MICROPROCESSORS AND MICROCONTROLLERS: Lab Experiments

Lab Program	Program Details
	Software Programs using 8086
1	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.
2	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.
3	Read an alphanumeric character and display its equivalent ASCII code at the center of the screen.
4	Reverse a given string and check whether it is a palindrome or not.
5	Read two strings, store them in locations STR1 and STR2. Check whether they are equal or not and display appropriate messages. Also display the length of the stored strings.
6	Develop an assembly language program to compute nCr using recursive procedure. Assume that 'n' and 'r' are non-negative integers.
7	Read the current time from the system and display it in the standard format on the screen.
8	Write a program to simulate a Decimal Up-counter to display 00-99.
9	Read a pair of input co-ordinates in BCD and move the cursor to the specified location on the screen.
10	Write a program to create a file (input file) and to delete an existing file.

Microprocessors & Microcontrollers

Lab programs - Using 8086

1. 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.

Code -

```
.model small
display macro msg
 lea dx, msg
 mov ah, 09h
 int 21h
endm
.data
list db 01h, 05h, 07h, 10h, 12h, 14h
number equ ($-list)
Key db 011h
msg1 db 0dh, 0ah, "element found in the list... $"
msg2 db 0dh, 0ah, "search failed! element not found... $"
.code
start : mov ax, @data
 mov ds, ax
 mov ch, number-1
 mov cl, 00h
Again : mov si, offset list
 xor ax, ax
 cmp cl, ch
 je next
 jnc failed
```

```
next : mov al, cl
   add al, ch
    shr al, 01h
    mov bl, al
    xor ah, ah
    mov bp, ax
   mov al, ds:[BP][SI]
   cmp al, key
   je success
     jc inclow
   mov ch, bl
   dec ch
   jmp again
    inclow : mov cl, bl
      inc cl
     jmp again
 success : display msg1
      jmp final
  failed: display msg2
      final : mov ah, 4ch
         int 21h
       end start.
```

———— X ————

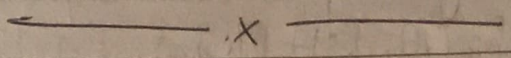
2. design & develop an assembly program to sort a given set of 'n' 16-bit No in ascending order. Use bubble sort.

Code -

```
.Model small
display macro msg
  lea dx, msg
  mov ah, 09h
  int 21h
endm
.data
  list db 02h, 01h, 09h, 0F4h, 34h, 05h
  number equ $-list
msg1 db 0dh, 0ah, "1 >> Ascending order $"
msg2 db 0dh, 0ah, "2 >> descending order $"
msg3 db 0dh, 0ah, "3 >> exit $"
msg4 db 0dh, 0ah, "enter your choice :: $"
msg5 db 0dh, 0ah, "Invaild choice entered $"
  .code
  start: mov ax, @data
    mov ds, ax
    lea si, list
  mov ch, number-1
   display msg1
   display msg2
   display msg3
   display msg4
  mov ah, 01h
    int 21h
  sub al, 30h
   cmp al, 01h
    je ascsort
     cmp al, 02h
```

```
        je dessort
        cmp al, 03h
        je final
ascsort: mov bl, 00h
Again : mov si, offset list
        mov cl, 00h
        mov bh, ch
        sub bh, bl
Npass : cmp cl, bh
        jnc next
        mov al, [si]
        mov bp, 01h
        cmp al, DS : [BP][SI]
        jc - nope
        Xchg Al, [si+1]
        Xchg [si], al
- Nope: Inc cl
        Inc si
        jmp Npass
next : inc bl
        cmp bl, ch
        jc again
        jmp final
dessort: mov bl, 00h
again1 : mov si, offset List
        mov cl, 00h
        mov bh, ch
        sub bh, bl
Npass1 : cmp cl, bh
        jnc next
        mov Al, [si]
        mov bp, 01h
```

```
cmp al, ds:[BP][SI]
jnc_nope1
xchg al, [SI+1]
xchg [SI], al
Nope1: inc cl
       inc SI
       jmp npass1
Next1: inc bl
       cmp bl, ch
       jb again1
final: mov ah, 4Ch
       int 21h
       end start
```



3. Read an Alphanumeric character & display its equivalent ascii code at the center of the screen.

Code -

```
.model small
display macro msg
 lea dx, msg
 mov ah, 09h
 int 21h
endm
dispchar macro
   mov ah, 02h
   int 21h
endm
.data
msg1 db 0dh, 0ah, "enter an Alphanumeric character: $"
msg2 db 0dh, 0ah, " Not an ---------- : $"
.code
start : mov ax, @data
   mov ds, ax
 display msg1
mov ah, 01h
int 21h
call check
 jc error
  push ax
mov ah, 00h
   mov al, 03h
    int 10h
mov ah, 02h
mov bh, 00h
mov dh, 12d
 mov dl, 40D
 int 10h
```

```
pop ax
aam
push ax
mov al, ah
xor ah, ah
aam
add ax, 3030h
mov dl, ah
push ax
dispchar
pop ax
mov dl, al
dispchar
pop ax
add al, 30h
mov dl, al
dispchar
mov ah, 07h
int 21h
jmp final
error: display msg2
jmp final
cmp al, 30h
je fret
jc err
cmp al, 39h
je fret
jnc next
jc fret
next: cmp al, 41h
je fret
jc err
```

```
cmp al, 5ah
  je fret
 jnc next1
  jc fret
next1: cmp al, 61h
  je fret
  jc err
cmp al, 7ah
  je fret
  jnc err
  jc fret
err: stc
 Ret
 fret: clc
   ret
 check endp
 final: mov ah, 4ch
    int 21h
   end start
```

---

4. Reverse a given String & check whether its a palindrome or not.

Code:

```
.model small
display macro msg
 lea dx, msg
 mov ah, 09h
 int 21h
endm
.data
msg1 db 0dh, 0ah, "enter string: $"
msg2 db 0dh, 0ah, "reverse string: $"
msg3 db 0dh, 0ah, "Input string is palindrome $"
msg4 db 0dh, 0ah, "Input is not a palindrome $"
 string db 80h dup(?)
 RString db 80h dup(?)
.code
Start: mov @data
 mov ds, ax
 display msg1
 mov si, offset string
 xor cl, cl
again: mov ah, 01h
 int 21h
 cmp al, 0dh
 je next
 mov [si], al
 inc si
 inc cl
 jmp again
next: mov [si], byte ptr '$'
 dec si
 mov ch, cl
```

```
    mov dl, offset Rstring
back: mov al, [si]
      mov [di], al
      dec si
      inc di
      dec ch
      jnz back
      mov [dl], byte ptr '$'
      display msg2
display Rstring
 mov si, offset string
 mov di, offset Rstring
Ag: mov al, [si]
     cmp al, [di]
     jne fail
       inc si
       inc di
     dec cx
     jz success
       jmp ag
fail: display msg4
      jmp final
Success: display msg3
  final: mov ah, 4ch
         int 21h
       end.
```

---

5. Read 2 strings store them in str1 & str2. & then whether they are equal or not & display msg & the length of the string.

Code:
```
.model small
display macro msg
  lea dx, msg
mov ah, 09h
int 21h
endm
.data
msg1 db 0dh, 0ah, "enter 1st String: $"
msg2 db 0dh, 0ah, "enter 2nd String: $"
msg3 db 0dh, 0ah, "length of 1st String: $"
msg4 db 0dh, 0ah, "length of 2nd String: $"
msg5 db 0dh, 0ah "Strings are equal $"
msg6 db 0dh, 0ah, "Strings are not equal $"
  String1 db 80h dup(?)
  String2 db 80h dup(?)
    .code
start: mov ax, @data
  mov ds, ax
  display msg1
mov si, offset string1
call read:str
  mov bl, cl
  display msg2
mov si, offset String2
call readstr
  push bx
    push cx
  display msg3
```

```
mov al, bl
call len_dis
display msg4
mov al, cl
call len_dis
  pop cx
  pop bx
cmp cl, bl
jne fail
mov si, offset string1
mov di, offset string2
cld
chk : mov al, [si]
    cmp al, [di]
    jne fail
      inc si
      inc di
      dec cl
    jnc chk
      display msg5
  jmp final
      len_dis proc near
      xor ah, ah
      add al, 00h
      aam
      add ax, 3030h
       mov bh, Al
      mov dl, Ah
      mov ah, 02h
       int 21h
     mov al, bh
      mov ah, 02h
```

```
    int 21h
  ret
   len_disp endp
readstr proc near
    xor d, cl
back: mov ah, 01h
    int 21h
    cmp al, 0dh
    je finish
   mov [si], al
    inc si
    inc cl
    jmp back
finish: mov [si], byte ptr '$'
    ret
  readstr endp
fail: display msg6
 final: mov ah, 4ch
    int 21h
  end start
```

---

store
67

6. Develop an Assembly language program to Compute nCr Using recursive procedure.

Code: N db 6; AIM is to find → $^6C_3$

```
R db 3
answer db 0
.code
Initds
mov al, N
mov bl, r
call ncr
mov al, answer
aam
add ax, 3030h
mov bx, ax
putchar bh
putchar bl
exit
ncr proc
cmp bl, 0
co = 1
jnc go1
add answer, 1
ret
go1 : cmp bl, al
cn = 1
jnc go2
add answer, 1
ret
go2 : cmp bl, 1
c1 = N
jne go3
add answer, al
```

```
ret
go3: dec al
  CN-1 = N
   cmp bl, al
   jne go4
  inc al
add answer, al
  ret
go4: push ax
   push bx ; N-1
  call ncr
   pop bx
    pop ax
 dec bx
   push ax
    push bx
   call ncr
    pop bx
  pop ax
     ret
  ncr endp
   end
```

---

7. Read the current time from the system & display it in the standard format on screen.

Code:

```
.model small
.code
mov ah, 2ch
int 21h
mov al, ch
aam
mov bx, ax
call disp
mov dl, 20h
mov ah, 02h
int 21h
mov al, cl
aam
mov bx, ax
call disp
mov al, 20h
mov ah, 02h
int 21h
mov al, dh
aam
mov bx, ax
int 21h
disp proc near
mov al, bh
add dl, 30h
mov ah, 02h
int 21h
mov dl, bl
add dl, 30h
int 21h — ret — disp endp — end
```

———— × ————

8. Write a program to stimulate a decimal Up-counter to display 00-99.

Code:

```
.model small
.code
mov cl, 00
  mov ah, 00h
  mov al, 03h
  int 10h
back: mov bh, 00h
  mov dh, 00h
  mov dl, 00h
  mov ah, 02h
  int 10h
mov al, cl
add al, 00h
  aam
add ax, 3030h
mov ch, al
  mov dl, ah
mov ah, 02h
int 21h
mov al, ch
mov ah, 02h
  int 21h
call delay
inc cl
xor ax, ax
  cmp cl, 100d
jne back
  je last
  delay proc near.
```

```
        push ax
        push bx
        push cx
    mov cx, 0FFFh
ag: mov bx, 0FFFFh
ag1: NOP
    dec bx
    jnz ag1
        dec cx
    jnz ag
        pop cx
    pop bx
        pop ax
    ret
        delay endp
last: mov ah, 4Ch
    int 21h
    end
```

---

9. Read a pair of input co-ordinates in bcd & move the cursor to the specified location.

code:

```
.model small
disp macro msg
lea dx, msg
mov ah, 09h
int 21h
endm
.data
row db 02 dup(0)
col db 02 dup(0)
msg1 db 0dh,0ah," enter x-co-ordinate : $"
msg2 db 0dh,0ah," enter y ——o—— : $"
.code
mov ax, @data
mov ds, ax
disp msg1
mov si, offset row
call read
disp msg2
mov si, offset col
call read
mov si, offset row
mov ah, [si]
inc si
mov al, [si]
sub ax, 3030h
aad
mov dh, al
mov si, offset col
mov ah, [si]
inc si
```

store
67

```
    mov al, [si]
    sub ax, 3030h
add
    mov dl, al
      mov ah, 00
     mov al, 03h
      int 10h
      mov ah, 02h
      int 10h
      jmp final
read  proc near
      mov cx, 02h
back: mov ah, 01h
        int 21h
      mov [si], al
        inc si
        dec cx
    jnz back
        ret
    read endp
    final: mov ah, 01h
        int 21h
      mov ah, 4ch
        int 21h
        end.
```

---------- X ----------

10. Write a program to create a file & delete existing file.

Code:

```
.model small
disp macro msg
lea dx, msg
mov ah, 09h
int 21h
endm
.data
msg1 db 0dh, 0ah, "enter file name for creation: $"
msg2 db 0dh, 0ah, "file created successfully! $"
msg3 db 0dh, 0ah, "file creation failed $"
msg4 db 0dh, 0ah, "enter file name for deletion $"
msg5 db 0dh, 0ah, "file deleted $"
msg6 db 0dh, 0ah, "file deletion failed $"
fname1 db 10 dup(0)
fname2 db 10 dup(0)
.code
mov ax, @data
mov ds, ax
disp msg1
mov si, 00
back1: mov ah, 01h
int 21h
cmp al, 0dh
je next1
mov fname1[si], al
inc si
jmp back1
next1: mov fname1[si], '$'
lea dx, fname1
mov cx, 00
```

```
mov ah, 3ch
int 21h
 jc cfail
disp msg 2
   jmp del
cfail : disp msg 3
 del : disp msg 4
 mov si, 00
back 2: mov ah, 01h
  int 21h
   cmp al, 0dh
  jc next 2
   mov fname 2 [si], al
  inc si
    jmp back 2
next 2: mov fname 2 [si], '$'
     lea dx, fname 2
 mov ah, 41h
    int 21h
     jc dfail
 disp msg 5
    jmp last
dfail : disp msg 6
last : mov ah, 4ch
    int 21h.
     end
```
