# Assembly Lab Tasks

**1**

**Lab Task 1.asm**

.MODEL SMALL

.STACK 100H

.DATA

.CODE

MAIN PROC

MOV AX, @DATA

MOV DS, AX

MOV AH, 2

MOV CX, 256

MOV DL, 0

PRINT\_LOOP:

INT 21H

INC DL

DEC CX

JNZ PRINT\_LOOP

MOV AH, 4CH

INT 21H

MAIN ENDP

END MAIN

**Lab Task 2.asm**

.MODEL SMALL  
.STACK 100H  
.DATA  
 M DB "The sum is: $"  
.CODE  
  
MAIN PROC  
   
 MOV AX, @DATA  
 MOV DS, AX  
   
 MOV CX, 49   
 MOV AX, 1   
 XOR SI, SI   
  
SUM\_LOOP:   
  
 ADD SI, AX   
 ADD AX, 4   
 LOOP SUM\_LOOP   
   
 MOV DX, SI   
  
 MOV AH, 9   
 LEA DX, M   
 INT 21H   
  
 MOV AH, 4CH   
 INT 21H  
  
MAIN ENDP  
END MAIN

## Lab Task 3.asm

.MODEL SMALL  
.STACK 100H  
.DATA  
 M DB "The sum is: $"  
.CODE  
  
MAIN PROC  
   
 MOV AX, @DATA  
 MOV DS, AX  
   
 MOV AX, 100   
 MOV BX, 0   
 MOV CX, 19   
  
SUM\_LOOP:  
  
 ADD AX, BX   
 SUB AX, 5   
 LOOP SUM\_LOOP   
   
 MOV AH, 9   
 LEA DX, M   
 INT 21H  
   
 MOV AH, 4CH   
 INT 21H  
  
MAIN ENDP  
END MAIN

## Lab Task 4.asm

.MODEL SMALL  
.STACK 100H  
.DATA   
 M DB "ENTER A CHARACTER: $"   
 T DB "THANK YOU.$"   
.CODE  
  
MAIN PROC   
   
 MOV AX, @DATA  
 MOV DS, AX  
   
 MOV AH,9  
 LEA DX, M  
 INT 21H   
  
 MOV AH, 1   
 INT 21H   
 MOV BL, AL   
   
 MOV AH, 2  
 MOV DL, 10  
 INT 21H  
  
 MOV DL, 13  
 INT 21H  
   
 MOV CX, 50  
   
DISPLAY\_LOOP:   
   
 MOV AH, 2  
 MOV DL, BL   
 INT 21H   
 LOOP DISPLAY\_LOOP  
   
 MOV AH,9  
 LEA DX, T  
 INT 21H   
  
 MOV AH, 4CH   
 INT 21H  
  
MAIN ENDP  
END MAIN

**2**

**LAB TASK 1**

.model small

.stack 100h

.data

a db 10,13, 'Enter a Character: $'

b db 10,13, 'Character is a Vowel. $'

c db 10,13, 'Character is not a Vowel. $'

.code

main proc

mov ax, @data

mov ds, ax

mov ah, 9

lea dx, a

int 21h

mov ah, 1

int 21h

mov bl, al

cmp bl, 'a'

je is\_vowel

cmp bl, 'e'

je is\_vowel

cmp bl, 'i'

je is\_vowel

cmp bl, 'o'

je is\_vowel

cmp bl, 'u'

je is\_vowel

cmp bl, 'A'

je is\_vowel

cmp bl, 'E'

je is\_vowel

cmp bl, 'I'

je is\_vowel

cmp bl, 'O'

je is\_vowel

cmp bl, 'U'

je is\_vowel

lea dx, c

mov ah, 9

int 21h

jmp exit

is\_vowel:

lea dx, b

mov ah, 9

int 21h

exit:

mov ah, 4Ch

int 21h

main endp

end main

**2**

.model small

.stack 100h

.data

a db 10,13, 'Enter a character: $'

b db 10,13, 'Converted character: $'

.code

main proc

mov ax, @data

mov ds, ax

mov ah, 9

lea dx, a

int 21h

mov ah, 1

int 21h

mov bl, al

cmp bl, 'a'

jl not\_lowercase

cmp bl, 'z'

jg not\_lowercase

xor bl, 32

jmp convert\_done

not\_lowercase:

cmp bl, 'A'

jl not\_uppercase

cmp bl, 'Z'

jg not\_uppercase

xor bl, 32

jmp convert\_done

not\_uppercase:

jmp convert\_done

convert\_done:

mov ah, 9

lea dx, b

int 21h

mov ah, 2

mov dl, bl

int 21h

exit:

mov ah, 4ch

int 21h

main endp

end main

**3**

.model small

.stack 100h

.data

a db 10,13, 'Enter a Character: $'

b db 10,13, 'The Entered Character is not a Number. $'

c db 10,13, 'The Number is Even. $'

d db 10,13, 'The Number is Odd. $'

.code

main proc

mov ax, @data

mov ds, ax

mov ah, 9

lea dx, a

int 21h

mov ah, 1

int 21h

mov bl, al

cmp bl, '0'

jl not\_a\_number

cmp bl, '9'

jg not\_a\_number

mov al, bl

sub al, '0'

test al, 1

jz is\_even

lea dx, d

mov ah, 9

int 21h

jmp exit

is\_even:

lea dx, c

mov ah, 9

int 21h

jmp exit

not\_a\_number:

lea dx, b

mov ah, 9

int 21h

exit:

mov ah, 4ch

int 21h

main endp

end main

**3**

**1**

.model small

.stack 100h

.data

input\_msg db 'Enter a Binary Input:$'

output\_msg db 'Binary Output:$'

newline db 10, 13, '$'

.code

main proc

mov ax, @data

mov ds, ax

mov ah, 9

lea dx, input\_msg

int 21h

mov cx, 8

mov si, 0

mov bl, 0

input\_loop:

mov ah, 1

int 21h

cmp al, '0'

je zero\_bit

cmp al, '1'

je one\_bit

jmp invalid\_input

zero\_bit:

shl bl, 1

jmp next\_bit

one\_bit:

shl bl, 1

or bl, 1

next\_bit:

inc si

cmp si, 8

jne input\_loop

mov ah, 9

lea dx, newline

int 21h

mov ah, 9

lea dx, output\_msg

int 21h

mov cx, 8

mov si, 7

output\_loop:

mov dl, bl

and dl, 10000000b

cmp dl, 0

je print\_zero

mov dl, '1'

jmp print\_char

print\_zero:

mov dl, '0'

print\_char:

mov ah, 2

int 21h

shl bl, 1

loop output\_loop

mov ah, 4Ch

int 21h

invalid\_input:

mov ah, 9

lea dx, newline

int 21h

mov ah, 9

lea dx, newline

int 21h

main endp

end main

**2**

.model small

.stack 100h

.data

prompt db 'Enter a Hex Input: $'

error\_msg db 'Invalid Input! Please Enter a Valid Hex Number (0-9, A-F).', 0Dh, 0Ah, '$'

newline db 0Dh, 0Ah, '$'

.code

main:

mov ax, @data

mov ds, ax

mov ah, 09h

lea dx, prompt

int 21h

get\_hex\_input:

xor bx, bx

mov cx, 2

read\_loop:

mov ah, 01h

int 21h

mov ah, 0

cmp al, '0'

jl invalid\_input

cmp al, '9'

jg check\_upper\_case

sub al, '0'

jmp store\_digit

check\_upper\_case:

cmp al, 'A'

jl invalid\_input

cmp al, 'F'

jg invalid\_input

sub al, 'A' - 10

store\_digit:

shl bx, 4

or bx, ax

loop read\_loop

mov ah, 09h

lea dx, newline

int 21h

mov al, bl

shr al, 4

cmp al, 9

jbe display\_digit\_number

add al, 'A' - 10

jmp output\_char

display\_digit\_number:

add al, '0'

output\_char:

mov dl, al

mov ah, 02h

int 21h

mov al, bl

and al, 0Fh

cmp al, 9

jbe display\_digit\_number\_lower

add al, 'A' - 10

jmp output\_char\_lower

display\_digit\_number\_lower:

add al, '0'

output\_char\_lower:

mov dl, al

mov ah, 02h

int 21h

exit:

mov ah, 4Ch

int 21h

invalid\_input:

mov dx, offset error\_msg

mov ah, 09h

int 21h

jmp get\_hex\_input

**3**

.model small

.stack 100h

.data

prompt db 'Enter a Binary Number:$'

error\_msg db 'Invalid input! Please Enter A Valid Binary Number.', 0Dh, 0Ah, '$'

newline db 0Dh, 0Ah, '$'

.code

main:

mov ax, @data

mov ds, ax

mov ah, 09h

lea dx, prompt

int 21h

get\_binary\_input:

xor bx, bx

mov cx, 8

read\_loop:

mov ah, 01h

int 21h

cmp al, '0'

je valid\_input

cmp al, '1'

je valid\_input

mov dx, offset error\_msg

mov ah, 09h

int 21h

jmp get\_binary\_input

valid\_input:

sub al, '0'

shl bx, 1

or bx, ax

loop read\_loop

mov ah, 09h

lea dx, newline

int 21h

mov al, bl

shr al, 4

call display\_hex\_digit

mov al, bl

and al, 0Fh

call display\_hex\_digit

exit:

mov ah, 4ch

int 21h

display\_hex\_digit:

cmp al, 9

jbe display\_digit\_number

add al, 'A' - 10

jmp output\_char

display\_digit\_number:

add al, '0'

output\_char:

mov dl, al

mov ah, 02h

int 21h

ret

**4**

.model small

.stack 100h

.data

prompt db 'Enter a Hexadecimal Number:$'

error\_msg db 'Invalid Input! Please Enter a Valid Hexadecimal Number.', 0Dh, 0Ah, '$'

binary\_msg db 'Binary Number: $'

binary\_result db '0000$', 0

newline db 0Dh, 0Ah, '$'

.code

main:

mov ax, @data

mov ds, ax

get\_hex\_input:

mov ah, 09h

lea dx, prompt

int 21h

mov ah, 01h

int 21h

mov bl, al

cmp bl, '0'

jl invalid\_input

cmp bl, '9'

jle valid\_digit

cmp bl, 'A'

jl invalid\_input

cmp bl, 'F'

jg invalid\_input

valid\_digit:

cmp bl, '9'

jle convert\_digit

sub bl, 'A' - 10

jmp convert\_to\_binary

convert\_digit:

sub bl, '0'

convert\_to\_binary:

lea di, binary\_result

mov cl, 4

convert\_loop:

mov al, bl

and al, 08h

shr al, 3

add al, '0'

mov [di], al

inc di

shl bl, 1

dec cl

jnz convert\_loop

mov ah, 09h

lea dx, binary\_msg

int 21h

lea dx, binary\_result

int 21h

lea dx, newline

int 21h

exit:

mov ah, 4Ch

int 21h

invalid\_input:

mov ah, 09h

lea dx, error\_msg

int 21h

jmp get\_hex\_input

end main