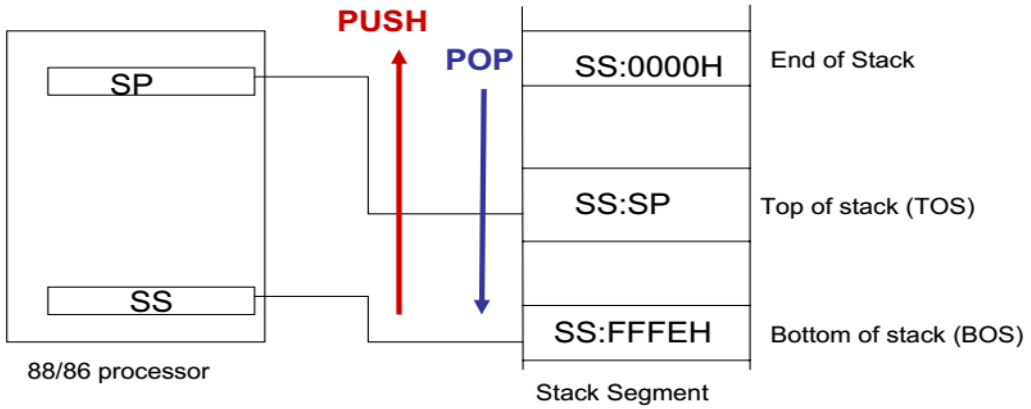
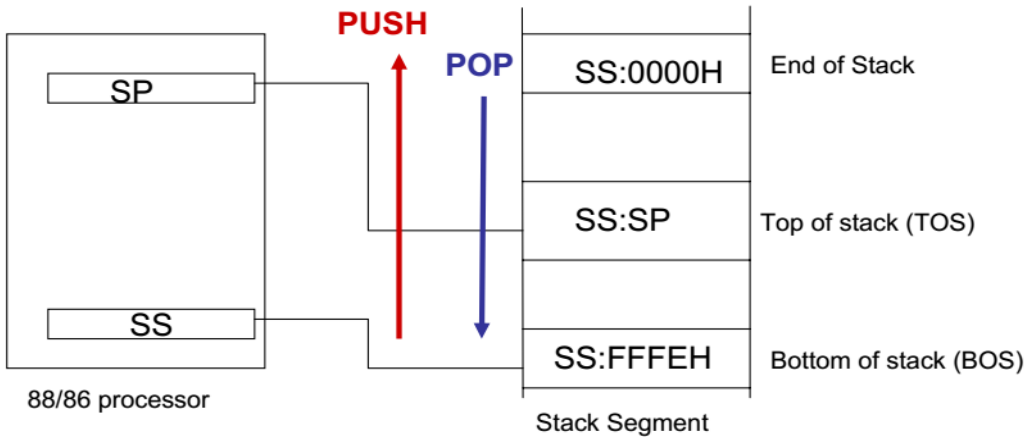
Lecture 10: Push-pop, Subroutine, Loop Instruction

- **The stack** is 64 KB used for temporary storage of information.
- Organized from software point of view as 32 KWs
- SP contains an offset value to the stack segment. Therefore, (SS:SP) is the physical address of last storage location in the stack to which data were pushed. It is referred to as Top Of Stack (TOS)
- SP is initiated to FFFEH which is referred to as The Bottom of Stack (BOS)
- Processor pushes one word at a time
- When a word is pushed, the SP is automatically decremented by 2 creating new location for two bytes and then the word is stored in this location.
- Stack grows from bottom of stack SS:FFFEH towards the end of stack SS:0000H



- **The stack** is 64 KB used for temporary storage of information.
- Organized from software point of view as 32 KWs
- SP contains an offset value to the stack segment. Therefore, (SS:SP) is the physical address of last storage location in the stack to which data were pushed. It is referred to as Top Of Stack (TOS)
- SP is initiated to FFFEH which is referred to as The Bottom of Stack (BOS)
- Processor pushes one word at a time
- When a word is pushed, the SP is automatically decremented by 2 creating new location for two bytes and then the word is stored in this location.
- Stack grows from bottom of stack SS:FFFEH towards the end of stack SS:0000H



Lecture 10: Push-pop, Subroutine, Loop Instruction

PUSH and POP Instructions

The stack is used as temporary storage of registers and memory locations
PUSH is used to store data and POP is used to retrieve it back from the stack

Mnemonic	Meaning	Format	Operation	Flags Affected
PUSH	Push word onto stack	PUSH S	((SP)) ← (S) (SP) ← (SP)-2	None
POP	Pop word off stack	POP D	(D) ← ((SP)) (SP) ← (SP)+2	None

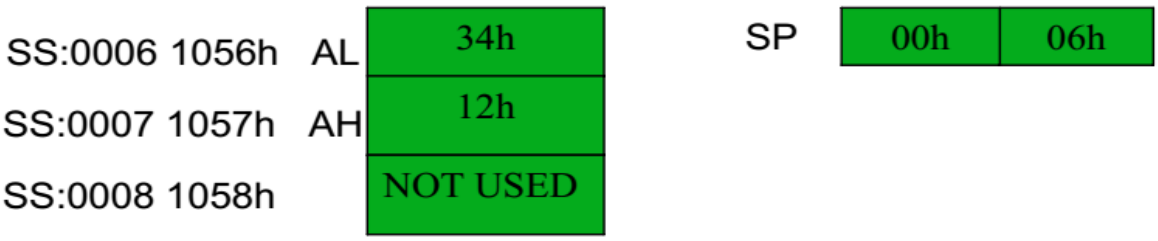
Operand (S or D)
Register
Seg-rcg (CS illegal)
Memory

Allowed operands for PUSH and POP instructions

Example for PUSH

Given
SS=0105h
SP=0008h
AX=1234h
What is the outcome of the instruction PUSH AX
BOS = 01050+FFFEh=1104h
TOS=01050+0008h=1058H

Decrement the SP by 2 and write AX into the word location 1056h.



Lecture 10: Push-pop, Subroutine, Loop Instruction

What is the outcome of the following instructions

POP AX
POP BX

If originally (SS:SP)=1056H

1056	34
1057	12
1058	BB
1059	AA

Read into the specified register from the stack and increment the stack pointer for each POP operation

At the first POP
(AX) = 1234h (SS:SP) = 1058h

At the second POP
(BX) = AABBh and (SS:SP) =105AH

Push and Pop Instructions

Saved parameters must be retrieved in the reverse order (First PUSHed, Last POPed)

To save registers and parameters on the stack

```
PUSH XX
PUSH YY
PUSH ZZ
```

Main body of the subroutine

```
.
.
.
.
.
```

To restore registers and parameters from the stack

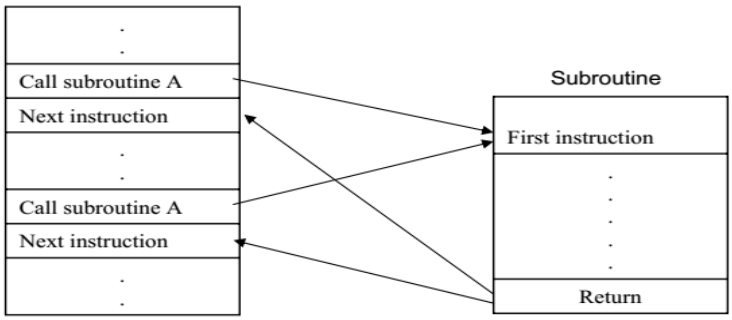
```
POP ZZ
POP YY
POP XX
RET
```

Return to main program

Lecture 10: Push-pop, Subroutine, Loop Instruction

Subroutines and Subroutine Handling Functions

• A subroutine is a special segment of a program that can be called for execution from any point in the program
•A RET instruction must be included at the end of the subroutine to initiate the return sequence to the main program environment
Examples:
CALL 1234h
CALL BX
CALL [BX]
CALL DWORD PTR [DI]



Mnemonic	Meaning	Format	Operation	Flags Affected
CALL	Subroutine call	CALL operand	Execution continuous from the address of the subroutine specified by the operand. Information required to return back to the main program such as IP and CS are saved on the stack.	None

Subroutine call instruction

Operand
Near-proc
Far-proc
Memptr16
Regptr16
Memptr32

Allowed operand

The operand of the call instruction initiates an intersegment of intrasegment c
The intrasegment call causes contents of IP to be saved on Stack.
The Operand specifies new value in the IP that is the first instruction in the Subroutine.
The Intersegment call causes contents of IP and CS to be saved in the stack and new values to be loaded in IP and CS that identifies the location of the First instruction of the subroutine.

Execution of RET instruction at the end of the subroutine causes the original Values of IP and CS to be POPed from stack.

Mnemonic	Meaning	Format	Operation	Flags Affected
RET	Return	RET	Return to the main program by restoring IP (and CS for far-proc).	None

Return instruction

Calling a NEAR proc

The CALL instruction and the subroutine it calls are in the same segment.

Save the current value of the IP on the stack.

Then load the subroutine's offset into IP.

Calling Program	Subroutine	Stack

```
Main proc                sub1  proc
001A: call sub1          0080: mov ax,1
001D: inc ax             …..
.                        ret
Main endp                sub1 endp
```

1ffd	1D
1ffe	00
1fff	(not used)

Calling a FAR proc

The CALL instruction and the subroutine it calls are in the "Different" segments.

Save the current value of the CS and IP on the stack.

Then load the subroutine's CS and offset into IP.

Calling Program	Subroutine	Stack

```
Main proc                    sub1  proc
1FCB:001A: call sub1  ---->  4EFA:0080: mov ax,1
1FCB:001D: inc ax     <----  …..
                             …
 ….                          ret
Main endp                    sub1 endp
```

1ffb	CB
1ffc	1F
1ffd	1D
1ffe	00
1fff	N/A

Nested Procedure Calls

A subroutine may itself call other subroutines.

Ex: in the following program identify the stack contents just before execution of The instruction RET in SUBC if at the beginning of main program (SS)=F1C0H and (SP) = 00FEH.

```
Example:
 Main proc
  000A call SUBA
  000C XLAT
     …..
 main endp

SUBA proc
0030    mov al, bl
          …
        call SUBB
0040    ret
SUBA endp
```

```
SUBB proc
0050    nop
          …
        call SUBC
0060    ret
SUBB endp

SUBC proc
          …
0079    nop
007A    ret
SUBC endp
```

F1CF8	xx	xx
F1CFA	60	00
F1CFC	40	00
F1CFE	0C	00

Stack Contents

Loop and Loop Handling Instructions

Mnemonic	Meaning	Format	Operation
LOOP	Loop	LOOP Short-label	(CX) ← (CX)-1 Jump is initiated to location definition by short-label if (CX)≠0; otherwise, execute next sequential instruction
LOOPE/LOOPZ	Loop while equal/loop while zero	LOOPE/LOOPZ short-label	(CX)←(CX)-1 Jump to location definition by short-label if (CX)≠0 and ZF=1; otherwise, execute next sequential instruction
LOOPNE/LOOPNZ	Loop while not equal/loop while not zero	LOOPNE/LOOPNZ short-label	(CX)← (CX)-1 Jump to location defined by short-label if (CX)≠0 and ZF=0; otherwise, execute next sequential instruction

Lecture 10: Push-pop, Subroutine, Loop Instruction

Loop

```
NEXT:    MOV CX,COUNT
                .
                .
                .
                .
         LOOP NEXT
```

Load count for the number of repeats

Body of routine that is repeated

Loop back to label NEXT if count not zero

```
EX:
         MOV AX,DATASEGADDR
         MOV DS,AX
         MOV SI,BLK1ADDR
         MOV DI,BLK2ADDR
         MOV CX,N
NXTPT:   MOV AH,[SI]
         INC SI
         INC DI
         LOOP NXTPT
         HLT
```

LOOP NXTPT = DEC CX, JNZ NXTPT

Nested Loops

```
MOV CX,A
BACK:…
….
…
…
LOOP BACK
```

Nested Loop

```
         MOV CX,A
OUTER:   PUSH CX
         MOV CX,99
INNER:   NOP
         …
         …
         …
         LOOP INNER
         POP CX
         LOOP OUTER
```