Tasm

Ascending order of 10 no. in array

Code:

```
data segment
num dw - 0AAAAH , 0FFFFH, 6666H, 5555H, 4444H,
  3333H, 1234H, 1111H, 0BBBBH, 0999H
Count dw 0009H.
Code segment
assume cs:code, ds:data
start: mov ax, data
       mov ds, ax
       mov dx, count
back1: lea si, num.        (Load effective addr)
       mov cx, dx
back2: mov ax, [si]
       cmp ax, [si+2]
       JC next                    JNC (Descending)
       xchg [si+2], ax
next:  mov [si], ax
       inc si
       inc si
       loop back2
       dec dx
       JNZ back1
       int 03h
code ends
end start
```

Result :- 1111, 1234, 3333, 4444, 5555, 6666, 0AAAAH, 0BBBBH, 0FFFFH.

*NOTE Eg: SI = 0500 then SI+2 = 0502.

mov [si], ax mean storing ax in 0500 as SI here is 0500

WAP 2 8-bit Add Sub mul div
" 2 16-bit " " " "
WAP 32 x 32 bit multi
64 ÷ 16 bit div

Q WAP to add two 8-bit no.

```
MOV   AL, 09H
MOV   BL, 05H
ADD   AL, BL
END.
```

Q write a display routine to display above addn ans.

```
mov ch 02H.
mov cL, 04H.
mov bh, al.
L2 rol bh, cl
mov dl, bh.
AND dL, 0FH
cmp dl, 09.
JBE L1; dL, 07
L1: ADD dl, 30H.
mov ah, 02.
int 21H.
dec ch
JNZ L2
mov ah, 4ch.
int 21H
end.
```

Q. Subtract two 8-bit no.

```
MOV AL, 0AH
MOV BL, 04H.
SUB AL, BL
END.
```

0A - 04 = 06H

Q. Multiply two 8-bit no.

```
MOV AL, 09H
MOV BL, 02H
MUL BL
END
```

(18)d = (12)H.
AX = 0012 H.

Q. Add 8-bit BCD no.

```
MOV AL, 09H
MOV BL, 02H.
ADD AL, BL
DAA
MOV BH, AL
END
```

9+2 = 0B
0B+06 = 11 H. → A
BH = 11H.

Q. Sub 8-bit BCD no.

```
MOV AL, 32H
MOV BL, 17H.
SUB AL, BL
DAS
MOV BH, AL
END
```

AL = 32-17 = 1BH.
AL = 1B-06 = 15H
BH = 15H.

Q Divide 16-bit numbers by an 8-bit no → word in AX, byte in BL WAP to AX ÷ BL

```
MOV AX, 000FH
MOV BL, 08H
DIV BL
MOV DX, AX
END
```

```
      1
08) 000F
   - 08
    07
```

(R) AH = 07H, (Q) AL = 01H

Q. Divide 16-bit signed no. by a 16-bit signed no. -AX ÷ -BX.

```
MOV AX, 8000H
MOV BX, 2000H
IDIV BX
```

AX = 8000 ÷ 2000 = 0004H

AH = 00H (R), AL = 04H (Q).

Q ADD / SUB two 16-bit no.

```
MOV AX, 1234H
MOV BX, 0100H
ADD AX, BX / SUB AX, BX
MOV BX, AX
END
```

ADD

AX = 1334H

SUB

```
AX = 1234
   - 0100
   1134H
```

Q To find Factorial of a Number using Recursive procedure.

$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$

$n! = n \times (n-1) \times (n-2) \times \cdots 2 \times 1$

```
MOV   AX, 01        Initialize AX with 01
MOV   BX, num       Load the num
CALL  FACT.
mov   di, AX        store lsb in di
mov   BX, DX        " msb in bx
```

```
FACT_PROC NEAR.          // factorial procedure.
    cmp bx, 01           if bx = 1
    JZ  L1               yes ZF=1
L2: MUL bx               AX=BX * AX = 01
    dec bx
    cmp bx, 01.          bx ≠ 1  ZF=0
    JNE L2
    ret
L1  mov AX, 01.
    ret
    fact endp
    endp.
```

w/o proc

```
MOV AX, 01
MOV CL, 05
L2: CMP CX, 01
    JZ  L1
    MUL CX
    DEC CX
    JNZ L2
L1: MOV 4000, AX
    END
```

Fibonacci Series.

Eg:- Fibonacci of 10 number (ie N=10)

Soln = 0+0=0, 0+1=1, hence = 0, 1, 1, 2, 3, 5, 8, 13, 21, 34

1+1 = 2
1+2 = 3
2+3 = 5
3+5 = 8
5+8 = 13
8+13 = 21
13+21 = 34
21+34 = 55
34+55 = 89

hexa => = 0, 1, 1, 2, 3, 5, 8, 0D, 15, 22.

```
MOV   CX, 000A      Initialize Counter
MOV   SI, offset res
MOV   AL, 00H       Initial AL=0 ie 1st term of series
MOV   BL, 01H       " BL=1 ie next " of series.
MOV [SI], al
inc SI
mov [SI], bl
inc SI
L1: mov al, [SI-2]
mov bl, [SI-1]
ADD al, bl
mov [SI], al
inc SI
loop L1
mov si, offset res
MOV bp, 10
mov bl, [SI]
HLT END
```

AL = 00
BL = 01

0+1 = 1
1+1 = 2
1+2 = 3

0000 - 00
0001 - 01
0002 - 01
0003 - 02
0004 -

Q. ~~Design~~ write 8086 ALP to print the flag register.

```
start
MOV ax, data
MOV DS, ax.
mov DS, offset msg
mov ah, 09H
int 21 h.
pushf  decrements SP-2 & copies word from
pop bx  copies word from stack to AX reg.
mov flag, bx  2 successive mL contents
mov CX, 16
mov bx, 8000H
L1: mov ax, flag
AND ax, bx
JZ L2.
mov dl, 31H
mov ah, 02H
int 21H
jmp L3.
L2: MOV dl, 30H.
mov ah, 02H
int 21 h.
mov dl,
mov ah, 02H
int 21 H
mov ah, 02H
int 21H
ror bx, 1
Loop L1
mov ah, 4CH        Code ends
int 21H            end start
```

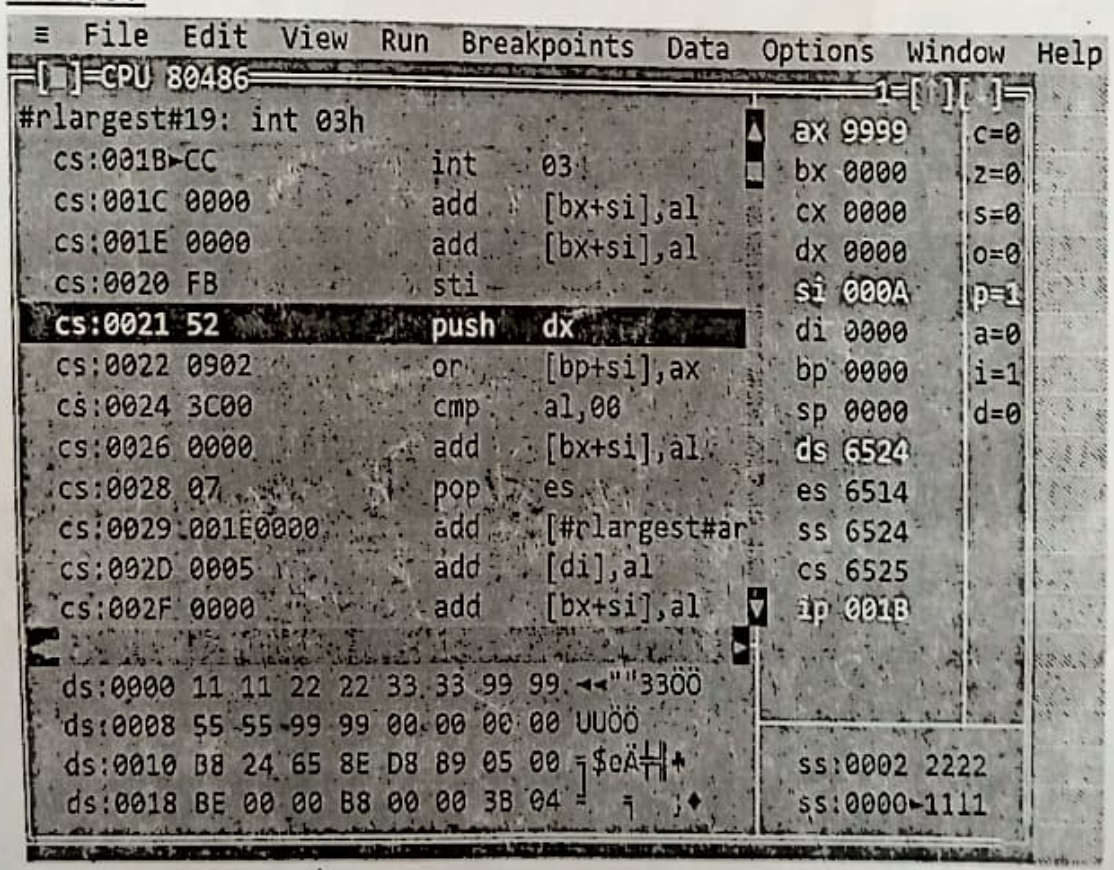
Aim:- WAP to find largest number in an array.

RLARGEST.ASM

```
data segment
array dw 1111h,2222h,3333h,9999h,5555h
largest dw 1 dup (0)
data ends
code segment
assume cs:code,ds:data
start:mov ax,data
mov ds,ax
mov cx,0005h
lea si,array
mov ax,0000h
next:cmp ax,[si]
jnc skip
mov ax,[si]
skip:inc si
inc si
loop next
mov largest,ax
int 03h
code ends
end start
```

21:14

OUTPUT :-



5. a. Reverse a given string and check whether it is a palindrome or not.

```
.model small
.data
        str1    db "alam"
        slen    equ ($-str1)
        str2    db 40 dup(0)
        msg1    db "Palindrome$"
        msg2    db "Not Palindrome$"
.code
start:  mov     ax,@data
        mov     ds,ax
        mov     es,ax

        mov     cx,slen
        mov     si,offset str1
        add     si,slen - 1
        mov     di,offset str2
up:     mov     al,[si]
        mov     [di],al
        dec     si
        inc     di
        loop    up

        mov     si,offset str1
        mov     di,offset str2
        mov     cx,slen
        cld
repe    cmpsb
        jne     down

        mov     dx,offset msg1
        jmp     down1
down:   mov     dx,offset msg2
down1:  mov     ah,09h
        int     21h
        int     3
        end     start
```

Conclusion:

This program reverse the string provided in data segment by keeping the original string as it is and compares both the strings. It will check each and every character. If all the characters are same then the given string is said to be as palindrome and it will display a message "**palindrome**" on screen otherwise the given string is not palindrome and it will display a message "**not palindrome**" on screen.

KIT

Largest / Greatest

ADDR	Opcode	Mnemonic	Opr	Comment
0400	BE	MOV SI, 0500		Initialize SI.
01	00			
02	05			
0403	B9	MOV CX, 0010		" Counter
0404	10			
0405	00			
0406	B4	MOV AH, 00		" AH.
0407	00			
L2: 0408	3A	CMP AH, [SI]		AH - [SI] → if SI>AH CF ZF SF = 1 0 1; SI<AH = 0 0 0; SI=AH = 0 1 0
0409	24			
040A	73	JNB 040E	L1	if SI is smaller CF=0 takes jump as AH is not below SI; if SI greater CF=1 no jump / JNC 040E CF=0 takes jump
040B	02			
040C	8A	MOV AH, [SI]		greater value in AH.
040D	24			
L1: 040E	46	INC SI		
040F	E0	LOOPNE 0408	L2	Dec CX ZF=0. if ZF=1 next inst
0410	F7			
0411	88	MOV [SI], AH		
0412	24			
0413	F4	HLT.		

opcode for SI only

Unsigned Jmp Opcn no SF is checked.

JNB → if not below, CF=0. takes jump

Loopne → loop if not equal.

CX = CX - 1

if CX <> 0 & ZF = 0 then jump to

else no jump.