ASM LAB ASSIGNMENT 2

Name: Hrithvik Kondalkar Roll: 002211001088

1. Write an assembly language program to display the first 10 Fibonacci numbers.

```
tostr macro number, strng1
    mov si, offset strng1
    mov bh, cl;storing outer loop cx
    mov cl, 10
    mov ch, 1
    mov bl, 0
    modnumb:
        xor ax,ax
        mov al, number
        div cl
        mov dl, ah
        xor ax,ax
        mov al, dl
        div ch
        cmp ah, dl
        je unloadstack
        add al, 48
        xor ah, ah
        push ax
        mov ch, cl
        mov al, cl
        mul ten
        mov cl, al
        inc bl
        jmp modnumb
    unloadstack:
        xor cx,cx
        mov cl, bl
        loop1:
            pop dx
            mov [si], dx
            inc si
        loop loop1
    xor cx,cx
    mov cl, bh
endm
next macro
        pop ax
        pop bx
        ;a has latest value b has previous value now
        mov dx, ax
        add dx, bx
        mov bx, ax
        mov ax, dx
        push bx
        push ax
        ;dx contains next value for fibronacci
        ;stack top also contains latest value at top
    endm
.model small
.stack 100h
.data
```

```
    ten db 10
    numofitr db 10
    currnum db 0
    strng db 10 dup("$")
.code
    main proc
        mov ax, @data
        mov ds, ax

        mov dx, 1
        push dx
        mov dx, 0
        push dx
        mov dx, 48
        mov ah, 02h
        int 21h
        mov dx, 32
        mov ah, 02h
        int 21h
        xor cx,cx
        mov cl, numofitr
        fibronacci:
            next
            mov currnum, dl
            tostr currnum, strng
            mov dx, offset strng
            mov ah, 09h
            int 21h
        loop fibronacci
        mov ah, 4ch
        int 21h
    main endp

    end main
```

2. Write an assembly language program to search the largest number in an array of ten 8-bit numbers. The array elements will be stored in the data segment.

```
tostr macro number, string
    mov si, offset string
    mov ax, number
    mov ch, 10
    mov cl ,0

    divloop:
        div ch
        add ah, 48
        cmp al, 0
        mov dl, ah
        mov dh, 0
        push dx
        je unstack
        inc cl
        mov ah, 0

    jmp divloop
    unstack:
        mov ch, 0
        add cl, 1
        poploop:
            pop dx
            mov [si], dx
            inc si
        loop poploop
endm

.model small
.stack 100h
.data
    num db 1,3,3,5,10,2,3,4,12,11
    str1 db 5 dup("$")
    msg1 db 'largest element : $'
.code
    main proc
        mov ax, @data
        mov ds, ax
        mov si, offset num

        mov cx, 9
        mov dx, 0
        mov dl, [si]
        push dx
        check:
            inc si
            pop ax
            mov dh, 0
            mov dl, [si]
            cmp dx, ax
            jg d
            push ax
            jmp skipd
            d:
                push dx
            skipd:
```

```
        loop check

        mov dx, offset msg1
        mov ah, 09h
        int 21h
        pop dx
        tostr dx, str1
        mov dx, offset str1
        mov ah, 9
        int 21h
        mov ah, 4ch
        int 21h
    main endp
end main
```

3. Write an assembly language program to sort in descending order using bubble sort algorithm a given set of byte sized unsigned numbers in memory.

```
swp macro si, ax
    mov ax, [si]
    mov byte ptr [si], ah
    inc si
    mov byte ptr [si], al
    dec si
endm
printarr macro arra
    push si
    mov si, offset arra
    push cx
    mov cx, numofelem
    mov ah, 02h
    mov dx, '['
    int 21h
    mov dx, ' '
    int 21h
    prloop:
        mov dh, 0
        mov dl, [si]
        add dl, '0'
        int 21h
        mov dx, ' '
        int 21h
        inc si
    loop prloop
    mov dx, ']'
    int 21h
    pop cx
    pop si
endm
.model small
.stack 100h
.data
    arr1 db 2,8,4,1,3,7
    lastindex dw 5
    numofelem dw 6
```

```
.code
    main proc
        mov ax, @data
        mov ds, ax


        mov cx, lastindex
        loop1:
            push cx
            mov ah, 02h
            mov dx, 10
            int 21h
            mov dx, 13
            int 21h
            mov si, offset arr1
            mov cx, lastindex
            printarr arr1
            loop2:
                mov dx, [si]
                cmp dl, dh
                jle finloop
                swap:
                    swp si, ax
                finloop:
                    inc si
            loop loop2
            pop cx
        loop loop1
        mov ah, 4ch
        int 21h
    main endp
end main
```

4. Write an assembly language program to search for a given 8-bits key using linear search in an array of 10 numbers. The search key will be asked to enter from the keyboard. A message should be displayed indicating whether the search was a success or a failure. If it is a successful case, the position of the number in the array is to be displayed.

```
.model small
.stack 100h
.data
    arr1 db 25,4,54,255,3,5,7,12,3,21
    msg1 db 'enter 8-bit key to search in array : $'
    msg2 db ' (overflow) key too large! $'
    msg3 db 'key not found in array$'
    msg4 db 'key found at index : $'
.code
    main proc
        mov ax, @data
        mov ds, ax

        mov dx, offset msg1
        mov ah, 09h
        int 21h

        mov cx, 10
        mov dx, 0
        uinput:
            mov ah, 01h
            int 21h
            cmp al, 13
            je linsrch
            sub al, '0'
            mov bl, al
            mov al, dl
            mov ah, 0
            mul cl
            jc overflw
            add al, bl
            ;cmp ah, 0
            jc overflw
            mov dl, al
        jmp uinput
        overflw:
            mov dx, offset msg2
            mov ah, 09h
            int 21h
            jmp trmin
        linsrch:
            mov di, 0
            mov cx, 10
            mov si, offset arr1
            srloop:
                cmp dl, [si]
                je found
                inc si
                inc di
            loop srloop
            mov dl, 13
            mov ah, 02h
            int 21h
            mov dl, 10
```

```
            int 21h
            mov dx, offset msg3
            mov ah, 09h
            int 21h
            jmp trmin
            found:
                mov dl, 13
                mov ah, 02h
                int 21h
                mov dl, 10
                int 21h
                mov dx, offset msg4
                mov ah, 09h
                int 21h
                mov dx, di
                add dl, '0'
                mov ah, 02h
                int 21h



        trmin:
            mov ah, 4ch
            int 21h
    main endp
end main
```

5. Write a program to check whether a 16-bit number is a palindrome or not. The number will be entered from the keyboard.

```
.model small
.stack 100h
.data
    uin dw 0
    msg1 db 'enter 16-bit number : $'
    msg2 db '[overflow] number too large'
    newl db 13,10,'$'
    msg4 db 'is not palindrome$'
    msg3 db 'is palindrome$'
.code
    main proc
        mov ax, @data
        mov ds, ax

        mov si, 0
        mov dx, offset msg1
        mov ah, 09h
        int 21h
        mov bx, 10
        uinput:
            mov ah, 01h
            int 21h
            cmp al, 13
            je palindr
            sub al, '0'
            mov dl, al
            mov dh, 0
            push dx
            mov ax, uin
            mul bx
            jc ovrflw
            pop cx
            add ax, cx
            jc ovrflw
            inc si
            mov uin, ax
        jmp uinput
    ovrflw:
    mov dx, offset newl
    mov ah, 09h
    int 21h
    mov dx, offset msg2
    int 21h
    jmp trmin

    palindr:
        mov ax, si
        mov bh, 0
        mov bl, 2
        div bx
        mov di, dx
        mov dx, 0
        mov ch,0
        mov cl, al
        mov si, cx
```

```
        mov ax, uin
        mov bx, 10
        cmp cx, 0
        je pali
        loop1:
            div bx
            push dx
            mov dx, 0
        loop loop1
        mov cx, si
        cmp di, 0
        je cmploop
        div bx
        mov dx, 0
        cmploop:
            div bx
            pop di
            cmp di, dx
            jne notpali
            mov dx, 0
        loop cmploop
        pali:
        mov dx, offset newl
        mov ah,09h
        int 21h
        mov dx, offset msg3
        int 21h
        jmp trmin

        notpali:
            mov dx, offset newl
            mov ah,09h
            int 21h
            mov dx, offset msg4
            int 21h


    trmin:
    mov ah, 4ch
    int 21h
    main endp
end main
```

6. Write a program to display the G.C.D. of two numbers M and N. Assume that the variables M and N are declared and initialized in the data segment.

```
.model small
.stack 100h
.data
    m db 48
    n db 60
    res db ?
    msg1 db 'G.C.D. = $'
.code
    printdb proc
        push ax
        push bx
        push cx
        push dx

        mov al, [si]
        mov ah, 0
        mov bl, 10
        mov bh, 0
        mov cx, 0
        rep1:
            cmp al, bl
            jl exrep
            div bl
            inc cx
            mov dl, ah
            mov dh, 0
            push dx
            jmp rep1
        exrep:
            inc cx
            mov dl, al
            mov dh, 0
            push ax
        printloop:
            pop dx
            add dl, '0'
            mov ah, 02h
            int 21h
        loop printloop
        mov dl, ' '
        mov ah, 02h
        int 21h
        pop dx
        pop cx
        pop bx
        pop ax
        ret
    printdb endp

    main proc
        mov ax, @data
        mov ds, ax

        mov ax, 0
        mov al, m
        mov bx, 0
```

```
        mov bl, n

        cmp al, bl
        jge loop1

        mov dl, al
        mov al, bl
        mov bl, dl

        loop1:
            div bl
            mov al, bl
            mov bl, ah
            cmp ah, 0
            mov ah, 0
            jne loop1

        mov res, al
        mov dx, offset msg1
        mov ah, 09h
        int 21h
        mov si, offset res
        call printdb
        mov ah, 4ch
        int 21h

    main endp
end main
```

7. Write an assembly language program to compare two strings. .model small

```
.model small
.stack 100h
.data
    str1 db "Hello$"
    str2 db "World$"
    msg1 db "Strings are equal.$"
    msg2 db "Strings are not equal.$"
.code
main proc
    mov ax, @data
    mov ds, ax

    lea si, str1
    lea di, str2

    compare_loop:
        mov al, [si]
        mov bl, [di]

        cmp al, bl
        jne strings_not_equal

        cmp al, '$'
        je strings_equal

        inc si
        inc di
        jmp compare_loop

    strings_equal:
        mov dx, offset msg1
        mov ah, 09h
        int 21h

        jmp exit_program

    strings_not_equal:
        mov dx, offset msg2
        mov ah, 09h
        int 21h

    exit_program:
        mov ah, 4ch
        int 21h

main endp
end main
```

8. Write a program to add two 32-bit numbers and store the result in consecutive memory locations.

```
.model small
.stack 100h
.data
    num1 dd 2147483647
    num2 dd 2147483647
    result dd ?
    msg1 db 'overflow$'
    msg2 db 'result in hex : $'
.code
    printhex proc
        push ax
        push bx
        push cx
        push dx

        mov dl, [si]

        mov bl, dl
        shr dl, 1
        shr dl, 1
        shr dl, 1
        shr dl, 1
        and bl, 0Fh

        add dl, '0'
        cmp dl, '9'
        jle prnt
        add dl, 7
        prnt:
        mov ah, 02h
        int 21h

        mov dl, bl
        add dl, '0'
        cmp dl, '9'
        jle prnt2
        add dl, 7
        prnt2:
        mov ah, 02h
        int 21h


        pop dx
        pop cx
        pop bx
        pop ax

        ret
    printhex endp
    main proc
        mov ax, @data
        mov ds, ax

        mov cx, num1
        mov si, offset num1
        mov dx, [si+2]
```

```
        mov ax, num2
        mov si, offset num2
        mov bx, [si+2]

        add dx, bx
        jc ovrflw
        add cx, ax
        jnc saveres
        add dx, 1
        jc ovrflw

        saveres:
            push dx
            mov dx, offset msg2
            mov ah, 09h
            int 21h
            pop dx
            mov si, offset result
            mov word ptr[si+2], dx
            add result, cx
            add si,3
            call printhex
            dec si
            call printhex
            dec si
            call printhex
            dec si
            call printhex
            mov dx, 'h'
            mov ah, 02h
            int 21h

            jmp trmin

        ovrflw:
            mov dx, offset msg1
            mov ah, 09h
            int 21h
        trmin:
        mov ah,4ch
        int 21h
    main endp
end main
```

9. Assume that two variables x and y are stored in packed BCD format. Write an 8086 alp to add x and y using DAA and display the result in packed BCD format also. Do the same addition without using DAA.

```
.model small
.stack 100h
.data
    x db 42h
    y db 27h
    bcdres dw ?
    addreslt dw ?
    a dw ?
    b dw ?
    base10 db 10
    newl db 13, 10, '$'
    msg1 db 'sum of x and y using DAA : $'
    msg2 db 'sum of x and y using normal addition : $'
.code
    hexprint proc ;8bits
        ;data to print is pointed by si
        push ax
        push bx
        push cx
        push dx

        mov bl, [si]
        mov bh, [si]

        shr bh, 1
        shr bh, 1
        shr bh, 1
        shr bh, 1
        and bl, 00001111b
        ;bh has higher 4 bits
        ;bl has lower 4 bits

        mov dx, 0
        prhigh:
            mov dl, bh
            add dl, '0'
            cmp dl, '9'
            jle exechigh
            add dl, 7
        exechigh:
            mov ah, 02h
            int 21h

        prlow:
            mov dl, bl
            add dl, '0'
            cmp dl, '9'
            jle execlow
            add dl, 7
        execlow:
            mov ah, 02h
            int 21h
        pop dx
        pop cx
        pop bx
```

```
    pop ax
    ret
hexprint endp
bcdtodec proc ;8bits
    ;si points to bcd format
    ;di points to converted decimal
    push ax
    push bx
    push cx
    push dx

    mov dx, 0
    mov dl, base10
    mov bh, 0
    mov ch, 0

    mov bl, [si]
    mov cl, [si]
    shr bl, 1
    shr bl, 1
    shr bl, 1
    shr bl, 1
    and cl, 00001111b
    ;bl has higher 4 bits
    ;cl has lower 4 bits

    mov ax, 0
    mov al, bl
    mul dl
    add ax, cx

    mov word ptr [di], ax
    pop dx
    pop cx
    pop bx
    pop ax
    ret
bcdtodec endp
printword proc
    ;si must point to word
    push ax
    push bx
    push cx
    push dx

    mov al, [si]
    mov ah, 0
    mov cx, 10
    mov bx, 0

    rep1:
        xor dx, dx
        div cx
        push dx
        inc bx
        test ax, ax
        jnz rep1
```

```
    printloop:
        pop dx
        add dl, '0'
        mov ah, 02h
        int 21h
        dec bx
        jnz printloop

    mov dl, ' '
    mov ah, 02h
    int 21h

    pop dx
    pop cx
    pop bx
    pop ax
    ret
printword endp

main proc
    mov ax, @data
    mov ds, ax

    mov ax, 0
    mov al, x
    add al, y
    daa
    mov bcdres, ax

    mov dx, offset msg1
    mov ah, 09h
    int 21h

    mov si, offset bcdres
    call hexprint

    mov si, offset x
    mov di, offset a
    call bcdtodec
    mov si, offset y
    mov di, offset b
    call bcdtodec

    mov ax, a
    add ax, b
    mov addreslt, ax

    mov dx, offset newl
    mov ah, 09h
    int 21h
    mov dx, offset msg2
    mov ah, 09h
    int 21h

    mov si, offset addreslt
    call printword

    mov ah, 4ch
```

```
        int 21h
    main endp
end main
```

10 Assume that two variables x and y are stored in packed BCD format. Write an 8086 alp to add x and y using DAA and display the result in packed BCD format also. Do the same addition without using DAA.

```
.model small
.stack 100h
.data
    file1 db "abc.txt",0
    file2 db "abc1.txt",0
    msg1 db "file doesn't exist$"
    msg2 db "rename successful$"
.code
main proc
    mov ax, @data
    mov ds, ax
    mov es, ax
    mov dx, offset file1
    mov di, offset file2

    mov ah, 56h
    int 21h

    jc nofile
    mov dx, offset msg2

    mov ah, 09
    int 21h
    jmp endk


    nofile:
    mov dx, offset msg1
    mov ah, 09
    int 21h

    endk:
        mov ah, 4ch
        int 21h
main endp
end main
```

11. Write a swap procedure that accepts the address of two words, and it exchanges the contents of those words. Write a program to initialize two variables and after the execution of the swap, the procedure displays the contents of the words. (Parameter passing needs to be done).

```
.model small
swap macro addrs1, addrs2
    push si
    push di
    push ax
    push bx

    mov si, addrs1
    mov di, addrs2

    mov ax, [si]
    mov bx, [di]
    mov word ptr [si], bx
    mov word ptr [di], ax

    pop bx
    pop ax
    pop di
    pop si
endm
.model small
.stack 100h
.data
    num1 dw 12
    num2 dw 15
    msg1 db 'numbers in order before swap : $'
    msg2 db 'numbers in order after swap : $'
    newl db 10 , 13 , '$'
.code
    printword proc
        ;si must point to word
        push ax
        push bx
        push cx
        push dx

        mov al, [si]
        mov ah, 0
        mov cx, 10
        mov bx, 0

        rep1:
            xor dx, dx
            div cx
            push dx
            inc bx
            test ax, ax
            jnz rep1

        printloop:
            pop dx
            add dl, '0'
            mov ah, 02h
            int 21h
            dec bx
```

```
            jnz printloop

        mov dl, ' '
        mov ah, 02h
        int 21h

        pop dx
        pop cx
        pop bx
        pop ax
        ret
    printword endp
    main proc
        mov ax, @data
        mov ds, ax

        mov cx, offset num1
        mov dx, offset msg1
        mov ah, 09h
        int 21h
        mov dx, offset num2

        mov si, cx
        call printword
        mov si, dx
        call printword

        swap cx, dx

        mov dx, offset newl
        mov ah, 09h
        int 21h
        mov dx, offset msg2
        mov ah, 09h
        int 21h

        mov si, offset num1
        call printword
        mov si, offset num2
        call printword

        mov ah, 4ch
        int 21h
    main endp
end main
```

12.Write an assembly language program to multiply two 3x3 matrices of signed 8-bit integers. Display result. Assume that each of the elements of the product matrix can be stored in an 8-bit location.

```
.model small
.stack 100h
.data
    matrix1 db 1, 2, 3, 4, 5, 6, 7, 8, 9  ; First 3x3 matrix
    matrix2 db 9, 8, 7, 6, 5, 4, 3, 2, 1 ; Second 3x3 matrix
    result db 0, 0, 0, 0, 0, 0, 0, 0, 0   ; Resultant matrix
    temp db ?
    dim db 3
    dimw dw 3
    newl db 10, 13,'$'
.code
printword proc
    push ax
    push bx
    push cx
    push dx

    mov al, [si]
    mov ah, 0
    mov cx, 10
    mov bx, 0

    rep1:
        xor dx, dx
        div cx
        push dx
        inc bx
        test ax, ax
        jnz rep1

    printloop:
        pop dx
        add dl, '0'
        mov ah, 02h
        int 21h
        dec bx
        jnz printloop

    mov dl, ' '
    mov ah, 02h
    int 21h

    pop dx
    pop cx
    pop bx
    pop ax
    ret
printword endp



matxprnt proc
    mov cx, dimw
    prloop1:
        mov dx, offset newl
```

```
        mov ah, 09h
        int 21h
        push cx
        mov cx, dimw
        prloop2:
            mov si, di
            call printword
            inc di
        loop prloop2
        pop cx
    loop prloop1

    ret
matxprnt endp

main proc
    mov ax, @data
    mov ds, ax

    mov cx, dimw

    mov bx, offset matrix2

    mov di, offset result

    rowsel:
        mov si, offset matrix1
        mov ax, dimw
        sub ax, cx
        mul dim
        add si, ax
        push cx
        mov ch,0
        mov cl, dim
        colsel:
            mov temp, 0
            mov bx, offset matrix2
            mov ax, dimw
            sub ax, cx
            add bx, ax
            push cx
            mov cx, dimw
            mov temp, 0
            push si
            calc:

                mov al, [si]
                mov dl, [bx]
                mov ah, 0
                mul dl
                add temp, al
                inc si
                add bx, dimw
            loop calc
            pop si
            mov al, temp
            mov [di], al
            inc di
```

```
            pop cx
        loop colsel
        pop cx
    loop rowsel

    mov di, offset result
    call matxprnt

    mov ah, 4Ch
    int 21h

main endp
end main
```

13 Write an assembly language program to get the screen width (no of cols) using BIOS interrupt and calculate the no. of rows from the appropriate word location in BIOS data area and clear the screen using BIOS interrupt.

```
.model small
.stack 100h
.data
    cols db ?
    rows db ?
    msg1 db 0dh,0ah,'Total no of rows =','$'
    msg2 db 0dh,0ah,'Total no of columns =','$'
    msg3 db 0dh,0ah,'Press any key to clear screen','$'
    newl db 10, 13, '$'
.code
    printdb proc
        push ax
        push bx
        push cx
        push dx

        mov al, [si]
        mov ah, 0
        mov bl, 10
        mov bh, 0
        mov cx, 0
        rep1:
            cmp al, bl
            jl exrep
            div bl
            inc cx
            mov dl, ah
            mov dh, 0
            push dx
            jmp rep1
        exrep:
            inc cx
            mov dl, al
            mov dh, 0
            push ax
        printloop:
            pop dx
            add dl, '0'
            mov ah, 02h
            int 21h
        loop printloop
```

```
    mov dl, ' '
    mov ah, 02h
    int 21h
    pop dx
    pop cx
    pop bx
    pop ax
    ret
printdb endp

main proc
    mov ax, @data
    mov ds, ax
    mov ah, 0fh
    int 10h
    mov cols, ah

    mov dx, offset msg2
    mov ah, 09h
    int 21h
    mov si, offset cols
    call printdb

    push ds
    mov ax, 0040h
    mov ds, ax
    mov si, 004ch
    mov ax, [si]
    pop ds

    mov cx, 0
    mov cl, cols
    shr al, 1
    div cl
    mov ah, 0
    mov cl, 2
    div cl
    mov rows, al

    mov dx, offset msg1
    mov ah, 09h
    int 21h
    mov si, offset rows
    call printdb

    mov dx, offset msg3
    mov ah, 09h
    int 21h
    mov ah, 01h
    int 21h
    mov cx, 0
    mov cl, rows
    mov dx, offset newl
    mov ah, 09h
    clrsc:
        int 21h
    loop clrsc
```

```
        mov ah, 4ch
        int 21h
    main endp
end main
```