1. **Apply Assembly Language Programming to enter and display 8-bit no's**

.model small

.stack

.data

msg1 db 10,13, "Enter the 8 bit number: $"

msg2 db 10,13, "Display the 8 bit number: $"

num db ?;

.code

.startup

mov ah,09; 09 use for dispaly msg on the output screen

lea dx,msg1;

int 21h;

mov ah,01; 01 use to enter single char or digit from user

int 21h;

sub al,30h;

mov cl,4;

shl al,cl; shl use to shift the digit from right to left

mov bh,al;

mov ah,01;

int 21h;

sub al,30h;

add bh,al;

mov num,bh;

mov ah,09;

lea dx,msg2;

int 21h;

mov bl,num;

and bl,0f0h;

mov cl,4;

shr bl,cl;

add bl,30h;

mov dl,bl;

mov ah,02;

int 21h;

mov bh,num;

and bh,0fh;

add bh,30h;

mov dl,bh;

mov ah,02;

int 21h;

.exit

end

Text

Description automatically generated

1. **Apply Assembly Language Programming to perform addition and subtraction of two 16 bits numbers using macros and procedure.**

.model small

.stack

.data

M1 DB 10, 13, "Addition $"

M2 DB 10, 13, "Subtraction $"

num1 DW 5678H;

num2 DW 3321H;

RES DW ?;

.code

Disp Macro XX;

MOV AH, 09;

LEA DX, XX;

INT 21H;

ENDM

.startup

Disp M1;

MOV AX, num1;

ADD AX, num2;

MOV RES, AX;

CALL DISP1

Disp M2;

MOV AX, num1;

SUB AX, num2;

MOV RES, AX;

CALL DISP1

JMP last;

DISP1 PROC

MOV BX, RES;

AND BH, 0F0H;

MOV CL, 4;

SHR BH, CL;

ADD BH, 30H;

MOV DL, BH;

MOV AH, 02;

INT 21H;

MOV BX, RES;

AND BH, 0FH;

ADD BH, 30H;

MOV DL, BH;

MOV AH, 02;

INT 21H;

AND BL, 0F0H;

MOV CL, 4;

SHR BL, CL;

ADD BL, 30H;

MOV DL, BL;

MOV AH, 02;

INT 21H;

MOV BX, RES;

AND BL, 0FH;

ADD BL, 30H;

MOV DL, BL;

MOV AH, 02;

INT 21H;

RET

Disp1 ENDP;

last:

.exit

end

Text

Description automatically generated

1. **Apply Assembly Language Programming to covert HEX to BCD.**

.model small

.stack

.data

;m2 db enter hexadecimal number : $"

m1 DB 10,13 , "Equivalent bcd no : $"

; num DW 0ffffH ; Or

num DW 0abcdh

.code

Disp Macro xx

mov ah,09

lea dx,xx

int 21h

endm

.startup

mov ax,num

mov dx,0

mov cl,0

mov bx,0ah

back:

mov dx,0

div bx

push dx

inc cl

cmp ax,0

jnz back

disp m1

back1:

pop dx

add dl,30h

mov ah,02

int 21h

dec cl

jnz back1

.exit

end

Graphical user interface, text

Description automatically generated

1. **Write an ALP To find negatives numbers in the given array.**

.MODEL SMALL

.STACK

.DATA

M1 DB 10,13,"NEGATIVE NUMBER:$"

M2 DB 10,13,"$"

ARRAY DB 50H,0AFH,25H,98H,0FEH

NARRAY DB 5 DUP(0)

.CODE

DISP MACRO XX

MOV AH,09

LEA DX,XX

INT 21H

ENDM

.STARTUP

MOV CH,0

LEA SI,ARRAY

LEA DI,NARRAY

MOV CL,5

BACK:

MOV DL,[SI]

AND DL,80H

JZ POSITIVE

MOV BL,[SI]

MOV [DI],BL

INC DI

INC CH

POSITIVE:

INC SI

DEC CL

JNZ BACK

DISP M1

LEA DI,NARRAY

BACK1:

DISP M2

MOV BH,[DI]

AND BH,0F0H

MOV CL,4

SHR BH,CL

CMP BH,9

JG AA

ADD BH,30H

JMP AA1

AA:

ADD BH,37H

AA1:

MOV DL,BH

MOV AH,02

INT 21H

MOV BH,[DI]

AND BH,0FH

CMP BH,9

JG AA2

ADD BH,30H

JMP AA3

AA2:

ADD BH,37H

AA3:

MOV DL,BH

MOV AH,02

INT 21H

INC DI

DEC CH

JNZ BACK1

.EXIT

END

Text

Description automatically generated

1. **Apply Assembly Language Programming to perform string operations. (i)Accept, (ii) Display, (iii) Concatenation (iv) Compare**

.MODEL SMALL

.STACK

.DATA

M1 DB,10,13," ENTER STRING 1:$"

M2 DB,10,13," ENTER LENGTH OF STR 1:$"

M3 DB,10,13," DISPLAY 1ST STRING:$"

M4 DB,10,13," ENTER STRING 2:$"

M5 DB,10,13," ENTER LENGTH OF STR 2:$"

M6 DB,10,13," DISPLAY 2ND STRING:$"

M7 DB,10,13," CONCATINATION OPERATION:$"

M8 DB,10,13," COMPARE OPERATION:$"

M9 DB,10,13," NOT EQUAL:$"

M10 DB,10,13," EQUAL:$"

STR1 DB 80,?,80 DUP(0)

STR2 DB 80,?,80 DUP(0)

LEN1 DB ?

LEN2 DB ?

.CODE

DISP MACRO XX

MOV AH,09

LEA DX,XX

INT 21H

ENDM

.STARTUP

DISP M1

MOV AH,0AH

LEA DX,STR1

INT 21H

DISP M2

LEA SI,STR1+1

MOV CL,[SI]

MOV LEN1,CL

ADD CL,30H

MOV DL,CL

MOV AH,02

INT 21H

DISP M3

MOV CL,LEN1

LEA SI,STR1+2

BACK:

MOV DL,[SI]

MOV AH,02

INT 21H

INC SI

DEC CL

JNZ BACK

DISP M4

MOV AH,0AH

LEA DX,STR2

INT 21H

DISP M5

LEA SI,STR2+1

MOV CL,[SI]

MOV LEN2,CL

ADD CL,30H

MOV DL,CL

MOV AH,02

INT 21H

DISP M6

MOV CL,LEN2

LEA SI,STR2+2

BACK1:

MOV DL,[SI]

MOV AH,02

INT 21H

INC SI

DEC CL

JNZ BACK1

DISP M8

LEA SI,STR1+2

LEA DI,STR2+2

MOV CL,LEN1

MOV CH,LEN2

CMP CL,CH

JNZ NE

BACK2:

MOV DL,[SI]

CMP DL,[DI]

JNZ NE

INC SI

INC DI

DEC CL

JNZ BACK2

DISP M10

JMP CONCATE

NE:DISP M9

CONCATE:

LEA SI,STR1+2

LEA DI,STR2+2

MOV CL,LEN1

MOV CH,LEN2

BACK3:

INC SI

DEC CL

JNZ BACK3

BACK4:

MOV DL,[DI]

MOV [SI],DL

INC SI

INC DI

DEC CH

JNZ BACK4

MOV CL,LEN1

MOV CH,LEN2

ADD CL,CH

LEA SI,STR1+2

BACK5:

MOV DL,[SI]

MOV AH,02

INT 21H

INC SI

DEC CL

JNZ BACK5

.EXIT

END

Text

Description automatically generated

1. **Apply Mixed Language Programming to design calculator**

**Code for Left Shift and Right Shift in C++**

#include <iostream.h>

#include <conio.h>

int main()

{

clrscr();

int a, b, c, d;

cout << "Enter Your Number : - " << endl;

cin >> a;

cout << "Enter No to shift: - " << endl;

cin >> b;

asm mov ax, a;

asm mov cx, b;

asm shl ax, cl;

asm mov c, ax;

cout << "Shift left :- " << c << endl;

cout << "Enter no to shift: - " << endl;

cin >> d;

asm mov ax, a;

asm mov cx, d;

asm shr ax, cl;

asm mov c, ax;

cout << "Shift Right :- " << c << endl;

getch();

return 0;

}

A picture containing text

Description automatically generated

**Code for Calculator in .C**

#include<stdio.h>

#include<conio.h>

void main() {

int a = 6, b = 3, c;

clrscr();

asm {

mov ax,a

mov bx,b

add ax,bx

mov c,ax

}

printf("%d",c);

getch();

}

1. **Develop program to interface mouse driver drivers.**

.model small

.stack

.data

M1 DB 10,13," Mouse driver present$"

.code

disp macro xx

mov ah,09h

lea dx,xx

int 21h

endm

.startup

mov ax,0000

int 33h

cmp ax,00

je last

disp M1

MOV AX,0004

MOV CX,0

MOV DX,0

int 33h

mov ax,0007

mov cx,0000

mov dx,00ffh

int 33h

mov ax,0008

mov cx,0000

mov dx,00ffh

int 33h

pixel:

mov ax,0001

int 33h

mov ax,0003

int 33h

cmp bx,01

je left

jmp right

left:

mov ax,0011h

int 10h

mov ah,0ch

int 10h

right:

cmp bx,02

je last

jmp pixel

last:

mov ax,00

int 10h

.exit

end

