23/03/2025 Aakanksha Mandal 110189000

Q 1) Fill in the blanks below? 10 Points

Q1.

	Before shift	After shift	
Instruction	AL / AX(last two lines)	AL /AX	CF
shl AL,1	1010 1110	0101 1100	1
shr AL,1	1010 1110	0101 0111	0
mov CL,3 shl AL,CL	0110 1101	0110 1000	1
mov CL,5 shr AX,CL	1011 1101 0101 1001	0000 0101 1110 1010	1
ror AX,CL	1011 1101 0101 1001	1100 1101 1110 1010	1

1) shl AL,1

Before: 1010 1110

After: 0101 1100

CF: 1

2) shr AL,1

Before: 1010 1110

After: 0101 0111

CF: 0

3) mov CL,3
shl AL,CL

Before: 0110 1101

Shift 1: 1101 1010 (CF=0)

Shift 2: 1011 0100 (CF=1)

Shift 3: 0110 1000 (CF=1)

CF: 1

4) mov CL,5
shr AX,CL

Before: 1011 1101 0101 1001

Shift 1: 0101 1110 1010 1100 (CF=1)

Shift 2: 0010 1111 0101 0110 (CF=0)

Shift 3: 0001 0111 1010 1011 CF=0)

Shift 4: 0000 1011 1101 0101 (CF=1)

Shift 5: 0000 0101 1110 1010 (CF=1)

CF: 1

5) ror AX,CL

Before: 1011 1101 0101 1001

Shift 1: 1101 1110 1010 1100 (CF=1)

Shift 2: 0110 1111 0101 0110 (CF=0)

Shift 3: 0011 0111 1010 1011 (CF=0)

Shift 4: 1001 1011 1101 0101 (CF=1)

Shift 5: 1100 1101 1110 1010 (CF=1)

CF: 1

Q2.

Q 2) Do you need to know the initial contents of the AX register in order to determine the contents of the AX register after executing the following code? If so, explain why. Otherwise, find the AX contents. (5 points)

a)

```
(a)
    mov     DX,AX
    not     AX
    or      AX,DX
```

We do not need to know the inital contents of the AX register in order to determine the contents of the AX register after executing the code. This is because:

- *mov DX, AX* copies the value of AX into DX. So now, DX contains the same values as AX.
- *not AX* inverts all the bits in AX. So, if a bit was 0, it becomes 1 and vice-versa.
- *or AX, DX* compares each bit of AX with the corresponding bit in DX. If either of the bits is 1, the result is 1. If both bits are 0, the result is 0. Since DX contains the orignal value of AX:
  - any bit that was originally 1 in AX (before the NOT operation) became 0 in AX after the NOT. But since the corresponding bit in DX was 1, the OR operation will be 0 OR 1 which is 1.
  - any bit that was originally 0 in AX (before the NOT operation) became 1 in AX after the NOT. Since the corresponding bit in DX was 0, the OR operation will be 1 OR 0 which is 1.

Therefore, after the OR operation, every bit in AX will be 1, regardless of the inital value of AX. So, all bits in AX will be 1, which is 1111 in binary and FFFF in hexadecimal.

b)

```
(b)
    mov     DX,AX
    not     AX
    and     AX,DX
```

We do not need to know the inital contents of the AX register in order to determine the contents of the AX register after executing the code. This is because:

- *mov DX, AX* copies the value of AX into DX. So now, DX contains the same values as AX.
- *not AX* inverts all the bits in AX. So, if a bit was 0, it becomes 1 and vice-versa.
- *and AX, DX* compares each bit of AX with the corresponding bit in DX. If both bits are 1, the result is 1. If either of the bits are 0, the result is 0. Since DX contains the orignal value of AX:
  - any bit that was originally 1 in AX (before the NOT operation) became 0 in AX after the NOT. But since the corresponding bit in DX was 1, the AND operation will be 0 AND 1 which is 0.
  - any bit that was originally 0 in AX (before the NOT operation) became 1 in AX after the NOT. Since the corresponding bit in DX was 0, the AND operation will be 1 AND 0 which is 0.

Therefore, after the AND operation, every bit in AX will be 0, regardless of the inital value of AX. So, all bits in AX will be 1, which is 0000 in both binary and hexadecimal.

Q3.

Q 3) In the following code fragment, state whether mov AX,10 or mov BX,1 is executed:(5 points)

```
(a)
    mov     CX,5
    sub     DX,DX
    cmp     DX,CX
    jge     jump1
    mov     BX,1
    jmp     skip1
jump1:
    mov     AX,10
skip1:
        . . .
```

mov BX,1 is executed and *mov AX,10* is not executed. This is because:

- *mov CX, 5* sets CX=5.
- *sub DX, DX* sets DX=0 because DX-DX=0.
- *cmp DX, CX* compares DX (0) with CX (5). DX-CX=0-5=-5 which means the less than condition is trueand the carry falg is set.
- *jge jump1* checks the sign flag and overflow flag to see if DX is greater than or equal to CX. Since the subtraction resulted in a negative value (-5), the condition DX>=CX is false, so the jump to jump1 is not performed.
- *mov BX, 1* is executed since the jump to jump1 is not performed. It sets BX=1.
- *jmp skip1* jumps to the skip1 label.
- *jump1: mov AX, 10* is skipped because the jump to jump1 was performed.
- *skip1:* the program continues from here.