**PMAS Arid Agriculture University Rawalpindi**

**University Institute of Information Technology**

**LAB MANUAL -X**

**Class/Program: BS (CS)** **Course: COAL (CS530)**



**Objectives:**

1. **Introduction to stack**
2. **PUSH Instruction**
3. **POP Instruction**
4. **LIFO Algorithm**
5. **Examples of Stack**

**Introduction to Stack**

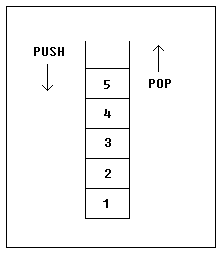
Stack is an area of memory for keeping temporary data. Stack is used by **CALL** instruction to keep return address for procedure, **RET** instruction gets this value from the stack and returns to that offset. Quite the same thing happens when **INT** instruction calls an interrupt, it stores in stack flag register, code segment and offset. **IRET** instruction is used to return from interrupt call.  
  
We can also use the stack to keep any other data,  
there are two instructions that work with the stack:  
  
**PUSH** - stores 16 bit value in the stack.  
  
**POP** - gets 16 bit value from the stack.

|  |
| --- |
| Syntax for **PUSH** instruction:  PUSH REG  PUSH SREG  PUSH memory  PUSH immediate  **REG**: AX, BX, CX, DX, DI, SI, BP, SP.  **SREG**: DS, ES, SS, CS.  **memory**: [BX], [BX+SI+7], 16 bit variable, etc...  **immediate**: 5, -24, 3Fh, 10001101b, etc... |

|  |
| --- |
| Syntax for **POP** instruction:  POP REG  POP SREG  POP memory  **REG**: AX, BX, CX, DX, DI, SI, BP, SP.  **SREG**: DS, ES, SS, (except CS).  **memory**: [BX], [BX+SI+7], 16 bit variable, etc... |

Notes:

* **PUSH** and **POP** work with 16 bit values only!
* Note: **PUSH immediate** works only on 80186 CPU and later!

The stack uses **LIFO** (Last In First Out) algorithm,  
this means that if we push these values one by one into the stack:  
**1, 2, 3, 4, 5**  
the first value that we will get on pop will be **5**, then **4**, **3**, **2**, and only then **1**.  
  
  
  
It is very important to do equal number of **PUSH**s and **POP**s, otherwise the stack maybe corrupted and it will be impossible to return to operating system. As you already know we use **RET** instruction to return to operating system, so when program starts there is a return address in stack (generally it's 0000h).  
  
**PUSH** and **POP** instruction are especially useful because we don't have too much registers to operate with, so here is a trick:

* Store original value of the register in stack (using **PUSH**).
* Use the register for any purpose.
* Restore the original value of the register from stack (using **POP**).

Here is an example:

|  |
| --- |
| ORG 100h  MOV AX, 1234h  PUSH AX ; store value of AX in stack.  MOV AX, 5678h ; modify the AX value.  POP AX ; restore the original value of AX.  RET  END |

Another use of the stack is for exchanging the values,  
here is an example:

|  |
| --- |
| ORG 100h  MOV AX, 1212h ; store 1212h in AX.  MOV BX, 3434h ; store 3434h in BX  PUSH AX ; store value of AX in stack.  PUSH BX ; store value of BX in stack.  POP AX ; set AX to original value of BX.  POP BX ; set BX to original value of AX.  RET  END |

The exchange happens because stack uses **LIFO** (Last In First Out) algorithm, so when we push **1212h** and then **3434h**, on pop we will first get **3434h** and only after it **1212h**.

The stack memory area is set by **SS** (Stack Segment) register, and **SP** (Stack Pointer) register. Generally operating system sets values of these registers on program start.  
  
"**PUSH *source***" instruction does the following:

* Subtract **2** from **SP** register.
* Write the value of ***source*** to the address **SS:SP**.

"**POP *destination***" instruction does the following:

* Write the value at the address **SS:SP** to ***destination***.
* Add **2** to **SP** register.

The current address pointed by **SS:SP** is called **the top of the stack**.  
  
For **COM** files stack segment is generally the code segment, and stack pointer is set to value of **0FFFEh**. At the address **SS:0FFFEh** stored a return address for **RET** instruction that is executed in the end of the program.  
  
You can visually see the stack operation by clicking on [**Stack**] button on emulator window. The top of the stack is marked with "**<**" sign.

**Assignment Questions**

1. **Write a program to swap two numbers using stack?**
2. **Write a program to reverse a string?**
3. **Write a program input string from user and print it?**