CS 271Computer Architecture and Assembly Language

Self-Check for Lecture#9

Solutions are posted

Here is a partial "listing file" for a MASM program:

```
00000000        main        PROC
00000000                    call  intro
00000005                    call  getData
0000000A
;  ... more implementation code for main
                            exit  ;exit to operating system
0000001B        main        ENDP

0000001B        intro       PROC
;  ... Implementation code for intro
0000003E  C3                ret   ;return to calling procedure
0000003F        intro       ENDP

0000003F        getData     PROC
;  ... more implementation code for getData

00000058                    call  validate
0000005D        ;     ... more implementation code for getData

00000067  C3                ret   ;return to calling procedure
00000068        getData     ENDP

00000068        validate    PROC
;  ... Implementation code for validate
0000008A  C3                ret   ;return to calling procedure
0000008B        validate    ENDP
```

Show the contents of the specified registers before and after the execution of each statement (OK to use 4-digit hex). The first row is completed for you.

Show the contents of the system stack after each instruction. Fill in the System Stack "Memory Address" column. When a "Memory Contents" value is replaced, lightly cross out the previous value (instead of erasing it). The shaded parts are completed for you.

Address / Instruction	EIP before	EIP after	ESP before	ESP after
0000 call intro	0000	001B	0400	03FC
003E ret				
0005 call getData				
0058 call validate				
008A ret				
0067 ret				

System Stack

Memory Address	Memory Contents
03FC	0005
0400	xxxx

Given the following data segment:

```
 .data
x     DWORD     17
y     DWORD     20
z     DWORD     13
```

Trace the following code fragments:

1.
```
push        x
push        y
pop         x
pop         y
```

x contains __________   y contains __________

2. Start over with original values in the data segment
```
push        x
inc         x
pop         y
push        x
inc         x
pop         z
```

x contains __________   y contains __________   z contains __________

3. Start over with original values in the data segment
```
        mov         eax,x
        push        eax
        mov         ecx,4
again:
        push        x
        push        y
        push        z
        pop         x
        pop         z
        pop         y
        loop        again

        pop         z
```

x contains __________   y contains __________   z contains __________