Microprocessor and Assembly Language CSC-321

Sheeza Zaheer

Lecturer

COMSATS UNIVERSITY ISLAMABAD

LAHORE CAMPUS

The Stack

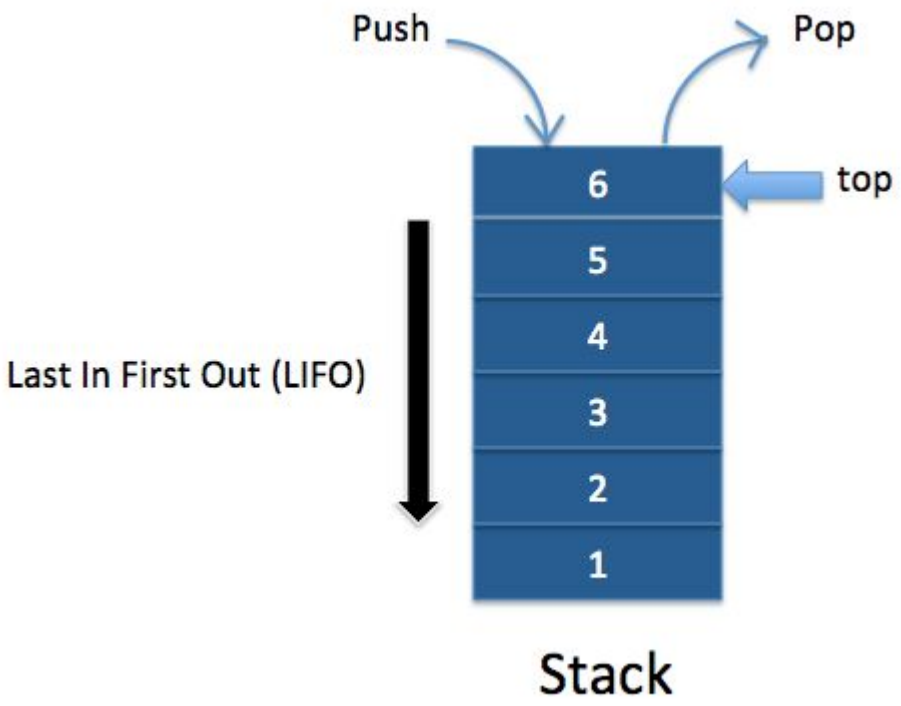
OUTLINE

- **Stack**
  - Introduction
  - Syntax
  - PUSH and POP instructions
  - Applications

- **References**
  - **Chapter 8, Section 8.1 & 8.2,** Ytha Yu and Charles Marut, "Assembly Language Programming and Organization of IBM PC

STACK

- One dimensional data structure
- Items are added and removed from one end of the structure i.e. Top of the Stack
- Last In First Out Structure



STACK

- .STACK directive used to reserve a block of memory for stack. The syntax is:

  .STACK 100h

- When program is assembled and load in memory:
  - SS (Stack Segment) Register: holds stack Segment Address
  - SP (Stack Pointer) Register: initialized to the value specified with the .STACK directive, represents empty stack position
  - When stack is not empty, SP represents the top of the Stack
- Stack grows toward the beginning of the memory

00F8	
00FA	
00FC	
00FE	
0100	

PUSH And PUSHF

- PUSH: Used to add a new source (16-bit register or memory word) on stack
- Syntax:

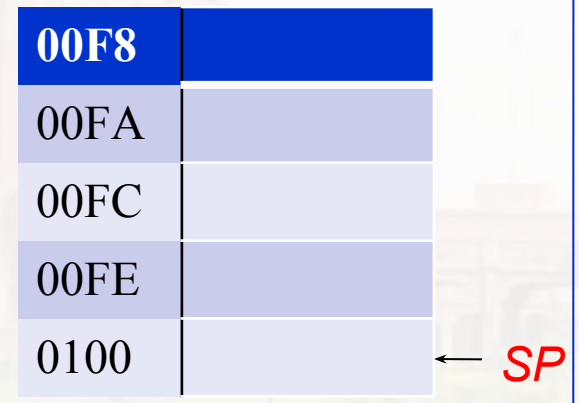
  PUSH *source*
- Execution of PUSH causes:
  - SP is decreased by 2
  - A copy of source contents is moved to the address specified by SS:SP.
  - Source remains unchanged

PUSHF

- PUSHF has no operand and it pushes the contents of FLAG register onto the stack.
  - Syntax:

    PUSHF

PUSH - Example

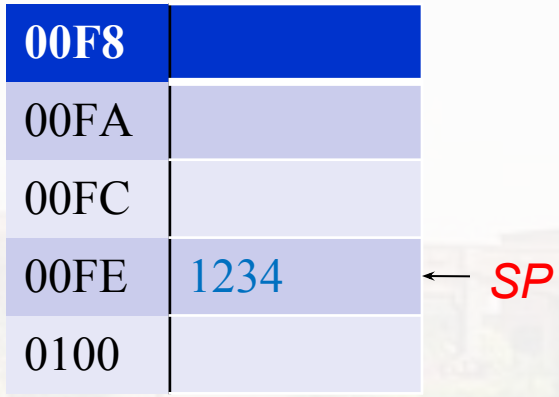


SP 0100

AX 1234

BX 5678

STACK EMPTY

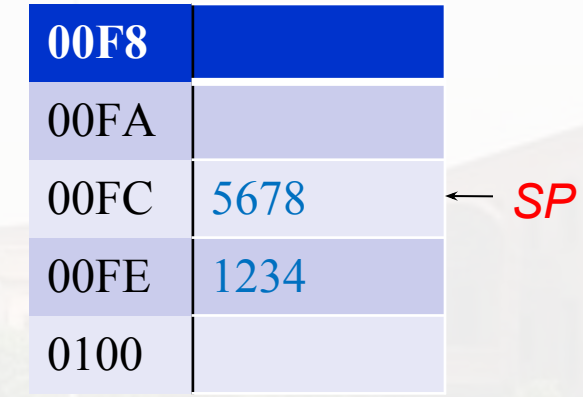


SP 00FE

AX 1234

BX 5678

After PUSH AX



SP 00FC

AX 1234

BX 5678

After PUSH BX

POP

- POP: Used to remove top item from stack to destination (i.e. a 16-bit register or memory word).
- Syntax:

  POP *destination*
- Execution of POP causes:
  - The contents of SS:SP (top of the stack) is moved to the destination.
  - SP is increased by 2

POPF

- POPF has no operand and pops the top of the stack into FLAG register.
- Syntax:

  POPF

POP - Example

Offset	Stack	
00F8		
00FA		
00FC	5678	← SP
00FE	1234	
0100		

SP	00FC
CX	FFFF
DX	0001

Before POP

Offset	Stack	
00F8		
00FA		
00FC	5678	
00FE	1234	← SP
0100		

SP	00FE
CX	5678
DX	0001

After POP CX

Offset	Stack	
00F8		
00FA		
00FC	5678	
00FE	1234	
0100		← SP

SP	0100
CX	5678
DX	1234

After POP DX (Stack Empty)

STACK Application

- To store and restore registers
- To reverse an input

Saving & Storing Registers


```
01  title Saving Registers
02
03  org 100h
04
05  .data
06  Message db "This is a Message.$"
07
08  .code
09  main proc
10
11       mov ax,20H
12       mov dx,20H
13
14       push ax                    ;save ax
15       push dx                    ;save dx
16
17       mov ah,9                   ;function display string
18       mov dx,offset message      ;dx points to the string
19       INT 21H                    ;call DOS
20
21       pop dx                     ;restore dx
22       pop ax                     ;restore ax
23
24       mov ax,4C00h
25       INT 21H
26
27  main endp
28
29  end main
30
```

Algorithm to Reverse an Input

- Convert the following algorithm to Assembly Language code:

```
Display a "?"
Initialize count to 0
Read a character
WHILE character is not carriage return Do
  push character onto the stack
  increment count
  read a character
END_WHILE
Go to a new line
For count times Do
  pop a character from the stack
  display it
END_FOR
```

For Practice

- Exercise Ch#8: Q2, Q3, Q6, and Q8.