Program 1 (a)

Design and develop an assembly language program to search a key element “X” in a list of ‘n’16-bit numbers. Adopt Binary search algorithm in your program for searching

```
.model small
        printf   macro msg
                 mov    ah,09h
                 mov     dx,offset msg
                 int     21h
                 endm
        exit     macro
                 mov    ah,4ch
                 int     21h
                 endm
        .data

        a        dw      1111h,2222h,3333h,4444h,5555h
        n        dw      ($-a)/2
        key      dw      5555h
        low_     dw      ?
        high_    dw      ?
        msg1     db      ‘successful search’
        msg2     db      ‘unsuccessful search’

.code
        mov      ax, @data
        mov      ds, ax
        mov      low,0
        mov      ax,n
        mov      high_,ax
        dec      high_

l1:     mov      si,low_
        cmp      si,high_
        jg       l4
        add      si,high_
        shr      si,1
        mov      mid,si
        mov      ax,key
        mov      si,mid
        shl      si,1
        cmp      ax,a[si]
        jne      l2
        printf   msg1
        exit
```

```
     l2:  cmp     ax,a[si]
          jg      l3
          mov     ax,mid
          dec     ax
          mov     high_,ax
          jmp     l1

l3:       mov     ax,mid
          inc     ax
          mov     low_,ax
          jmp     l1

l4:       printf  msg2
          exit
          end
```

Output :

```
Successful search
```

Progarm 1b

Design and develop an assembly program to demonstrate BCD Up-Down Counter (00-99) on the Logic Controller Interface.

```
.model small
.code
        mov     dx, 0e403h
        mov     al, 80h         ; 80h- all ports o/p
        out     dx, al
        mov     al, 0
        mov     dx, 0e400h
up:     out     dx, al
        call    delay
        add     al, 1
        daa                     ; convert hex no. to bcd
        call    stop
        cmp     al, 99h
        jne     up              ; if no. < 99h increment and display
down:   out     dx, al          ;else decrement and display
        call    delay
        add     al, 99h
        daa
        call    stop
        cmp     al, 99h
        jne     down
stop:   push    ax
        mov     ah, 1
        int     16h
        jne     exit
        pop     ax
        ret
exit:   mov     ah, 4ch
        int     21h
delay:  mov     si, 2fffh
ret2:   mov     di, 0ffffh
ret1:   dec     di
        jnz     ret1
        dec     si
        jnz     ret2
        ret
        end
```

Program 2a

Sort a given set of ‘n’ numbers in ascending order using the bubble sort algorithm.

```
.model small
.data
        array   db      85h, 95h, 25h, 45h, 55h, 15h, 65h, 45h
        len     dw      $ - array                   ; length of array
.code
        mov     ax, @data
        mov     ds, ax
        mov     bx,  len                            ; bx = no. of iterations
        dec     bx
np:     mov     cx, bx                              ; cx = no. of comparison in each iteration
        mov     si, 0
ni:     mov     al, array[si]
        inc     si
        cmp     al, array[si]
        jbe     next                                ; for descending order jae next
        xchg    al, array[si]                       ; exchange if (al< [si+1])
        mov     array[si-1],al
next:   loop    ni
        dec     bx
        jnz     np
        mov     ah, 4ch
        int     21h
        end
```

Output :

```
>d array
15 45 45 55 65 85 95
```

Program 2b

Design and develop an assembly program to display messages “FIRE” and “HELP” alternately with flickering effects on a 7-segment display interface for a suitable period of time. Ensure a flashing rate that makes it easy to read both the messages (Examiner does not specify these delay values nor is it necessary for the student to compute these values).

```
.model small
.data
        data1   db      86h, 0afh, 0cfh, 8eh ; seven-segment code for e,r,i,f respectively
        data2   db      8ch, 0c7h, 86h, 89h ; seven-segment code for p,l,e,h respectively
.code
        mov     ax, @data
        mov     ds, ax
        mov     dx, 0e403h
        mov     al, 80h
        out     dx, al
bak:    lea     si, data1               ;load the effective address of erif
        call    display                 ;display fire
        call    delay                   ;delay to have flickering effect
        lea     si, data2               ;load the effective address of pleh
        call    display                 ;display help
        call    delay                   ;delay to have flickering effect
        mov      ah, 1                  ;check for keystroke to stop display
        int     16h
        jz      bak
        mov     ah, 4ch
        int     21h
display: mov    cx, 04          ;cl- no. of letters to be displayed
bak2:   mov     bl, 08          ;bl- no. of segment in each led
        mov     al, [si]
next:   rol     al, 01
        mov     dx, 0e401h              ;each segment is outputted at a time
        out     dx, al
        push    ax
        mov     dx, 0e402h              ;port c to generate a serial clock pulse
        mov     al, 0ffh
        out     dx, al
        mov     al, 00
        out     dx, al
        dec     bl
        pop     ax
        jnz     next
        inc     si
        loop    bak2
        ret
```

```
delay:  mov     si, 2fffh
rep2:   mov     di, 0ffffh
rep1:   dec     di
        jnz     rep1
        dec     si
        jnz     rep2
        ret
        end
```

Result: The strings fire and help will be displayed in blinking fashion with an appropriate delay between each display so that the output can be read easily.

Program 3a

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

```
.model small
.stack 100
.data
        str     db      'malayalam'
        n       db      $-str
        rstr    db      10 dup(0)
        msg1    db      'String is palindrome$'
        msg2    db      'Not a palindrome$'

.code
        mov     ax, @data
        mov     ds, ax
        mov     es, ax
        mov     cl, n
        dec     cl
        mov     di, cx
        inc     cx
bak:    mov     ah, str[di]                         ;to reverse the string
        mov     rstr[si], ah                        ; to reverse the string
        inc     si
        dec     di
        loop    bak
        lea     si, str
        lea     di, rstr
        cld
        mov     cl,n                                ;cl=size of string
        repe    cmpsb                               ;to compare the 2 strings
        je      dmsg1                               ;jump on equal to found
        lea     dx, msg2                            ;if not equal display not found
        jmp     xit
dmsg1: lea      dx, msg1
xit:    mov     ah, 09h
        int     21h
        mov     ah, 4ch
        int     21h
        end
```

Output :

```
String is palindrome
```

Program 3b

Design and develop an assembly language program to
Generate the Sine Wave using DAC interface (The output of the DAC is to be displayed on the CRO).

```
.model small
.data
        a       db      00, 22, 43, 63, 81, 97, 109, 119, 125, 127     ; points to plot
.code
        mov     ax, @data
        mov     ds, ax
        mov     al, 80h
        mov     dx, 0e403h
        out     dx, al
        mov     si, 0ffffh              ;si-no. of sine waveforms
        mov     bx, 0                   ;bx-count no. of points
        mov     dx, 0e401h              ;portb as o/p port
b1:     mov     al, a[bx]
        add     al, 127                 ;0-255 levels possible. median is 127
        out     dx, al
        inc     bx
        cmp     bx, 9
        jb      b1
b2:     mov     al, a[bx]
        add     al, 127
        out     dx, al
        dec     bx
        cmp     bx, 0
        jnz     b2
b3:     mov     al, a[bx]
        mov     cl, 127
        sub     cl, al
        mov     al, cl
        out     dx, al
        inc     bx
        cmp     bx, 9
        jb      b3
b4:     mov     al, a[bx]
        mov     cl, 127
        sub     cl, al
        mov     al, cl
        out     dx, al
        dec     bx
        cmp     bx, 0
        jnz     b4
        dec     si
```

```
jnz     b1
mov     ah, 4ch
int     21h
end
```

Program 4a

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

```
.model    small
.data
          n         dw       5
          r         dw      3
          ncr       dw      0
.code
          mov       ax, @data
          mov       ds, ax
          mov       ax, n
          mov       bx, r
          call      ncrpro                        : recursive procedure to calculate ncr
          mov       ah, 4ch
          int       21h
ncrpro:   cmp       bx, ax                        ; if bx == ax, ncr = 1
          je        res1
          cmp       bx, 0                         ; if bx == 0, ncr = 1
          je        res1
          cmp       bx, 1                         ; if bx == 1, ncr = 1
          je        resn
          dec       ax
          cmp       bx, ax                        ; compare bx & ax
          je        incr                          ; if bx == ax, ncr = 1
          push      ax
          push      bx
          call      ncrpro
          pop       bx
          pop       ax
          dec       bx
          push      ax
          push      bx
          call      ncrpro
          pop       bx
          pop       ax
          ret
res1:     inc       ncr
          ret
incr:     inc       ncr
resn:     add       ncr, ax
          ret
          end
```

Output:

```
> d ncr 0A
```

Program 4b

Generate a Half Rectified Sine waveform using the DAC interface. (The output of the DAC is to be displayed on the CRO).

```
.model  small
.data
        a       db      00, 22, 43, 63, 81, 97, 109, 119, 125, 127
.code
        mov     ax, @data
        mov     ds, ax
        mov     al, 80h
        mov     dx, 0e403h
        out     dx, al
        mov     cx, 0ffffh
        mov     bx, 0
        mov     dx, 0e400h              ;port A as o/p port
b1:     mov     al, a[bx]
        add     al, 127
        out     dx, al
        inc     bx
        cmp     bx, 9
        jb      b1
b2:     mov     al, a[bx]
        add     al, 127
        out     dx, al
        dec     bx
        cmp     bx, 0
        jnz     b2
        mov     si, 14
rpt:    mov     al, 127         ;loop for half rectified wave
        out     dx, al
        dec     si
        jnz     rpt
        loop    b1
exit:   mov     ah, 4ch
        int     21h
        end
```

Program 5a

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.

```
.model small
.code
        mov     ah, 2ch                 ; func(2ch), to get the system time
        int     21h                     ; stores hrs, mins, secs in ch, cl, dh respectively
        mov     al, ch
        call    disp
        mov     dl, ':'
        mov     ah, 2
        int     21h
        mov     al, cl
        call    disp
        mov     dl, ':'
        mov     ah, 2
        int     21h
        mov     al, dh
        call    disp
        mov     ah, 4ch
        int     21h
disp    proc    near
        aam                             ; converts hex values to unpacked bcd format
        add     ax, 3030h
        mov     bx, ax
        mov     dl, ah
        mov     ah, 02
        int     21h
        mov     dl, bl
        int     21h
        ret
disp    endp
        end
```

Output :

```
09:55:45
```

Program 5b

Design and develop an assembly program to drive a Stepper Motor interface and rotate the motor in specified direction (clockwise or counter-clockwise) by N steps (Direction and N are specified by the examiner). Introduce suitable delay between successive steps.

```
.model small
.code
        mov     cx, 20
        mov     dx, 0e403h
        mov     al, 80h
        out     dx, al
        mov     dx, 0e400h
        mov     al, 88h
rep1:   out     dx, al
        call    delay
        rol     al, 1           ; gives clockwise motion

        ; ror   al,1            ; gives anti-clockwise motion

        dec     cx
        jnz     rep1
        mov     ah, 4ch
        int     21h
delay:  mov     si, 2fffh
back2:  mov     di, 0ffffh
back1:  dec     di
        jnz     back1
        dec     si
        jnz     back2
        ret
        end
```

Program 6

To write and simulate ARM assembly language programs for data transfer, arithmetic and logical operations (Demonstrate with the help of a suitable program).

1. **Data Transfer**.

   The below assembly level program moves the 32 bit data from register to register.

```
area  movt, code, readonly
entry
mov r1,#0005           ; Mov immediate 32 bit data to r1
mov r2,#0002           ; Mov immediate 32 bit data to r1
mov r3,r1              ; Register-Register movement
mov r4,r2              ; Register-Register movement
stop b stop             ; End of the program
end
```

Arithmetic Operations A. Addition, Subtraction and Multiplication:

```
area addt, code, readonly
entry
mov r1,#0005 ; Mov immediate 32 bit data to r1
mov r2,#0002 ; Mov immediate 32 bit data to r2
add r3,r2,r1 ; Add the contents present in r2 with the contents of r1 and store in r3
sub r5,r1,r2 ; Subtract; r5 = r1-r2
mul r6,r1,r2 ; Multiply
mov r7,r6
add r7,#2 ; Add immediate data
mov r8,r7
sub r8,#3 ; Subtract immediate data
mov r9,r8
stop b stop
end
```

Logical operations: To perform AND, Logical Shift operations,

```
area dis,code,readonly
entry
 mov r0,#0x83
mov r1,r0
and r1, # 0Xf0                   ; Perform Logical AND operation
 mov r2,r1
lsr r2, #4                 ; Perform Logical right Shift operation
mov r3, r0 and r3, # 0X0f
stop b stop end
```

Program 7

To write and simulate C Programs for ARM microprocessor using KEIL (Demonstrate with the help of a suitable program)

To write a C program to Blink a LED /Port Pin with LPC 2148 ARM 7 Microcontroller.

```
 #include <lpc214x.h>                    //Header File “x” can be wrt to controller
unsigned int delay;
 int main(void)
 {
 IO1DIR = (4);                           // Bit No 4 (0100) will be activated
while(1)                        // If True
{
IO1CLR = (04); // Clear Bit 04 of GPIO1 for
for (delay=0 ;delay <5000; delay++);// call delay
IO1SET = (04);
for (delay=0 ;delay <5000; delay++);// call delay
}
}
```

Program 8

Design and develop an assembly program to read the status of two 8-bit inputs (X & Y) from the Logic Controller Interface and display X*Y.

```
.model small
.data
        porta   dw      0e400h
        portb   dw      0e401h          ; These are the port addresses of 8255
        portc   dw      0e402h
        portd   dw      0e403h
.code
        mov     ax, @data               ; Initialization of data segment
        mov     ds, ax
        mov     al, 82h                 ; Control Word, Port A as output & Port B as input
        mov     dx, portd
        out     dx, al                  ; Place the control word in the control register
        mov     dx, portb
        in      al, dx          ; Read the first 8-bit value, i.e., X & store it in bl register
        mov     bl, al
        call    delay                   ; wait for some time
        in      al, dx                  ; Read the next 8-bit value, i.e., Y
        mul     bl            ; Multiply two 8-bits, i.e., X * Y. Result will be in ax register
        dec     dx
        out     dx, al                  ; lower 8-bits is displayed first
        call    delay
        mov     al, ah                  ; after a delay higher 8-bits is displayed
        out     dx, al
        mov     ah, 4ch                 ; terminate the program
        int     21h

delay   proc
        push    ax                      ; push the register value
        push    cx
        mov     ax, 6fffh               ; count values are moved to the bx and cx registers
agn1:   mov     cx, 0ffffh
agn:    loop    agn                     ; do no operation for the above count values
        dec     ax
        jnz     agn1
        pop     cx
        pop     ax                      ; pop the register value
        ret
delay   endp                            ; end the procedures
        end
```

Program 9

Generate a Fully Rectified Sine waveform using the DAC interface. (The output of the DAC is to be displayed on the CRO).

```
.model  small
.data
        a       db      00, 22, 43, 64, 82, 97, 110, 119, 125, 127
.code
        mov     ax, @data
        mov     ds, ax
        mov     al, 80h
        mov     dx, 0e403h
        out     dx, al
        mov     cx, 0ffffh      ;cx-no. of waveforms
        mov     bx, 0
        mov     dx, 0e401h
b1:     mov     al, a[bx]
        add     al, 128
        out     dx, al
        inc     bx
        cmp     bx, 9
        jb      b1
b2:     mov     al, a[bx]
        add     al, 128
        out     dx, al
        dec     bx
        cmp     bx, 0
        jnz     b2
        loop    b1
        mov     ah, 4ch
        int     21h
        end
```

Program 10

To interface Stepper motor with ARM processor-- ARM7TDMI/LPC2148. Write a program to rotate stepper motor

```
#include <LPC214X.h>
void delay();
void delay()
{
int i,j;
For (i=0; i<0xff; i++)
          For (j=0; j<0x25; j++);
          }
int main()
{
IO0DIR=0x000F0000;          ; Consider ARM port Pin from 16-19
                            ; And set these pins
While (1)
{
//while (IO0PIN & 0x00008000);
//while (! (IO0PIN & 0x00008000));
IO0PIN=0x00010000;
delay ();
IO0PIN=0x00020000;
delay ();
IO0PIN=0x00040000;
delay ();
IO0PIN=0x00080000;
delay();
}
                            }
```