SOFTWARE PROGRAMS:PART A

EXP NO: 1

Linear Search

Aim:
Design and develop an assembly language program to search a key element "X" in a list of 'n' 16-bit numbers. Adopt Linear search algorithm in your program for searching.

Algorithm

Step 1: Select the first element as the current element.

Step 2: Compare the current element with the target element. If matches, then go to step 5.

Step 3: If there is a next element, then set current element to next element and go to Step 2.

Step 4: Target element not found. Go to Step 6.

Step 5: Target element found and return location.

Step 6: Exit process.

```
.model small
print macro msg
lea dx,msg
mov ah,09h
int 21h
endm

data segment
array dw 1111h,2222h,3333h,3344h,4455h
len equ($-array)
key dw 2221h
msg1 db "key found $"
msg2 db "key not found $"
```

```
data ends

    code segment
    assume cs:code,ds:data
    start: mov ax,data
    mov ds,ax
    lea si,array
    mov cx,len
    mov bx,key
    next:mov ax,[si]
    cmp ax,bx
    je found
    inc si
    dec cx
    jnz next
    jne nf
    nf:print msg2
            jmp exit
    found:print msg1
    exit:mov ah,4ch
            int 21h
            code ends
            end start
```

Expected Output:

The message “the search element is available at the particular position” is displayed if “the search element is available” else it displays search element is “not available”

Result:

The program used the binary search algorithm to find a particular element from an array of elements and at a specific location.

Input	Output
Enter the number of elements in the array:5 Enter the array elements: **Search element:**	**Element available at location 2**

EXP No: 2

Bubble Sort

Aim:

Design and develop an assembly program to sort a given set of 'n' 16-bit numbers in ascending order. Adopt Bubble Sort algorithm to sort given elements.

Algorithm:

Step 1 : Declare the array with the numbers that need to be sorted.
Step 2 : Initialize iteration count (n-1)
Step 3 : Initialize comparison counter
Step 4 : Compare num1 and num2
Step 5 : Num1<=num2 do not exchange
Step 6 : Num1>=num2 then exchange the number positions
Step 7: Decrement iteration counter, comparison counter
Step 8 : Terminate the program

PROGRAM

```
.model small

data segment
a db 11h,33h,99h,22h,44h
len equ($-a)
data ends

  code segment
  assume cs:code,ds:data
        org 1000h
        start: mov ax,data
        mov ds,ax
        mov bx,len
        dec bx
        outloop: mov cx,bx
        lea si,a
        inloop: mov al,a[si]
        inc si
        cmp al,a[si]
                jnb nochange
                xchg al,a[si]
                mov a[si-1],al
                nochange: loop inloop
                dec bx
                jnz outloop
                mov ah,4ch
                int 21h
                code ends
                end start
```

Expected Output:

Trace the program after the debug command –t to get the location of SI then type the command d ds:00 to find the declared array use the debug command –g for executing the program go to the same location to find the sorted array

Result: The 8086 assembly program sorts the declared array using bubble sort algorithm in ascending order.

At source before execution	**11**	**33**	**99**	**22**	**44**

EXP No: 3

Palindrome

Aim:

Develop an assembly language program to reverse a given string and verify whether it is a palindrome or not. Display the appropriate message.

Algorithm:

Step 1 : Create display macro to display the message

Step 2 : Declare the string

Step 3 : Declare the message to display

Step 4 : Find the reverse of string and store in string1

Step 5 : Is string=string1,display it is a palindrome

Step 6 : Else if display not apalindrome

Step 7 : Terminate the program

Program

```
.model small
data segment
s db "madam"
l dw $-s
rs db 10 dup(?)
m1 db "palindrome $"
m2 db"not palindrome $"
data ends

code segment
assume cs:code,ds:data
start:mov ax,data
        mov ds,ax
        mov es,ax
        mov cx,l
        lea si,s
        lea di,rs
        add di,cx
        dec di
  b:mov al,[si]
        mov [di],al
        inc si
        dec di
        loop b
        lea si,s
        lea di,rs
        cld
```

```
        mov cx,l
        repe cmpsb
        jne np
        lea dx,m1
        mov ah,09h
        int 21h
        jmp d
np:lea dx,m2
        mov ah,09h
        int 21h
 d:mov ah,4ch
 int 21h
 code ends
 end start
```

Expected output: The string declared is checked with the original and reverse of the string and if the original and the reverse is equal then it is palindrome else it is not apalindrome.

Result: The entered string is reversed and compared with the original string to see if it is a palindrome or not, appropriate messages are displayed

INPUT	**OUTPUT**
MADAM	PALINDROME

HELLO	NOT A PALINDROME

EXP No: 4

Compute ncr using recursive procedure

Aim:

Develop an assembly language program to compute nCr using recursive procedure. Assume that ‘n’ and ‘r’ are non-negative integers.

Algorithm:

Step 1 :Initialize the values forn,r,res.
Step 2 :Call ncr procedure
Step 3 : If r=0,res=1 goto step
Step 4 :Else r=r-1
:Step 6 :Subtract n-r Multiply (n-r)*res
Step 7 :Res=(n-r)*res/2
Step 9 :Return to step 2 Save the result in res
Step 10 :Terminate the program

PROGRAM

```
.model small
data segment
s1 db "enter n:$"
s2 db 10,13,"enter r: $"
s3 db 10,13,"error n<r $"
s4 db 10,13, "ncr:$"
n db ?
r db ?
nn db ?
rr db ?
diff db ?          ;used to store n-r
data ends
code segment
assume cs:code,ds:data
start:
mov ax,data
mov ds,ax
lea dx,s1
mov ah,09h
int 21h
mov ah,1h        ;1h service routine for input from user
int 21h
sub al,30h
mov n,al
mov ch,0h
mov cl,n
mov ax,1h
call fact
mov nn,al
lea dx,s2
mov ah,09h
int 21h
mov ah,01h
int 21h
sub al,30h
mov r,al
mov ah,n
cmp ah,al
jb pe
mov ch,0h
mov cl,r
mov ax,01h
call fact
```

```
mov rr,al
mov ah,n
mov al,r
sub ah,al
mov diff,ah
mov ax,1h
mov ch,0h
mov cl,diff
call fact
mov cl,rr
mul cl
mov cl,al
mov al,nn
div cl
aam
mov bx,ax
lea dx,s4
mov ah,09h
int 21h
add bx,3030h
mov dl,bh
mov ah,02h
int 21h
mov dl,bl
mov ah,02h
int 21h
jmp exit
pe:
lea dx,s3
mov ah,09h
int 21h
exit:
mov ah,4ch
int 21h
fact proc
cmp cl,0h
je f0
f:
mul cl
loop f
ret
f0:
ret
fact endp
```

```
code ends
end start
```

Expected Result: The nCr is calculated using the recursive

procedure. N and R are non-negative integers.

Result: For the value N=4, R=2 the result is =6

EXP NO: 5

Read the Current Time and Date from the System and Display

Aim:

Design and develop an assembly language program to read the current time and Date from the system and display it in the standard format on the screen.

Program

```
.model small
.stack
.data
m1 db "current time:$"
m2 db 10,13,"current date:$"
```

```
hr db ?
min db ?
s db ?
day db ?
month db ?
year dw ?
.code
mov ax,@data
mov ds,ax
mov ah,2ch
int 21h
mov hr,ch
mov min,cl
mov s,dh
mov ah,2ah
int 21h
mov day,dl
mov month,dh
mov year,cx
lea dx,m1
mov ah,09
int 21h
mov cl,hr
mov ch,0
call disp
mov dl,':'
mov ah,2
int 21h
mov cl,min
mov ch,0
call disp
mov dl,':'
mov ah,02
int 21h
mov cl,s
```

```
mov ch,0
call disp
lea dx,m2
mov ah,09
int 21h
mov cl,day
mov ch,0
call disp
mov dl,'/'
mov ah,02
int 21h
mov cl,month
mov ch,0
call disp
mov dl,'/'
mov ah,02
int 21h
mov cx,year
call disp
mov ah,4ch
int 21h
disp proc
mov bx,0
n:mov al,bl
add al,1
daa
mov bl,al
jnc n1
add al,1
daa
mov bh,al
n1:loop n
mov dl,bl
and dl,0f0h
mov cl,4
```

```
shr dl,cl
add dl,30h
mov ah,02
int 21h
mov dl,bl
and dl,0fh
add dl,30h
int 21h
ret
disp endp
End
```

OUTPUT:

Date: 23/07/2022
Time:10:15:58

```
/*************************************************************************

 * Stepper motor Direction control

 * Developed by

 * Advanced Electronics Systems. Bengaluru.

 *-------------------------------------------------------------------

 * A stepper motor direction is controlled by shifting the voltage across
```

```
* the coils. Port lines : P0.12 to P0.15

*********************************************************************/



#include <LPC21xx.H>



void clock_wise(void);

void anti_clock_wise(void);



unsigned long int var1,var2;

unsigned int i=0,j=0,k=0;



int main(void)

{

        PINSEL0 = 0x00FFFFFF;               //P0.12 to P0.15 GPIo

        IO0DIR |= 0x0000F000;               //P0.12 to P0.15 output
```

```
        while(1)
        {
                for(j=0;j<50;j++)       // 20 times in Clock wise Rotation
                        clock_wise();

                for(k=0;k<65000;k++);   // Delay to show  anti_clock Rotation

                for(j=0;j<50;j++)       // 20 times in  Anti Clock wise Rotation
                        anti_clock_wise();

                for(k=0;k<65000;k++);   // Delay to show clock Rotation

        }                                                       // End of while(1)

}                                                               // End of main
```

```
void clock_wise(void)

{

        var1 = 0x00000800;                      //For Clockwise

        for(i=0;i<=3;i++)               // for A B C D Stepping

        {

                var1 = var1<<1;          //For Clockwise

        var2 = ~var1;

        var2 = var2 & 0x0000F000;



                IO0PIN = ~var2;



        for(k=0;k<3000;k++);          //for step speed variation

        }



}
```

```
void anti_clock_wise(void)

{

        var1 = 0x00010000;              //For Anticlockwise

        for(i=0;i<=3;i++)               // for A B C D Stepping

        {

        var1 = var1>>1;                 //For Anticlockwise

        var2 = ~var1;

        var2 = var2 & 0x0000F000;



                IO0PIN = ~var2;

        for(k=0;k<3000;k++);            //for step speed variation



        }

}
```

```
_____

_______

////// "LCD DISPLAY" To display the predefined data Date:21/01/2012 //////

#include<lpc214x.h>

#include<stdio.h>

//Function prototypes

void lcd_init(void);

void wr_cn(void);

void clr_disp(void);

void delay(unsigned int);

void lcd_com(void);

void wr_dn(void);

void lcd_data(void);
```

```
unsigned char temp1;

unsigned long int temp,r=0;

unsigned char *ptr,disp[] = "pda,",disp1[] = "cse";



int main()

{

        IO0DIR = 0x000000FC;                //configure o/p lines for lcd

        IO0PIN = 0X00000000;



        delay(3200);                        //delay

        lcd_init();                 //lcd intialisation

        delay(3200);                        //delay

        clr_disp();                                 //clear display

        delay(3200);                //delay
```

```
//.....LCD DISPLAY TEST.....//

        temp1 = 0x80;                           //Display starting address       of first line 1 th pos

        lcd_com();



        ptr = disp;

        while(*ptr!='\0')

        {

                temp1 = *ptr;

                lcd_data();

                ptr ++;

        }



        temp1 = 0xC0;                   // Display starting address of second line 4 th pos

        lcd_com();



        ptr = disp1;
```

```
        while(*ptr!='\0')
        {
                temp1 = *ptr;
                lcd_data();
                ptr ++;
        }
        while(1);
}  //end of main()


// lcd initialisation routine.
void lcd_init(void)
{
        temp = 0x30;
        wr_cn();
        delay(3200);
```

```
        temp = 0x30;

        wr_cn();

        delay(3200);



        temp = 0x30;

        wr_cn();

        delay(3200);



        temp = 0x20;   // change to 4 bit mode from default 8 bit mode

        wr_cn();

        delay(3200);



// load command for lcd function setting with lcd in 4 bit mode,

// 2 line and 5x7 matrix display



        temp = 0x28;
```

```
        lcd_com();

        delay(3200);



// load a command for display on, cursor on and blinking off

        temp1 = 0x0C;

        lcd_com();

        delay(800);



// command for cursor increment after data dump

        temp1 = 0x06;

        lcd_com();

        delay(800);



        temp1 = 0x80;  // set the cursor to beginning of line 1

        lcd_com();

        delay(800);
```

```
}


void lcd_com(void)
{
        temp = temp1 & 0xf0;
        wr_cn();
        temp = temp1 & 0x0f;
        temp = temp << 4;
        wr_cn();
        delay(500);
}


// command nibble o/p routine
void wr_cn(void)                //write command reg
{
        IO0CLR  = 0x000000FC;           // clear the port lines.
```

```
        IO0SET          = temp;                                 // Assign the value to the PORT lines

        IO0CLR  = 0x00000004;               // clear bit  RS = 0

        IO0SET          = 0x00000008;           // E=1

        delay(10);

        IO0CLR  = 0x00000008;

}



// data nibble o/p routine

void wr_dn(void)                        ////write data reg

{

        IO0CLR = 0x000000FC;        // clear the port lines.

        IO0SET = temp;                          // Assign the value to the PORT lines

        IO0SET = 0x00000004;        // set bit  RS = 1

        IO0SET = 0x00000008;        // E=1

        delay(10);

        IO0CLR = 0x00000008;
```

```
}


// data o/p routine which also outputs high nibble first
// and lower nibble next
void lcd_data(void)
{
        temp = temp1 & 0xf0;
        temp = temp ;//<< 6;
        wr_dn();
        temp= temp1 & 0x0f;
        temp= temp << 4;
        wr_dn();
        delay(100);
}


void clr_disp(void)
```

```
{
// command to clear lcd display
        temp1 = 0x01;
        lcd_com();
        delay(500);
}

void delay(unsigned int r1)
{
        for(r=0;r<r1;r++);
}
```

```
/*Program to demonstrate keyboard operation  Date:11/11/2011

Takes a key from key board and displays it on LCD screen*/



#include<lpc21xx.h>

#include<stdio.h>



/******* FUNCTION PROTOTYPE*******/



void lcd_init(void);

void clr_disp(void);

void lcd_com(void);

void lcd_data(void);

void wr_cn(void);

void wr_dn(void);

void scan(void);

void get_key(void);
```

```
void display(void);

void delay(unsigned int);

void init_port(void);



unsigned long int scan_code[16]= {0x00EE0000,0x00ED0000,0x00EB0000,0x00E70000,

                                  0x00DE0000,0x00DD0000,0x00DB0000,0x00D70000,

                                  0x00BE0000,0x00BD0000,0x00BB0000,0x00B70000,

                                  0x007E0000,0x007D0000,0x007B0000,0x00770000};




unsigned char ASCII_CODE[16]= {'0','1','2','3',

                               '4','5','6','7',

                               '8','9','A','B',

                              'C','D','E','F'};
```

```
unsigned char  row,col;


unsigned char temp,flag,i,result,temp1;

unsigned int r,r1;

unsigned long int var,var1,var2,res1,temp2,temp3,temp4;

unsigned char *ptr,disp[] = "4X4 KEYPAD";

unsigned char disp0[] = "KEYPAD TESTING";

unsigned char disp1[] = "KEY = ";

int main()

{

        //   __ARMLIB_enableIRQ();

        init_port();                //port intialisation

        delay(3200);                        //delay

        lcd_init();                 //lcd intialisation

        delay(3200);                        //delay

        clr_disp();                                 //clear display
```

```
delay(500);                 //delay



//.....LCD DISPLAY TEST.....//

ptr = disp;

temp1 = 0x81;                       // Display starting address

lcd_com();

delay(800);



while(*ptr!='\0')

{

          temp1 = *ptr;

lcd_data();

   ptr ++;

}



//.....KEYPAD Working.....//
```

```
        while(1)
        {
                get_key();
        display();
        }


}  //end of main()


void get_key(void)              //get the key from the keyboard
{
        unsigned int  i;
        flag = 0x00;
        IO1PIN=0x000f0000;
        while(1)
        {
                for(row=0X00;row<0X04;row++)     //Writing one for col's
```

```
{
        if( row == 0X00)
{
        temp3=0x00700000;
                }
else if(row == 0X01)
{
        temp3=0x00B00000;
                }
                else if(row == 0X02)
                {
        temp3=0x00D00000;
                }
else if(row == 0X03)
                {
        temp3=0x00E00000;
```

```
                }

                var1 = temp3;

  IO1PIN = var1;              // each time var1 value is put to port1

  IO1CLR =~var1;              // Once again Conforming (clearing all other bits)

  scan();

  delay(100);                 //delay

  if(flag == 0xff)

  break;

        } // end of for

        if(flag == 0xff)

        break;

}  // end of while



for(i=0;i<16;i++)

{

        if(scan_code[i] == res1)        //equate the scan_code with res1
```

```
                {
                        result =  ASCII_CODE[i];      //same position value of ascii code
                        break;                        //is assigned to result
                }
        }
}// end of get_key();


void scan(void)
{
        unsigned long int t;
        temp2 = IO1PIN;                         // status of port1
        temp2 = temp2 & 0x000F0000;                     // Verifying column key
        if(temp2 != 0x000F0000)                 // Check for Key Press or Not
        {
                delay(1000);                      //delay(100)//give debounce delay check again
                temp2 = IO1PIN;
```

```
            temp2 = temp2 & 0x000F0000;             //changed condition is same


        if(temp2 != 0x000F0000)                 // store the value in res1
        {
            flag = 0xff;
          res1 = temp2;
          t = (temp3 & 0x00F00000);          //Verfying Row Write
          res1 = res1 | t;                      //final scan value is stored in res1
        }
        else
        {
            flag = 0x00;
        }
        }
} // end of scan()
```

```
void display(void)

{

        ptr = disp0;

        temp1 = 0x80;                   // Display starting address of first line

        lcd_com();



        while(*ptr!='\0')

        {

                temp1 = *ptr;

                lcd_data();

                ptr ++;

        }



        ptr = disp1;

        temp1 = 0xC0;                   // Display starting address of second line

        lcd_com();
```

```
        while(*ptr!='\0')
        {
                temp1 = *ptr;
        lcd_data();
                ptr ++;
        }
        temp1 = 0xC6;                           //display  address for key value
        lcd_com();
        temp1 = result;
        lcd_data();
}

void lcd_init (void)
{
        temp = 0x30;
```

```
        wr_cn();

        delay(3200);



        temp = 0x30;

        wr_cn();

        delay(3200);



        temp = 0x30;

        wr_cn();

        delay(3200);



        temp = 0x20;

        wr_cn();

        delay(3200);



// load command for lcd function setting with lcd in 4 bit mode,
```

```
// 2 line and 5x7 matrix display


        temp = 0x28;

        lcd_com();

        delay(3200);


// load a command for display on, cursor on and blinking off

        temp1 = 0x0C;

        lcd_com();

        delay(800);


// command for cursor increment after data dump

        temp1 = 0x06;

        lcd_com();

        delay(800);
```

```
        temp1 = 0x80;
        lcd_com();
        delay(800);
}

void lcd_data(void)
{
        temp = temp1 & 0xf0;
        wr_dn();
        temp= temp1 & 0x0f;
        temp= temp << 4;
        wr_dn();
        delay(100);
}

void wr_dn(void)                        ////write data reg
```

```
{
        IO0CLR  = 0x000000FC;       // clear the port lines.
        IO0SET = temp;                          // Assign the value to the PORT lines
        IO0SET = 0x00000004;        // set bit  RS = 1
        IO0SET = 0x00000008;        // E=1
        delay(10);
        IO0CLR = 0x00000008;
}

void lcd_com(void)
{
        temp = temp1 & 0xf0;
        wr_cn();
        temp = temp1 & 0x0f;
        temp = temp << 4;
        wr_cn();
```

```
        delay(500);
}

void wr_cn(void)                //write command reg
{
        IO0CLR  = 0x000000FC;           // clear the port lines.
        IO0SET          = temp;                         // Assign the value to the PORT lines
        IO0CLR  = 0x00000004;           // clear bit  RS = 0
        IO0SET          = 0x00000008;           // E=1
        delay(10);
        IO0CLR  = 0x00000008;
}

void clr_disp(void)
{
// command to clear lcd display
```

```
        temp1 = 0x01;
        lcd_com();
        delay(500);
}

void delay(unsigned int r1)
{
        for(r=0;r<r1;r++);
}

void init_port()
{
        IO0DIR = 0x000000FC;        //configure o/p lines for lcd
        IO1DIR = 0XFFF0FFFF;
}
```

```
_____

// LEd and Buzzer turns ON and OFF for every sec as set by Timer0 match interrupt


#include <LPC21xx.h>

#include "Timer0.h"


unsigned int delay=0;


int main ()

{

        InitTimer0(1000);                       // for 1 sec delay -

        IO0DIR = 0x00000200 ;                   // Configure P0.9 as output

        IO1DIR          |=0X02000000;                 // for LED  P1.25

        T0TCR = 0x01;                                 // start timer
```

```
	while(1)
	{
		IO0SET = 0x00000200 ;	//Buzzer ON
		IO1SET = 0x02000000 ;	//led ON

		while(tmr_ovflg == 0);
		tmr_ovflg =0;

		IO0CLR = 0x00000200 ;	//Buzzer off
		IO1CLR = 0x02000000 ;	//led off

	while(tmr_ovflg == 0);
		tmr_ovflg =0;
	}
}
```