Ankara University
Computer Engineering Department
COM325 Microprocessors
Lab 3

Q-1

- Write down the contents of registers when following instructions executed. Assume that contents of AX, BX, CX and DX are 0000H initially. Also address of data1 and data2 are 0000H and 0002H respectively.

```
data1 DW 1234H
data2 DW 5678H
mov si, 514ah
mov [si], 6583h
lodsb
lea di, data2
lea bx,[di]
mov cx,[di]
mov dx,[0000H]
```

AX	BX	CX	DX

A-1

- Write down the contents of registers when following instructions executed. Assume that contents of AX, BX, CX and DX are 0000H initially. Also address of data1 and data2 are 0000H and 0002H respectively.

```
data1 DW 1234H
data2 DW 5678H
mov si, 514ah
mov [si], 6583h
lodsb
lea di, data2
lea bx,[di]
mov cx,[di]
mov dx,[0000H]
```

AX	BX	CX	DX
0083H	0002H	5678H	1234H

Q-2

- Explain the difference between the SUB and CMP instruction.

A-2

- Explain the difference between the SUB and CMP instruction.
  - The comparison instruction (CMP) is a subtraction that changes **only** the flag bits; the destination operand **never** changes. However SUB instruction literally does substruction and the result is stored in destination. (Note that SUB can also change the flag bits.)

Q-3

- Where is the product found when two 8-bit numbers are multiplied?

A-3

- Where is the product found when two 8-bit numbers are multiplied?
  - AH (most significant) and AL (least significant)

Q-4

- What happens to the O and C flag bits when two numbers multiply?

A-4

- What happens to the O and C flag bits when two numbers multiply?
  - The O and C flags contain the state of the most significant portion of the product. If the most significant part of the product is zero, then C and O are zero.

Q-5

- Write a procedure named CUBE which includes sequence of instructions that cube the 8-bit number found in BL. Make sure that your result is a 16-bit number.

A-5

- Write a procedure named CUBE which includes sequence of instructions that cube the 8-bit number found in DL. Make sure that your result is a 16-bit number.

```
CUBE PROC
  MOV AL,DL
  MUL DL
  MUL DL
  RET
CUBE ENDP
```

Q-6

- In which register is the quotient (bölüm) and remainder (kalan) found, when performing 8-bit division?
- In which register is the quotient (bölüm) and remainder (kalan) found, when performing 16-bit division?

A-6

- In which register is the quotient (bölüm) and remainder (kalan) found, when performing 8-bit division?
  - AX is used for the dividend that is divided by the contents of any 8 bit register or memory location .
  - The **quotient** moves into **AL** with **AH** containing the whole number **remainder**

- In which register is the quotient (bölüm) and remainder (kalan) found, when performing 16-bit division?
  - The 16 bit number is divided into DX-AX, a 32 bit dividend .
  - The **quotient** appears in **AX** and the **remainder** in **DX**.

Q-7

- What errors are detected during a division?

A-7

- What errors are detected during a division?
  - Divide Overflow or Divide by Zero

```
; Divide Overflow
MOV AX,2FFFH
MOV BX,0002H

DIV BL

; Divide by Zero
MOV AX,2FFFH
MOV BX,0000H

DIV BL
```

Q-8

- Develop a short sequence of instructions that clears (0) the three leftmost bits of DH without changing the remainder of DH.

A-8

- Develop a short sequence of instructions that clears (0) the three leftmost bits of DH without changing the remainder of DH.
  - AND DH,1FH
  - DH: **** ****
  - 1FH: 0001 1111
  - Result: 000* ****

Q-9

- Develop a short sequence of instructions that sets (1) the rightmost 5 bits of BH without changing the remaining bits of BH.

A-9

- Develop a short sequence of instructions that sets (1) the rightmost 5 bits of BH without changing the remaining bits of BH.
    - OR BH, 1FH
    - DH: **** ****
    - 1FH: 0001 1111
    - Result: ***1 1111

Q-10

- Select the correct instruction to perform each of the following tasks:
  - Shift DI right three places, with zeros moved into the leftmost bit
  - Move all bits in AL left one place, making sure that a 0 moves into the rightmost bit position
  - Rotate all the bits of AL left three places

A-10

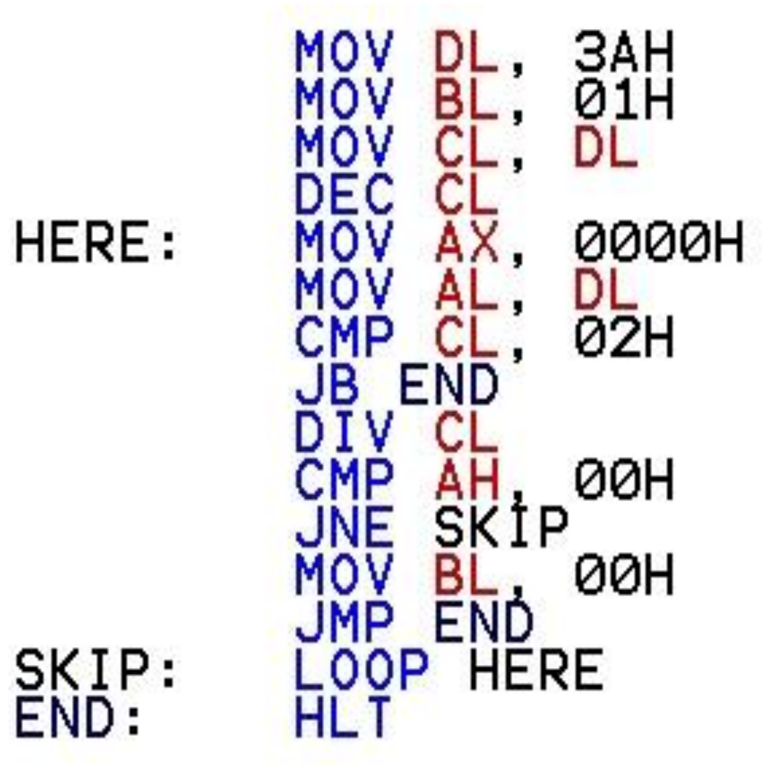
- Select the correct instruction to perform each of the following tasks:
  - Shift DI right three places, with zeros moved into the leftmost bit
    - SHR DI, 3
  - Move all bits in AL left one place, making sure that a 0 moves into the rightmost bit position
    - SHL AL, 1
  - Rotate all the bits of AL left three places
    - ROL AL, 3

Q-11

- Write an assembly code that finds whether a number is prime or not. If it is a prime number make BL = 1, If it is not make BL = 0. First MOV a number (e.g. : 59 (Decimal)) to DL and 1 to BL. Use JUMP and LOOP instructions.

A-11

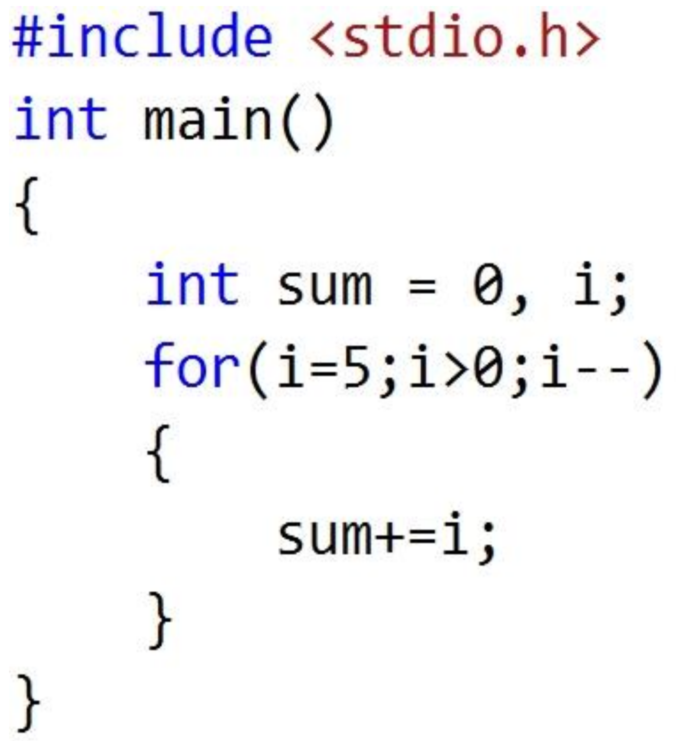
- Write an assembly code that finds whether a number is prime or not. If it is a prime number make BL = 1, If it is not make BL = 0. First MOV a number (e.g. : 59 (Decimal)) to DL and 1 to BL. Use JUMP and LOOP instructions.



```
         MOV DL, 3AH
         MOV BL, 01H
         MOV CL, DL
         DEC CL
HERE:    MOV AX, 0000H
         MOV AL, DL
         CMP CL, 02H
         JB END
         DIV CL
         CMP AH, 00H
         JNE SKIP
         MOV BL, 00H
         JMP END
SKIP:    LOOP HERE
END:     HLT
```

Q-12

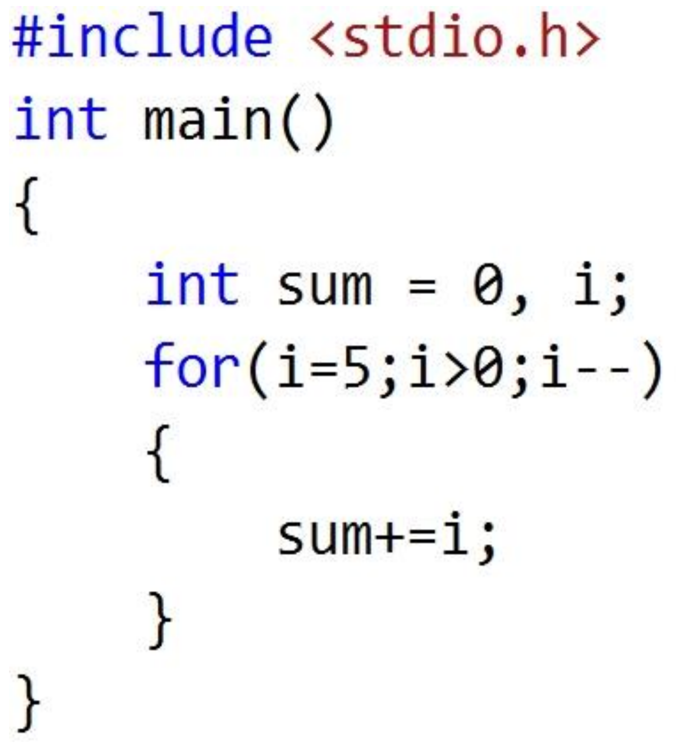
- Write the assembly language instructions for the following sequence(Use loop instructions)



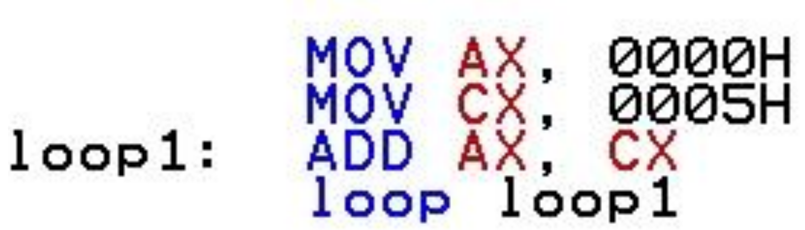
```c
#include <stdio.h>
int main()
{
    int sum = 0, i;
    for(i=5;i>0;i--)
    {
        sum+=i;
    }
}
```

A-12

- Write the assembly language instructions for the following sequence(Use loop instructions)



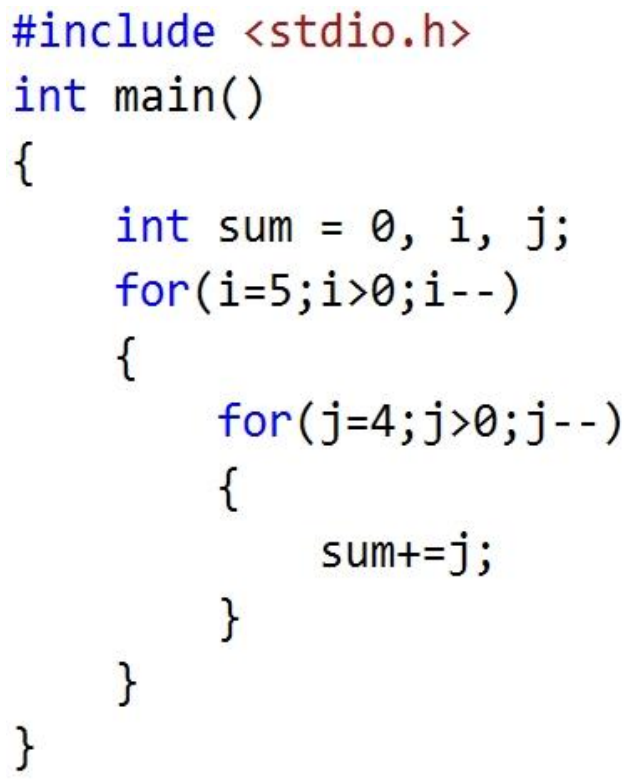
```
#include <stdio.h>
int main()
{
    int sum = 0, i;
    for(i=5;i>0;i--)
    {
        sum+=i;
    }
}
```



```
        MOV AX, 0000H
        MOV CX, 0005H
loop1:  ADD AX, CX
        loop loop1
```

Q-13

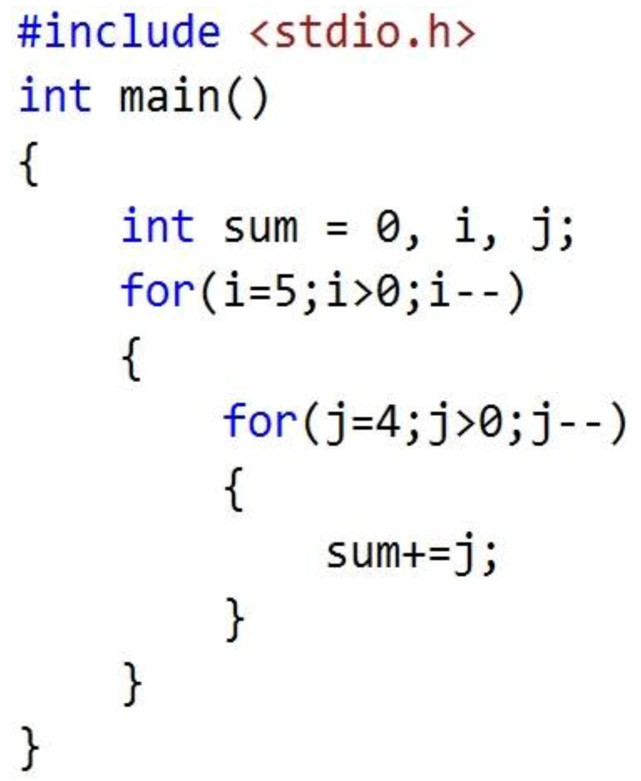
- Write the assembly language instructions for the following sequence(Use loop instructions)



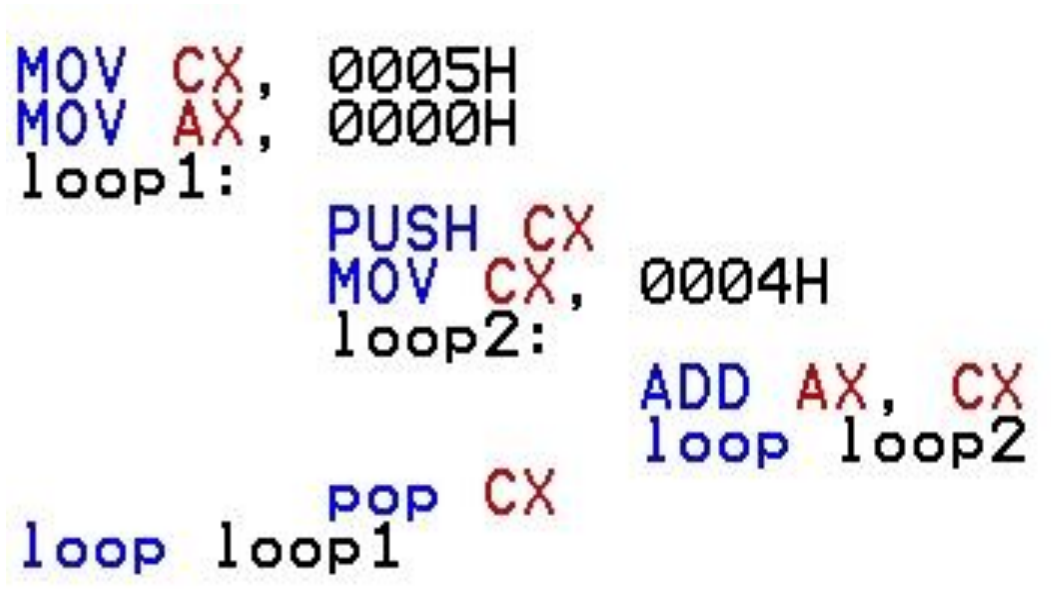
```
#include <stdio.h>
int main()
{
    int sum = 0, i, j;
    for(i=5;i>0;i--)
    {
        for(j=4;j>0;j--)
        {
            sum+=j;
        }
    }
}
```

A-13

- Write the assembly language instructions for the following sequence(Use loop instructions)



```c
#include <stdio.h>
int main()
{
    int sum = 0, i, j;
    for(i=5;i>0;i--)
    {
        for(j=4;j>0;j--)
        {
            sum+=j;
        }
    }
}
```



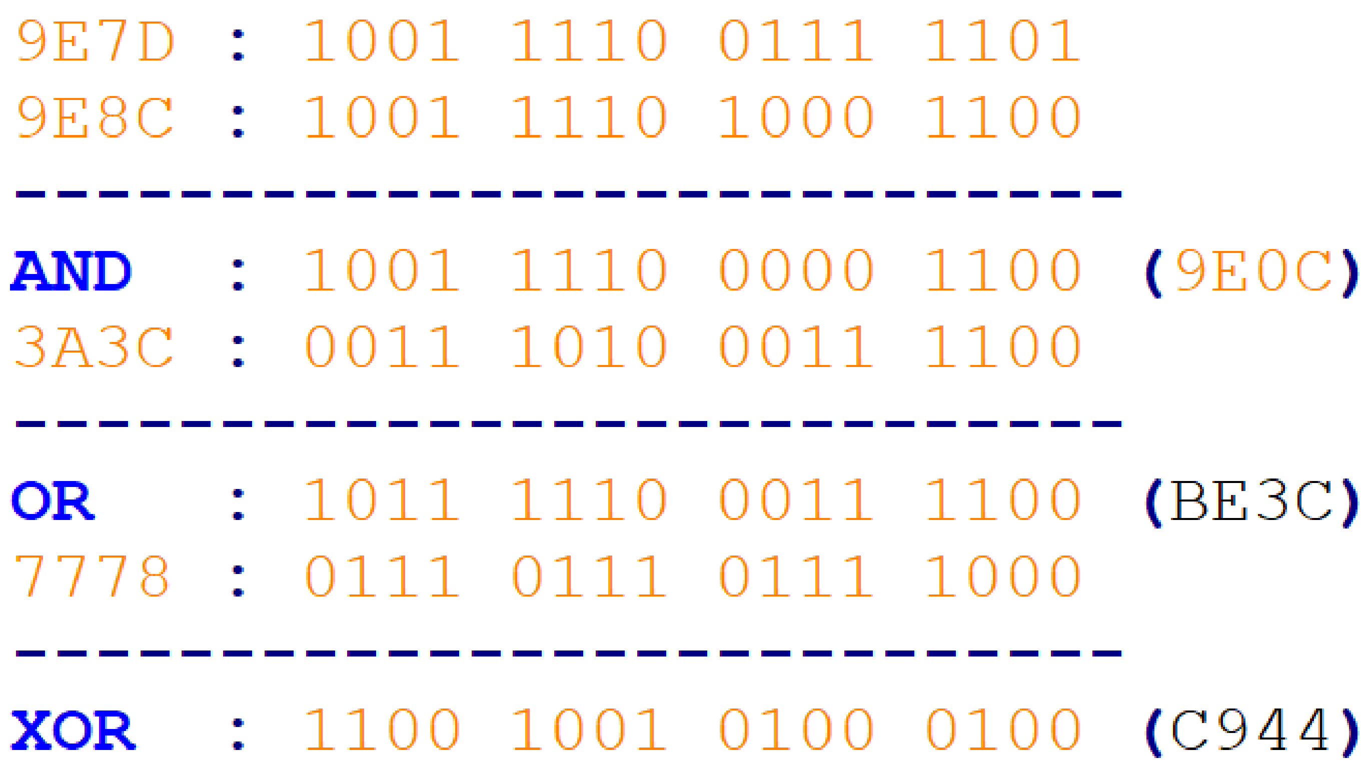
```
MOV CX, 0005H
MOV AX, 0000H
loop1:
          PUSH CX
          MOV CX, 0004H
          loop2:
                    ADD AX, CX
                    loop loop2
          pop CX
loop loop1
```

Q-14

- Assume AX= 9E7DH. AX is AND'ed with 9E8CH ,OR'ed with 3A3CH and XOR'ed 7778H. What is AX at last when these instructions applied respectively.

A-14

- Assume AX= 9E7DH. AX is AND'ed with 9E8CH ,OR'ed with 3A3CH and XOR'ed 7778H. What is AX at last when these instructions applied respectively.



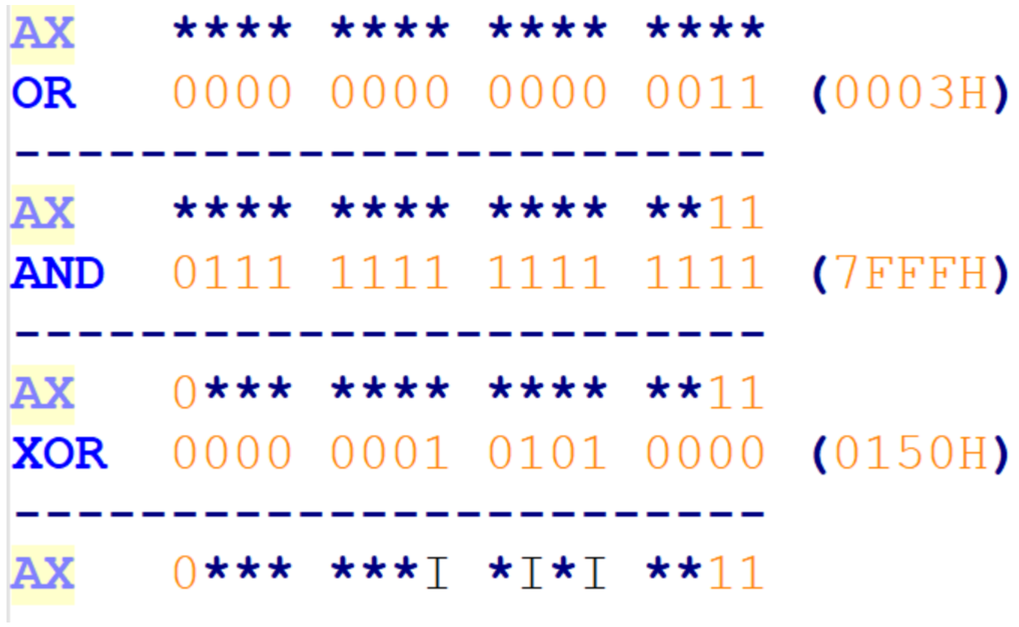
```
9E7D : 1001 1110 0111 1101
9E8C : 1001 1110 1000 1100
-----------------------------
AND  : 1001 1110 0000 1100 (9E0C)
3A3C : 0011 1010 0011 1100
-----------------------------
OR   : 1011 1110 0011 1100 (BE3C)
7778 : 0111 0111 0111 1000
-----------------------------
XOR  : 1100 1001 0100 0100 (C944)
```

Q-15

- Develop a sequence of instructions that sets (1) the rightmost 2 bits of AX; clears (0) the leftmost bit of AX; and inverts bits 4, 6 , and 8 of AX.

A-15

- Develop a sequence of instructions that sets (1) the rightmost 2 bits of AX; clears (0) the leftmost bit of AX; and inverts bits 4, 6 , and 8 of AX.
    - OR AX, 0003H
    - AND AX, 7FFFH
    - XOR AX, 0150H



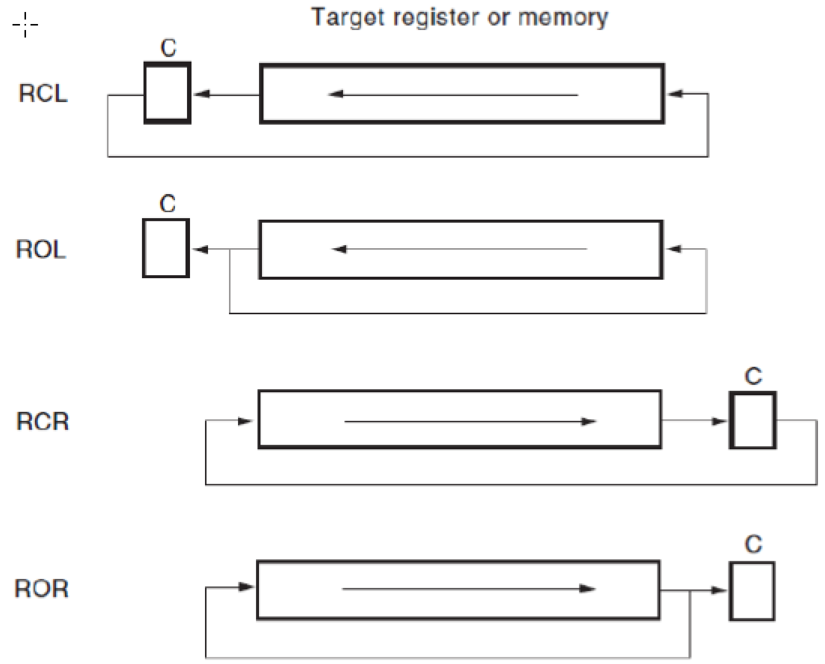
```
AX     **** **** **** ****
OR     0000 0000 0000 0011  (0003H)
-----------------------------
AX     **** **** **** **11
AND    0111 1111 1111 1111  (7FFFH)
-----------------------------
AX     0*** **** **** **11
XOR    0000 0001 0101 0000  (0150H)
-----------------------------
AX     0*** ***I *I*I **11
```

Q-16

- Select the correct instruction to perform each of the following tasks and calculate the data stored in AX at the end. Initially, CF is 0.
  - (AX=3007H) First rotate all the bits of AX left 3 places.
  - Then, shift AX right 4 places, with sign bit moved into the leftmost bit
  - Lastly, move all bits in AX left 2 places, making sure that 0's move into the rightmost bit position

A-16

- ROL AX, 3
- SAR AX, 4
- SHL AX, 2



```
AX     0011 0000 0000 0111 (3007H) CF=0
ROL    1000 0000 0011 1001 (8039H) CF=1
------------------------
AX     1000 0000 0011 1001
SAR    1111 1000 0000 0011 (F803H)
------------------------
AX     1111 1000 0000 0011
SHL    1110 0000 0000 1100 (E00CH)
```

Q-17

- Find how many odd and even numbers are in an array. Store the count for odd in BX, and for even in DX. Array can be seen in below

```
ARRAY DW 8, 11, 43, 56, 507, 608, 0, 123, 17, 13
count DW 10
```

A-17

```
org 100h

ARRAY DW 8, 11, 43, 56, 507, 608, 0, 123, 17, 13
count DW 10

cld; auto-increment
mov cx, count
mov si, offset ARRAY

mov bx, 0; count for odd numbers
mov dx, 0; count for even numbers

;main loop- through all elements
l0: lodsw ; load as word, since they are stored as words
    shr ax, 1; shift right the ax by 1 bit so that the least significant bit will be copied to cf
    jnc even: ; if carry is 1 then odd, otherwise even
    inc bx
    jmp cnt
even: inc dx
cnt:  ;continue
    loop l0
mov ax, 0; clear ax
hlt
```