

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.

. ~~model~~ ^msmall

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 05h

msg1 db 0dh, 0ah, "Element found in the list---\$"

msg2 db 0dh, 0ah, "Search failed!! Element not found
in the list \$"

. code

start: mov ax, @data
 mov ds, ax
 mov cx, number-1
 mov d, 00h

AGAIN : ~~le~~ lea si, list
xor ax, ax
cmp cl, ch
je next
jne 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
je 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 : max ah, uch
int 21h

end start

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX -

Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.

Object filename [binary.OBJ]:

Source listing [NUL.LST]:

Cross-reference [NUL.CRF]:

51468 + 465076 Bytes symbol space free

0 Warning Errors

0 Severe Errors

C:\>link binary.obj

Microsoft (R) Overlay Linker Version 3.60

Copyright (C) Microsoft Corp 1983-1987. All rights reserved.

Run File [BINARY.EXE]:

List File [NUL.MAP]:

Libraries [LIB]:

LINK : warning L4021: no stack segment

C:\>binary.exe

SEARCH FAILED !! ELEMENT NOT FOUND IN THE LIST

C:\>_

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.

model small

Display macro msg

lea dx, msg

mov ah, 09h

endm

- data

N DB 5

list db 02H, 01H, 34H, 0F4H, 09H, 05H

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, "Invalid choice entered --- \$"

- code

mov ax, @data

mov ds, ax

mov cx, N

dec cx

display msg1

display msg2

display msg3

display msg4


```
mov ah, 01h
int 21h
sub al, 30H
cmp AL, 01H
JE AS
cmp al, 02h
JE des
cmp al, 03H
JE final
```

```
AS: outputloop1: mov ch, cl
               mov si, 00H
```

```
inloop1: mov AL, list[si]
           inc si
           cmp al, text[si]
           JL no_exchg1
           xchg AL, list[si]
           mov list[si-1], al
```

```
no_exch1: dec ch
           jnz inloop1
           dec cl
           jnz outloop1
           mov ah, 4Ch
           int 21h
```

des:

```
outloop2: mov ch, ch
           mov si, 00h
```

```
inloop2: mov al, text[si]
           inc si
           cmp al, list[si]
```


inc no-exchg
xchg al, list[si]
mov list[si-1], al

No-exchg2: dec di
jnz inloop2
dec cl
jnz outloop2
mov ah, 4ch
int 21h

final: mov ah, 4ch
int 21h

end.

local															reg	
source1 CS:IP (ACTIVE)																
052A:003D	8AE9	MOV	CH,CL												AX = 4C01	
052A:003F	BE0000	MOV	SI,0000												BX = 0000	
052A:0042	8A840B00	MOV	AL,Byte Ptr [SI+000B]												CX = 0000	
052A:0046	46	INC	SI												DX = 0042	
052A:0047	3A840B00	CMP	AL,Byte Ptr [SI+000B]												SP = 0000	
052A:004B	7208	JB	0055												BP = 0000	
052A:004D	86840B00	XCHG	AL,Byte Ptr [SI+000B]												SI = 0001	
052A:0051	88840A00	MOV	Byte Ptr [SI+000A],AL												DI = 0000	
052A:0055	FECF	DEC	CH												DS = 0532	
052A:0057	75E9	JNZ	0042												ES = 051A	
052A:0059	FEC9	DEC	CL												SS = 0529	
052A:005B	75E0	JNZ	003D												CS = 052A	
052A:005D	B44C	MOV	AH,4C												IP = 0061	
052A:005F	CD21	INT	21												FL = 0247	
052A:0061	8AE9	MOV	CH,CL												NU UP EI PL	
052A:0063	BE0000	MOV	SI,0000												ZR NA PE CY	
command																
0532:0000	E0 B4 4C CD 21 B4 4C CD-21 00 05 01 02 09 34 F4	..L.!														
0532:0010	05 0D 0A 31 3E 3E 41 53-43 45 4E 44 49 4E 47 20	...1>														
>																
←																
>↓																
<F8=Trace> <F10=Step> <F5=Go> <F6=Window> <F3=Display> <ESC=Cancel>																

.model small
- data

; Alphanumeric characters ASCII

msg1 db 0dh, 0ah, "enter alphanumeric character &"
res db 02 dup(0)

.code

mov ax, @data

mov ds, ax

lea dx, msg1

call disp

mov ah, 01h

int 21h

mov bl, al

mov cl, 4

shl al, cl

cmp al, 0ah

jc digit

ADD AL, 07h

digit: add al, 30h

mov res, al

and bl, 0fh

cmp bl, 0ah

jc digit1

add bl, 07h

digit1: add bl, 30h

mov res+1, bl

mov ah, 00h

mov al, 03h

int 10h

; TEXT MODE

mov ah, 02h

; Set the cursor pos

mov bh, 00h

; Page number

mov dh, 0ch

; Row 100 is 30h

mov al, 28h

; Column Val

int 10h

mov res+2, '\$'

lea dx, res

call disp

mov ah, 4ch

int 21h

disp proc near

mov ah, 09h

int 21h

ret

disp endp

end

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX



61

C:\>

. model small

; Comparing two strings program

display macro msg

lea dx, msg

mov ah, 09h

int 21h

endm

. data

msg1 db 0dh, 0ah, "enter first string : \$"

msg2 db 0dh, 0ah, "enter second string : \$"

msg3 db 0dh, 0ah, "length of first string: \$"

msg4 db 0dh, 0ah, "length of second 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

~~call readstr~~

mov si, offset string1

call readstr

mov bl, cl

display msg2

mov si, offset string2

call readstr

; store the length of the first string


```

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

```

; compare the lengths
 ; if lengths are equal, process
 next statement

```

mov si, offset string1
mov di, offset string2
cld

```

```

chk: mov al, [si]      ; compare both the string
      cmp al, [di]
      jne fail
      inc si
      inc di
      dec cl
      jnz 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 dl, bh
mov ah, 02h
int 21h

```

```

ret
len-dis endp
readstr proc near
xor cl, cl
back: mov ah, 01h
int 21h
cmp al, 0dh
je finish
mov [si], al
inc si
inc cl
jump back

```

```

finish: mov [si], byte ptr '$'
ret

```

```

readstr endp

```

```

fail: display msg6

```


final: mov ah, 4ch
int 21h

end start

C:\>strcmp.exe

ENTER FIRST STRING : ala

ENTER SECOND STRING : ala

LENGTH OF FIRST STRING: 03

LENGTH OF SECOND STRING: 03

---STRINGS ARE EQUAL---

C:\>

To calculate NCR using recursion.

- model small

display macro msg

lea dx, msg

mov ah, 09h

int 21h

endm

- data

msg1 db 0dh, 0ah, "Enter the value of n: "

msg2 db 0dh, 0ah, "Enter the value of r: "

msg3 db 0dh, 0ah, "Calculated successfully: "

msg4 db 0dh, 0ah, "Calculation Failed: "

nrc dw 0

- code

mov ax, @data

mov ds, ax

xor ax, ax

xor bx, bx

display msg1

xor ax, ax

call read

push ax

display msg2

xor ax, ax

call read

mov bx, ax

pop ax

call calc
xor cx, ~~cx~~ cx
mov cx, ecx
mov ebx, ecx
int 21h

calc proc near
cmp ax, bx
je r1
cmp bx, 0
je r1
cmp bx, 1
je r3
dec ax
cmp bx, ax
je r2
push ax
push bx
call calc
pop bx
pop ax
dec bx
push ax
push bx
call calc
pop bx
pop ax
ret

r1 : inc mcr
ret

r2 : inc mcr

r3 : add mcr, ax
ret

read proc next

mov ax, 01h

int 21h

xor ax, ax

sub ax, 30h

ret

end

To read system time .

- model small
- code

mov ah, 2ch

int 21h

mov al, ch

call

mov bx, ax

call disp

mov al, ':'

mov ah, 02h

int 21h

mov al, d

call

mov bx, ax

call disp

mov dl, ':'

mov ah, 02h

int 21h

mov al, dh

call

mov bx, ax

call disp

mov ah, 02h

int 21h

disb proc near
mov dl, bl

add dl, 30h

mov ah, 02h

int 21h

mov dl, bl

add al, 30h

mov ah, 02h

int 21h

ret

~~end~~ disb endp
end

Microsoft (R) Overlay Linker Version 3.60

Copyright (C) Microsoft Corp 1983-1987. All rights reserved.

LINK : warning L4021: no stack segment

C:\MASM>clock

15:33:37.737

BCD Counter

model small

code

mov cl, 00h

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

cmp

add ax, 3030h

mov ch, al

mov dl, ah

mov ah, 02h

int 21h

mov dl, ch

mov ah, 02h

int 21h

all delay

inc d
xor ax, ax
cmp cl, 100d
jne back
je last

delay proc near

push ax

push bx

push cx

mov cx, 00ffh

ag: mov bx, 0fffh

agi: mov

dec bx

jnz agi

dec cx

jnz ag

pop cx

pop bx

pop ax

ret

delay endp

last: mov ah, 4ch

int 21h

end



DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: BCDCOUNT



06_


```

.model small
disb macro msg
    lea dx, msg
    mov ah, 09h
    int 21h

```

; cursor program.

```
endm
```

```
.data
```

```
row db 02 dup(0)
```

```
col db 02 dup(0)
```

```
msg1 db 0dh, 0ah, "Enter X-coordinate: "
```

```
msg2 db 0dh, 0ah, "Enter Y-coordinate: "
```

```
msg3 db 0dh, 0ah, "cursor displayed at correct coordinates "
```

```
.code
```

```
mov ax, @data
```

```
mov ds, ax
```

```
disb msg1
```

```
mov si, offset row
```

```
call read
```

```
disb msg2
```

```
mov si, offset col
```

```
call read
```

```
mov si, offset row
```

```
mov ah, [si]
```

```
inc si
```

```
mov al, [si]
```

```
sub ax, 3030h
```

```
add
```


mov dh, al
mov si, offset cl
mov ah, [si]

inc si

mov al, [si]

~~add~~ add

mov dl, al

mov ah, 00

mov al, 02h

int 10h

mov ah, 02h

int 10h

disb msg3

jmp final

read proc near

mov cx, 02h

back: mov ah, 01h

int 21h

mov [si], al

inc si

dec cx

jmp back

ret


read endp

final: mov ah, 01h

int 21h

more ah, ~~by~~eh

int 21h

end 



DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: CURSOR



File program:

model small

display macro msg

lea dx, msg

mov ah, 09h

int 21h

endm

- data

msg1 db 0dh, 0ah, "Enter the filename for creation: \$"

msg2 db 0dh, 0ah, "File created successfully: \$"

msg3 db 0dh, 0ah, "File creation unsuccessful: \$"

msg4 db 0dh, 0ah, "Enter the filename for deletion: \$"

msg5 db 0dh, 0ah, "File Deleted successfully: \$"

msg6 db 0dh, 0ah, "File deletion failed: \$"

frame1 db 10 dup(0)

frame2 db 10 dup(0)

- code

mov ax, @data

mov ds, ax

display msg1

mov si, 00


```
back1: mov al, 01h  
       int 21h  
       cmp al, 0dh  
       je next1  
       mov frame1[si], al  
       inc si  
       jmp back1
```

```
next1: mov frame1[si], '$'  
       lea dx, frame1  
       mov cx, 00  
       mov ah, 3dh  
       int 21h  
       jc cfail  
       display msg2  
       jmp del
```

```
cfail: display msg3
```

```
del: display msg4  
     mov si, 00
```

```
back2: mov ah, 01h  
       int 21h  
       cmp al, 0dh  
       je next2  
       mov frame2[si], al  
       inc si  
       jmp back2
```


next 2 : mov frame 2 [si], '\$'

lea dx, frame 2

mov ah, 4ch

int 21h

jc dfail

display msg 5

jump final

dfail : display msg 6

final : mov oh, 4ch

int 21h

end

Enter the filename for creation: gk

File Created Successfully:

Enter the filename for deletion: gk

File Deleted Successfully:

C:\MASM>

; Program :: Reverse a given string and check whether it
; is a palindrome or not.

. 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 a palindrome. \$"

msg4 db 0dh, 0ah, "Input string is not a palindrome string. \$"

string db 80h dup(?)

rstring db 80h dup(?)

. code

start: mov ax, @data
mov ds, ax
display msg1

; Take the string from keyboard character by character.

mov si, offset string

~~mov~~ xor cl, cl

gain: mov ah, 01

int 21h

cmp al, 0dh

je next

mov [si], al

inc si

inc cl

0 = 0 zero -

0 = 0 alhabet.

jmp again

next : mov [si], byte ptr '\$'
; string input over

dec si

mov ch, cl

; Reverse the string and store in Rstring

mov di, offset rstring

back : mov al, [si]

mov [di], al

dec si

inc di

dec ch

jnz back

mov [di], byte ptr '\$'

display msg2

display rstring

mov si, offset rstring

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 msg3

jmp final

success: display msg3

final: mov ah, 4ch
int 21h

end


```
C:\MASM>masm pal;;  
Microsoft (R) Macro Assembler Version 5.00  
Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.
```

```
51680 + 464864 Bytes symbol space free
```

```
0 Warning Errors  
0 Severe Errors
```

```
C:\MASM>link pal;;
```

```
Microsoft (R) Overlay Linker Version 3.60  
Copyright (C) Microsoft Corp 1983-1987. All rights reserved.
```

```
LINK : warning L4021: no stack segment
```

```
C:\MASM>pal
```

```
Enter the String: madam
```

```
Reverse String : madam  
Input String is palindrome.
```

```
C:\MASM>_
```