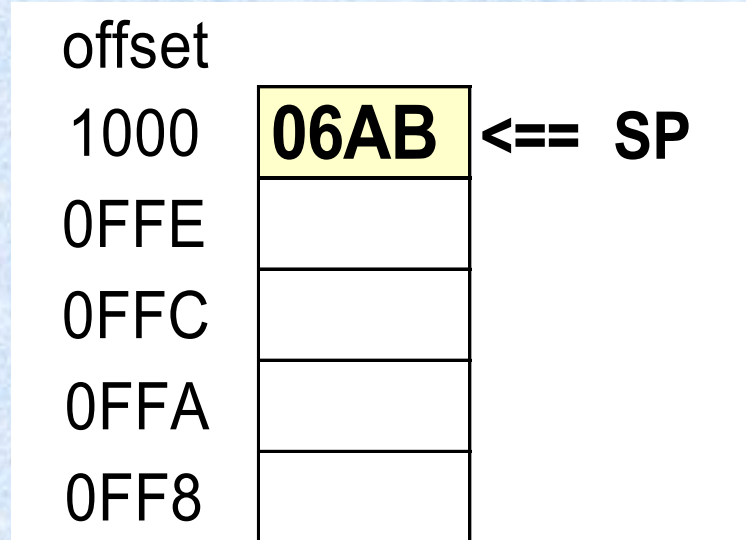


Stack Operations

- LIFO structure (last-in,first-out)
 - The last value put into the stack is the first value taken out
- Runtime stack
 - A memory array that is managed directly by **HARDWARE** in the CPU, using two registers: **SS** and **SP**.
 - Modified by instructions **CALL**, **RET**, **PUSH**, and **POP**

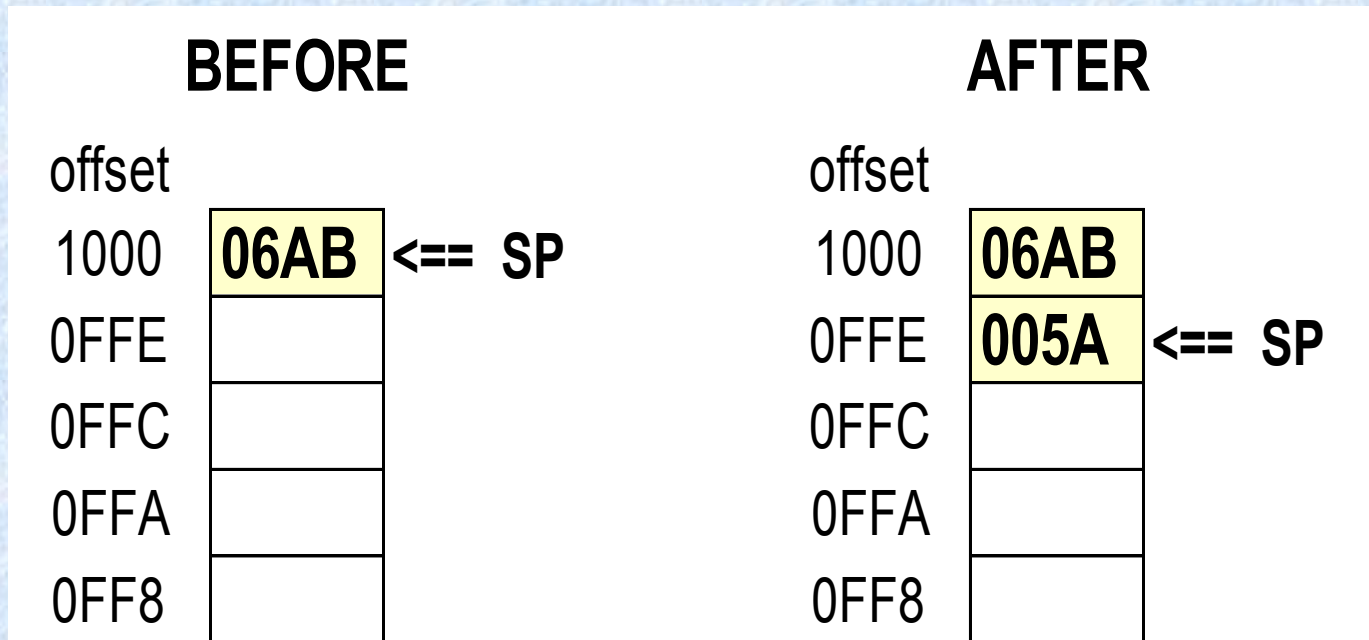
Stack Pointer Register (SP)

- Points to *LAST* integer to be added to (pushed onto) stack.



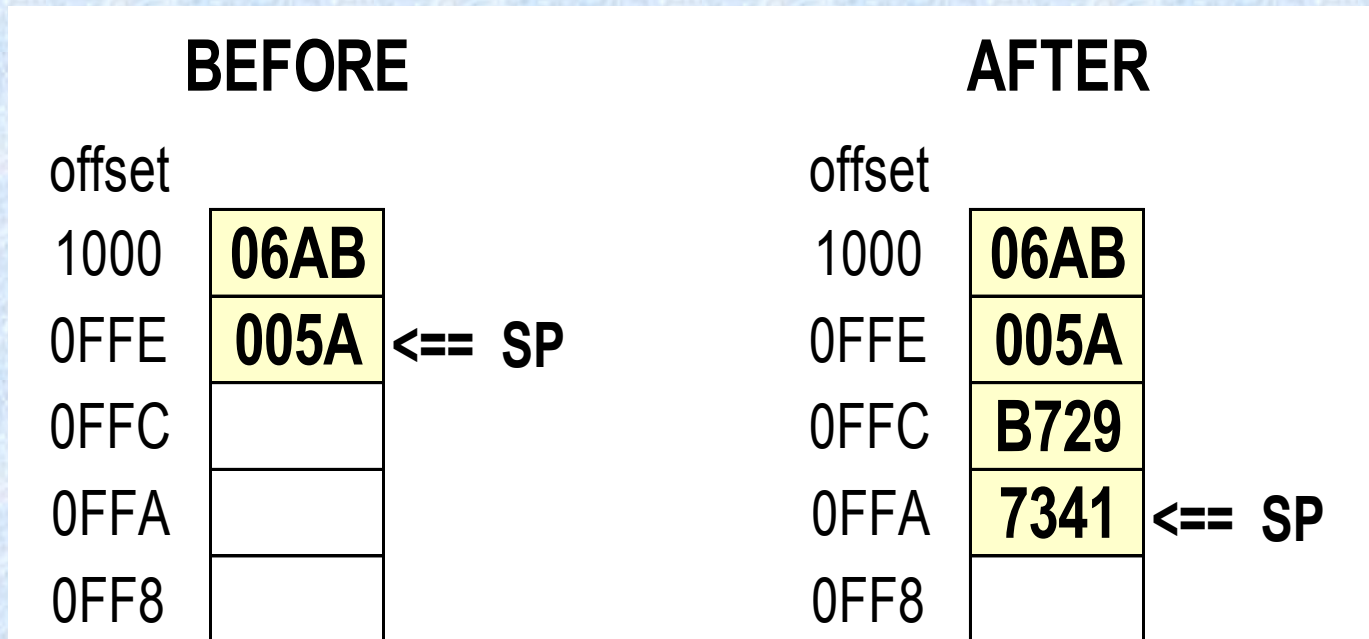
Push Operation

- PUSH 00A5



Push Operation(cont)

- PUSH 0B729h
- PUSH 7341h



Pop Operation

- POP DX
- After the operation, DX = 7431h

BEFORE		AFTER	
offset		offset	
1000	06AB	1000	06AB
0FFE	005A	0FFE	005A
0FFC	B729	0FFC	B729 <== SP
0FFA	7341 <== SP	0FFA	7341
0FF8		0FF8	

Stack Information

- PUSH and POP must be 16- or 32-bit values
 - No 8-bit register, memory operands
- After a POP, the data still resides in the stack but is overwritten on the next push instruction.
- The stack grows downward in memory.

Stack Applications

- A temporary save area for registers when they are used for more than one purpose.
- For CALL instructions, the returning CS:IP address is saved on the stack.
- Passing arguments to procedures are passed on the stack.
- Local variables inside a procedure are created on the stack.

Other PUSH/POP Instructions

- PUSHF
 - Pushes the values of the 16-bit flags register
- PUSHA
 - Pushes all 16-bit registers on the stack in the following order (AX,CX,DX,BX,SP,BP,SI,DI)
- POPF
- POPA
- PUSHFD/POPFD
- PUSHAD/POPAD

Write a program to reverse a string using the stack

- Use the \$ operator to determine the length of the string
- Use loops to move through the string
- Use Writestring to initially write the string correctly and again to write it backward. DX must point to the OFFSET of the string.
- Save the reversed string in the original string location.

RevString.asm

(data declaration)

```
Include Irvine16.inc
```

```
.data
```

```
String1 BYTE "This is a string",0
```

```
String_size = ($ - String1) - 1
```

Contents of Memory

- The \$ operator is the current location pointer. It points to the next available location in the data segment. If the first character is at location 0000, \$=0012h

$$\text{String_size} = 0012 - 0001 - 1 = 0010h$$

0001

0012



T	h	i	s		i	s		a		s	t	r	i	n	g		
54	68	69	79	20	69	79	20	61	20	79	80	78	69	75	67	0	

RevString.asm

(Data Segment Initialization)

```
.code
```

```
Main PROC
```

```
    Mov ax, @data
```

```
    Mov ds,ax
```

```
    Mov dx, OFFSET String1
```

```
    Call Writestring
```

RevString.asm

(code to push string on stack)

Mov cx, String_size

Mov si,0

Lp1: Mov al, String1[si]

Push ax

inc si

Loop Lp1

Contents of Stack Memory

- Push must be 16-bits (or 32-bits)
- Only wrote to AL – Don't know what is in AH.
- After all pushes, SP = 00DE (assuming original SP=0100)

00DE				00E7								00EF				
?	g	?	s	?	n	?	i	?	r	?	t	?	s	?	?	a
?	67	?	79	?	75	?	69	?	78	?	80	?	79	?	20	61
00F1		00F3		00F5		00F7		00F9		00FB		00FD		00FF		
?	?	s	?	i	?	?	s	?	i	?	h	?	T			
?	20	?	79	?	69	?	20	?	79	?	69	?	68	?	54	

RevString.asm

(code to pop string off stack)

```
    Mov cx, String_size
    Mov si,0
Lp2: Pop ax
    Mov String1[si], al
    inc si
    Loop Lp2
```

```
    Call Writestring
```

Contents of Memory

0001



0011



g	n	i	r	t	s		a		s	i		s	i	h	T	
67	75	69	78	80	79	20	61	20	79	20	20	79	69	68	54	00

RevString.asm

(Program Termination)

```
Mov ah,4Ch
```

```
Int 21h
```

```
Main endp
```

```
END Main
```