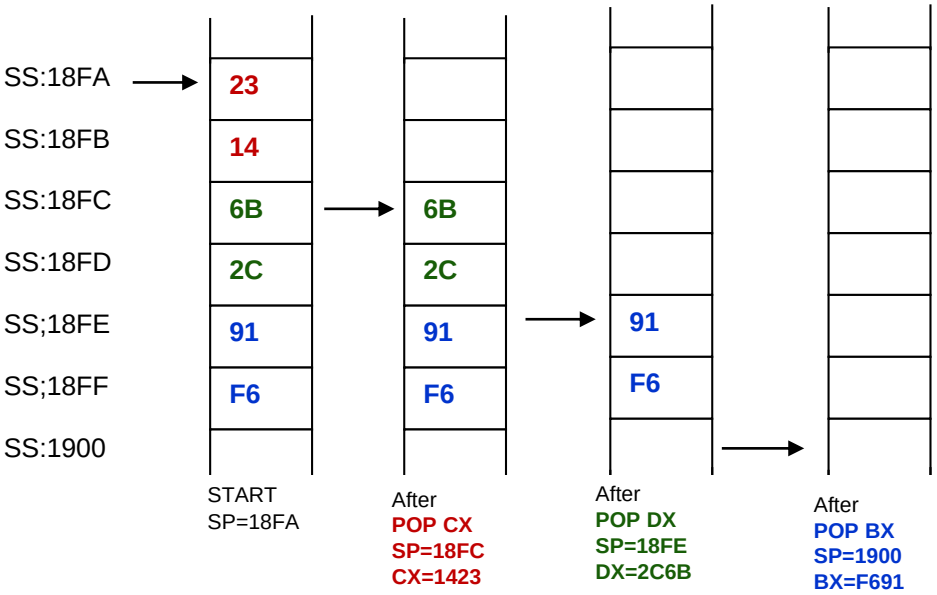
- ***Stack: Pushing and Popping Operations***

- ***Popping the stack***

  Loading the contents of the stack into the CPU register is called a ***pop***.

Example: Assume that the stack is shown below, and SP=18FAH, show the contents of the stack and registers as each of the following instructions is executed.

```
POP  CX
POP  DX
POP  BX
```



Solution:

Note that in 80x86 the byte in the Low address goes into the low byte. the byte in the high address goes into the high byte. **(Little Endian Convention)**

Introduction to Program Segments

- ***Logical vs. physical address of the stack***

- Calculating the physical address for the stack, the same principle is applied as was used for the code and data segments. Physical address depends on the value of stack segment (SS) register and the stack pointer (SP).

Example: If SS=3500H and SP:FFFEH

a) Calculate the physical address: 35000+FFFE = 44FFE
b) Calculate the lower range of the stack: 35000+0000 = 35000
c) Calculate the upper range of the stack segment: 35000+FFFF = 44FFF
d) Show the logical address of the stack: 3500:FFFE

Introduction to Assembly Language Programming

- ## The Flag Register (FR) and bit fields

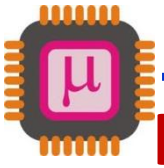
  - The **flag register** is a **16-bit register** sometimes referred as the ***status*** register. Although the register is 16-bit. Not all the bits are used.

  - **Conditional flags:** 6 of the flags are called the conditional flags, meaning that they indicate some condition that resulted after an instruction was executed. These 6 are: **CF, PF, AF, ZF, SF,** and **OF**.

  - The 16 bits of the flag registers:

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

R=	reserved	SF=	sign flag
U=	undefined	ZF=	zero flag
OF=	overflow flag	AF=	auxiliary carry flag
DF=	direction flag	PF=	parity flag
IF=	interrupt flag	CF=	carry flag
TF=	trap flag		



Introduction to Assembly Language Programming

- **The Flag Register (FR) and bit fields**

CF, the Carry Flag: This flag is set whenever there is a carry out, either from d7 after an 8-bit operation, or from d15 after a 16-bit data operation.

PF, the Parity Flag: After certain operations, the parity of the result's low-order byte is checked. If the byte has an even number of 1s, the parity flag is set to 1; otherwise, it is cleared.

AF, the Auxiliary Carry Flag: If there is a carry from d3 to d4 of an operation this bit is set to 1, otherwise cleared (set to 0).

ZF, the Zero Flag: The ZF is set to 1 if the result of the arithmetic or logical operation is zero, otherwise, it is cleared (set to 0).

SF, the Sign Flag: MSB is used as the sign bit of the binary representation of the signed numbers. After arithmetic or logical operations the MSB is copied into SF to indicate the sign of the result.

TF, the Trap Flag: When this flag is set it allows the program to single step, meaning to execute one instruction at a time. Used for debugging purposes.

IF, Interrupt Enable Flag: This bit is set or cleared to enable or disable only the external interrupt requests.

DF, the Direction Flag: This bit is used to control the direction of the string operations.

OF, the Overflow Flag: This flag is set whenever the result of a signed number operation is too large, causing the high-order bit to overflow into the sign bit.

Introduction to Assembly Language Programming

- ***Manipulating the Flag Register***

- There are two instructions that can be used to use/change the content of the flag register.
- **PUSHF Instruction**

PUSHF ; ***Copy the flag register into stack***

* SP register is decremented by 2

Example: Copy the content of the flag register into register AX.

```
PUSHF     ; copy the content of Flag register into the stack.
POP AX    ; copy from the stack into AX
```

Introduction to Assembly Language Programming

- ***Manipulating the Flag Register***

- There are two instructions that can be used to use/change the content of the flag register.
- **POPF Instruction**

POPF ; ***Copy from the stack into the flag register***

* SP register is incremented by 2

Example: Clear the flag bits. (Make the flag bits to be all 0)

```
MOV AX,0000H
PUSH AX  ; now top of the stack contains 16-bit 0.
POPF ; copy the content of stack into flag register.
```