4.2.8

1. Write an instruction that increments val2.

**Answer:** inc val2

2. Write an instruction that subtracts val3 from EAX.

**Answer:** sub eax, val3

3. Write instructions that subtract val4 from val2.

**Answer:** mov ax, val4 sub val2, ax

4. If val2 is incremented by 1 using the ADD instruction, what will be the values of the Carry and Sign flags?

**Answer:** CF = 0, SF = 1

5. If val4 is incremented by 1 using the ADD instruction, what will be the values of the Overflow and Sign flags?

**Answer:** OF = 1, SF = 1

6. Where indicated, write down the values of the Carry, Sign, Zero, and Overflow flags after each instruction has executed:

mov ax,7FF0h

add al,10h ; a. CF = **1** SF = **0** ZF = **1** OF = **0**

add ah,1 ; b. CF = **0** SF = **1** ZF = **0** OF = **1**

add ax,2 ; c. CF = **0** SF = **1** ZF = **0** OF = **0**

4.3.8

1. (True/False): The OFFSET operator always returns a 16-bit value.

**Answer:** False

2. (True/False): The PTR operator returns the 32-bit address of a variable.

**Answer:** False

3. (True/False): The TYPE operator returns a value of 4 for doubleword operands.

**Answer:** True

4. (True/False): The LENGTHOF operator returns the number of bytes in an operand.

**Answer:** False

5. (True/False): The SIZEOF operator returns the number of bytes in an operand.

**Answer:** True

4.4.5

4. (True/False): The following is an indexed operand: array[esi]

**Answer:** True

5. Fill in the requested register values on the right side of the following instruction sequence:

mov esi,OFFSET myBytes

mov al,[esi] ; a. AL = **10h**

mov al,[esi+3] ; b. AL = **40h**

mov esi,OFFSET myWords + 2

mov ax,[esi] ; c. AX = **3Bh**

mov edi,8

mov edx,[myDoubles + edi] ; d. EDX = **3**

mov edx,myDoubles[edi] ; e. EDX = **3**

mov ebx,myPointer

mov eax,[ebx+4] ; f. EAX =**2**

4.10

ExitProcess proto

.data

source BYTE "This is a string", 0

target BYTE SIZEOF source DUP('#')

.code

main proc

;initialize rcx SIZEOF source

MOV rcx, SIZEOF source

;initialize pointer to first character of target

MOV rbx, 0

;start loop

MYLOOP:

;copy current character in source to current character in target

MOV rdx, QWORD PTR [source + rcx - 1]

;mov that character to current character in target

MOV QWORD PTR[target + rbx], rdx

;increment target pointer

INC rbx

;end loop

LOOP MYLOOP

call ExitProcess

main endp

end