Assignment 2

1. Write an Assembly Language Program to count the number of occurrence of `55H` in a string of eight data bytes. The starting address of string is `DS: 0030H`. Store the count value in `DS:0040H`.

```
.model small
.stack 100h
.code
main proc
    mov ax, @data
    mov DS, ax
    mov ES, ax
    mov di, 0030h
    mov si, 0040h
    mov al, 55h
    mov dl, 00h
    mov cx, 0008h
    cld
l2:
    scasb
    jnz l1
    inc dl
l1:
    loop l2
    mov [si], dl
    int 03h
main endp
end
```

Note

`scasb` compares the DI ptr value with AL. Updates CF and ZF

Operation: $Al - DI$

Sets ZF if both are equal. Sets CF if DI is greater
If cld is not set then increses DI else decreases DI

Warning

always ensure to clear the direction flag(`cld`) if you wish to go forward else it can go in any direction

2. Write an Assembly Language Program to find out the location where `55H` is placed in a string of eight data bytes. The starting address of string is `DS: 0030H`.

```
.model small
.stack 100h
.code
  main proc
  mov ax, @data
  mov DS, ax
  mov ES, ax
  mov di, 0030h
  mov si, 0040h

  mov al , 55h

  mov dl, 00h
  mov cx, 0008h

  cld;
  l2:
    scasb
    jnz l1
      dec di
      mov [si], di
      inc di
      add si, 0002h
  l1:
    loop l2
  mov [si], dl
  int 03h
  main endp
end
```

3. Write an Assembly Language Program to compare two strings. The first string is stored from memory location `DS: 0030H` and the second sting is stored from `DS: 0040H`. Consider that the first byte of both strings contain the number of bytes contained in that string. If both strings are found equal, then show a

value `FFFFH` in address `DS: 0050H`, otherwise show `1111H`.

```
.model small
.stack 100h
.code
  main proc
  mov ax, @data
  mov DS, ax
  mov ES, ax

  mov si, 0030h
  mov di, 0040h
  mov cl , [si]
  cmp [di], cl
  jnz failed
  inc si
  inc di
  mov ch, 00h

  cld;
  looping:
    cmpsb
    jz l1
  failed:
    mov si, 0050h
    mov bx,1111h
    mov [si], bx
    jmp exit
  l1:
    loop looping
    mov si, 0050h
    mov bx,0ffffh
    mov [si], bx
  exit:
  int 03h
  main endp
end
```

Note

`cmpsb` compares the DI ptr value with SI. Updates CF and ZF

Operation: $SI - DI$

Sets ZF if both are equal. Sets CF if DI is greater
If cld is not set then increases DI & SI else decreases DI & SI

4. Write an Assembly Language Program to check if a string of five data bytes is palindrome or not. The string is stored from memory location DS: 0030H. If the string is found to be palindrome then place FFFFH in addresses DS: 0040H otherwise place 1111H.

```
.model small
.stack 100h
.code
  main proc
  mov ax, @data
  mov DS, ax
  mov ES, ax

  mov si, 0030h
  mov ch, 00h
  mov cl , [si]
  inc si
  mov di, si
  add di, cx
  dec di
  mov ah, 00h
  mov al, cl
  mov bl , 02h
  div bl
  mov cl, al


  looping:
    mov bl, [di]
    cmp [si], bl
    jz l1
  failed:
    mov si, 0050h
    mov bx,1111h
    mov [si], bx
    jmp exit
  l1:
    inc si
    dec di
    loop looping
    mov si, 0050h
```

```
    mov bx,0ffffh
    mov [si], bx
  exit:
  int 03h
  main endp
end
```

5. Write an Assembly Language Program to count the number of positive and negative numbers present in a series of eight data bytes. The starting address of the series is DS: 0040H. Store the count value of positive number in DS: 0040H and count value of negative number in DS: 0041H.

```
.model small
.stack 100h
.code
  main proc
  mov ax,@data
  mov ds,ax
  mov cx,0008h
  mov si,0040h
  mov bl,00h
  mov dl,00h

  l1:
  mov al,[si]
  inc si
  sub al,00h
  jns l2
  dec bl
  inc dl ;negative

  l2:
  inc bl ;positive
  loop l1

  mov si,0040h
  mov [si],bl
  mov [si+1],dl


  int 03h
  main endp
end
```

Summary

if al is negative is then `sub al,00h` will generate a sign flag and so dl will be increased else bl will be increased

6. Write an Assembly Language Program to separate the odd and even numbers from a series of 7 data bytes. The starting address of the series is DS: 0030H. Store the even numbers from DS: 0040H and the odd numbers from DS: 0050H.

```
.model small
.stack 100h
.code
  main proc
    mov ax, @data
    mov ds, ax
    mov es, ax
    mov bx,0030h
    mov si,0040h
    mov di,0050h
    mov cx,0007h

    l4:
    mov al,[bx]
    ror al,01h
    jnc l1
    rol al,01h
    mov [di],al ; for odd
    inc di

    jmp l3

    l1:
    rol al,01h
    mov [si],al ; for even
    inc si

    l3:
    inc bx
    loop l4


  int 03h
```

```
  main endp
end
```

Summary

`ror` moves the last digit to the front

```
Initial AL:       1 1 0 1 0 0 1 0
                  | | | | | | | |
                  V V V V V V V V

Step 1: Rotate Right by 1:
                  0 1 1 0 1 0 0 1
                  | | | | | | | |
                  V V V V V V V V

Final AL:         0 1 1 0 1 0 0 1
```

7. Write an Assembly Language Program to convert an 8-bit number stored in DS: 0030H into its equivalent ASCII value. Store the converted code from DS: 0050H.

```
.model small
.stack 100h
.code
  main proc
    mov ax, @data
    mov ds, ax
    mov si,0030h
    mov al,[si]
    mov ah,al
    and al,0fh
    cmp al,0ah
    jc l1
    add al,07h

    l1:
    add al,30h
    mov bx,0050h
    mov [bx],al
    mov al,ah
```

```
    and al,0f0h
    mov cl,04h
    rol al,cl
    cmp al,0ah
    jc l2
    add al,07h

    l2:
    add al,30h
    inc bx
    mov [bx],al



  int 03h
  main endp
end
```

Summary

`and al,0fh` clears the upper 4 bits
Eg:

```
AL:    00111010   (value in `AL`)
0Fh:   00001111   (binary mask `0Fh`)
----------------
Result: 00001010   (low nibble is preserved, high nibble is cleared)
```

Summary

`cmp al, 0ah` This thing checks whether the 8 bit number is within 1-10 or not. If yes then no need to change anything and jump to l1 and move forward else add 07h which will give from `A-Z` then go to l1

`add al,30h` 30h translates to 48 which is the ascii code for 0

Then same is done for the upper bit

`rol al,cl` rotates the bits by 4. $i.e$ the upper 4 bits goes to the lower section

8. Write an Assembly Language Program to find out the square root of a number stored in DS: 0030H. Store the

result in DS: 0040H.

```
.model small
.stack 100h
.code
  main proc
    mov ax, @data
    mov ds, ax

    mov si,0030h
    mov al,[si]
    mov cl,99h
    mov dl,00h
    mov bl,01h

    l1:
    inc dl
    sub al,bl
    jz end1
    add bl,02h
    loop l1

  end1:
  mov si,0040h
  mov [si],dl


  int 03h
  main endp
end
```

Summary

So the idea of the square of the number depends on the number of steps of the below operation

```
Initial Number:    16
                   |
                   V

Step 1: Subtract 1:
                   15
                   |
                   V

Step 2: Subtract 3:
                   12
```

```
                    |
                    V

Step 3: Subtract 5:
                    7
                    |
                    V

Step 4: Subtract 7:
                    0
                    |
                    V

Final Answer:       4 steps (Square root of 16)
```

9. Fibonacci series is defined as:

$$F(i) = F(i-1) + F(i-2)$$

For all $i > 2$ with $F(1) = F(2) = 1$ Write an Assembly language Program to generate the first ten elements of this sequence and store them from DS: 0030H.

```
.model small
.stack 100h
.code
    main proc
        mov ax, @data
        mov ds, ax

        mov si, 0030h
        mov al,01h
        mov [si],al
        inc si
        mov [si],al
        mov cl,08h
        mov bl,00h

        l1:
        add bl,[si]
        dec si
        add bl,[si]
        add si,02h
        mov [si],bl
        mov bl,00h
```

```
        loop l1

        int 03h
    main endp
    end
```