**# Greatest Among 3 number check**

| .MODEL SMALL    .STACK 100H  .DATA  msg db "Enter one number: $"  ans db "You entered $"  abig db "a big$"  bbig db "b big$"  cbig db "c big$"  a db ?  b db ?  c db ?  sum db ?  .CODE  MAIN PROC  ;iniitialize DS  MOV AX,@DATA  MOV DS,AX  ;enter your code here  mov ah, 9  lea bx, msg  mov dx, bx  int 21h    mov ah, 1  int 21h  mov a, al    mov ah, 2  mov dl, 10  int 21h  mov dl, 13  int 21h    mov ah, 9  lea bx, msg  mov dx, bx  int 21h  mov ah, 1  int 21h  mov b, al    mov ah, 2  mov dl, 10  int 21h  mov dl, 13  int 21h  mov ah, 9  lea bx, msg  mov dx, bx  int 21h  mov ah, 1  int 21h  mov c, al    mov ah, a  mov al, b  mov bh, c    cmp ah, al  jge ac    cmp al, bh  jge bc  ac:  cmp ah, bh  jge printa  jmp printc  bc:  cmp al, bh  jge printb  jmp printc    printa:  mov ah, 2  mov dl, 10  int 21h  mov dl, 13  int 21h  mov ah, 9  lea bx, abig  mov dx, bx  int 21h  jmp exit  printb:  mov ah, 2  mov dl, 10  int 21h  mov dl, 13  int 21h  mov ah, 9  lea bx, bbig  mov dx, bx  int 21h  jmp exit  printc:  mov ah, 2  mov dl, 10  int 21h  mov dl, 13  int 21h  mov ah, 9  lea bx, cbig  mov dx, bx  int 21h  jmp exit  exit:  ;exit to DOS  lesscheck:  mov ah, a  mov al, b  mov bh, c  cmp ah, al  jle acless  cmp al, bh  jle pbless    pcless:  mov ah, 2  mov dl, 10  int 21h  mov dl, 13  int 21h  mov ah, 2  mov dx, "c"  int 21h  jmp finally  acless:  cmp ah, bh  jle paless  jmp pcless  paless:  mov ah, 2  mov dl, 10  int 21h  mov dl, 13  int 21h  mov ah, 2  mov dx, "a"  int 21h  jmp finally  pbless:  mov ah, 2  mov dl, 10  int 21h  mov dl, 13  int 21h  mov ah, 2  mov dx, "b"  int 21h  jmp finally        finally:  MOV AX,4C00H  INT 21H  MAIN ENDP  END MAIN |
| --- |

**# SUM OF TWO NUMBER**

| .MODEL SMALL   .STACK 100H  .DATA a db ? b db ?  r db ? msg1 db "?:$" msg2 db 10,13, "THE SUM OF$" msg3 db "and $" msg4 db "is $"  .CODE MAIN PROC  ;iniitialize DS  **MOV** AX,@DATA **MOV** DS,AX   ; enter your code here   **mov** ah, 9  **mov** dx, offset msg1   **int** 21h    **mov** ah, 1  **int** 21h   **sub** al, 48  **mov** a, al    **mov** ah, 1  **int** 21h  **sub** al, 48   **mov** b, al     **mov** ah, a   **mov** al, b  **add** ah, al  **mov** r, ah    **mov** ah, 9   **mov** dx, offset msg2  **int** 21h     **mov** ah, 2   **mov** dl, a   **add** dl, 48  **int** 21h     **mov** ah, 9   **mov** dl, offset msg3  **int** 21h     **mov** ah, 2  **mov** dl, b   **add** dl, 48  **int** 21h     **mov** ah, 9   **mov** dl, offset msg4  **int** 21h     **mov** ah, 2  **mov** dl, r   **add** dl, 48  **int** 21h                    ;exit to DOS   **MOV** AX,4C00H **INT** 21H  MAIN ENDP  END MAIN |
| --- |

**# COMPARE 3 NUMBERS**

| .MODEL SMALL   .STACK 100H  .DATA a db ? b db ?  r db ? msg1 db "?:$" msg2 db 10,13, "A is bigger than 5$" msg3 db 10,13, "A is less than 5$" msg4 db 10,13, "a is $"  .CODE MAIN PROC  ;iniitialize DS  **MOV** AX,@DATA **MOV** DS,AX   ; enter your code here  **mov** ah, 9  **mov** dl, offset msg1   **int** 21h     **mov** ah, 1  **int** 21h  **sub** al, 48  **mov** a, al     **cmp** a, 5  **jge** big  **mov** a, 5   **mov** ah, 9   **mov** dl, offset msg3  **int** 21h    **mov** ah, 9   **mov** dl, offset msg4  **int** 21h     **mov** ah, 2  **mov** dl, a  **add** dl, 48   **int** 21h  **jmp** exit    big:   **mov** ah, 9   **mov** dl, offset msg2   **int** 21h   **jmp** exit       exit:   ;exit to DOS   **MOV** AX,4C00H **INT** 21H  MAIN ENDP  END MAIN |
| --- |

**# REVERSE STRING**

| ## REVERSE STRING .MODEL SMALL .STACK 100H .DATA l db ? b db ?  r db ? msg1 db "Enter Length:$" **str** db ?  ; TANJIM    .CODE MAIN PROC  ;iniitialize DS  **MOV** AX,@DATA **MOV** DS,AX   ; enter your code here    **mov** ah, 9   **mov** dl, offset msg1   **int** 21h     **mov** ah, 1  **int** 21h   **sub** al, 48  **add** al, 1  **mov** l, al    **mov** ah, 2   **mov** dx, 10   **int** 21h   **mov** ah, 2  **mov** dx, 13  **int** 21h  ; loop     **mov** cl, l   **mov** si, 0   loopstart:   **mov** ah, 1  **int** 21h   **mov** **str**[si], al  **inc** si   **dec** cl  **jnz** loopstart  output:   **mov** cl,l  **mov** ah,2  **mov** si,5  start:  **mov** dl, **str**[si]  **int** 21h  **sub** si, 1  **dec** cl  **jnz** start ;  ; mov cx, l ; mov si, 0 ; 5 ; mov bx, l ; show: ; mov ah, 2 ; mov si, dx ; mov dl, str[si]  ; int 21h  ; dec bx  ; cmp cl, 0  ; jle exit ; loop show ;  ;  ; mov si, 3 ; mov ah, 2 ; mov dl, str[si] ; int 21h ; exit:  ;exit to DOS   **MOV** AX,4C00H **INT** 21H  MAIN ENDP  END MAIN |
| --- |

**# NUMBER ODD/EVEN**

| .MODEL SMALL   .STACK 100H  .DATA a dw ? b db ?  msg1 db "Enter Number: $" evenmsg db 10,13, "Number is Even$" oddmsg db 10,13, "Number is Odd$"  .CODE MAIN PROC  ;iniitialize DS  **MOV** AX,@DATA **MOV** DS,AX   ; enter your code here   **mov** ah, 9   **mov** dx, offset msg1  **int** 21h     **mov** ah, 1  **int** 21h  **mov** a, ax   ; CHECK   **mov** ax, a  **mov** bl , 2     **div** bl ; rem -> ah, res -> al    **cmp** ah, 0   **jz** printeven  **mov** ah, 9   **mov** dx, oddmsg offset  **int** 21h   **jmp** exit:    printeven:  **mov** ah, 9   **mov** dx, evenmsg offset  **int** 21h   **jmp** exit  exit:   ;exit to DOS   **MOV** AX,4C00H **INT** 21H  MAIN ENDP  END MAIN |
| --- |

**# REVERSE STRING-2**

| .MODEL SMALL   .STACK 100H  .DATA   a db ?   .CODE  MAIN PROC   ;initialize DS  **MOV** AX,@DATA  **MOV** DS,AX    ; enter your code here   **mov** cx,0  **mov** ah,1 **int** 21h **mov** cl,al **sub** cl,30h  **mov** si,0  input: **int** 21h  **mov** a[si],al **inc** si  **loop** input  **mov** ah,2  **dec** si  print: **mov** dl,a[si] **sub** dl,20h **dec** si **int** 21h **cmp** si,0 **jl** exit **jmp** print  exit:      ;exit to DOS    **MOV** AX,4C00H **INT** 21H   MAIN ENDP  END MAIN |
| --- |

**#PALINDROME CHECK**

| .model small .data .stack 100h   ; add your data here!  s db "input length: $"  yes db " Palindrome$"  no db " Not a Palindrome $"  a db 100 dup(?)  p db " input string:$"  .code main proc   **mov** ax,@data  **mov** ds, ax    **mov** ah,9  **lea** dx,s  **int** 21h    **mov** ah,1  **int** 21h  **mov** bl,al  **sub** bl,30h  **and** bh,0    **mov** ah,9  **lea** dx,p  **int** 21h    **mov** cx,bx  **mov** si, 0  **mov** ah,1    input:  **int** 21h  **mov** dl,al  **mov** a[si],dl  **inc** si  **loop** input     **mov** cx,bx  **mov** si, 0    start:  **mov** dl, a[si]  **mov** dh, 0  **push** dx  **inc** si  **loop** start   **mov** cx,bx  **mov** si, 0    check:  **pop** dx   **mov** dh, 0  **cmp** a[si], dl  **jne** nomatch  **inc** si  **loop** check   **mov** ah, 9  **lea** dx, yes  **int** 21h  **jmp** exit    nomatch:  **mov** ah, 9  **lea** dx, no  **int** 21h   ;exit to DOS exit:  **MOV** AX,4C00H **INT** 21H  MAIN ENDP  END MAIN |
| --- |

**#FIBONACCI UPTO CX TIMES FROM 0**

| .MODEL SMALL .STACK 100H .DATA  t db 0    .CODE  MAIN PROC   ;initialize DS  **MOV** AX,@DATA  **MOV** DS,AX    ; enter your code here    ; first two   ; a - bh  ; b - bl   **mov** bh, 0  **mov** bl, 1  **mov** cx, 7  fib:  **mov** t, bl  **add** bl, bh  **mov** ah, 2  **mov** dl, bh   **add** dl, 48  **int** 21h  **mov** bh, t    **cmp** cl, 0   **jz** exit  **loop** fib   ;int 21h    exit: ;exit to DOS    **MOV** AX,4C00H **INT** 21H   MAIN ENDP  END MAIN |
| --- |

#REVERSE SORT + PRINT

| .MODEL SMALL .STACK 100H .DATA A DB 7,3,1,2,5  B DB ? .CODE MAIN PROC  ;iniitialize DS  **MOV** AX,@DATA **MOV** DS,AX   ; enter your code here **mov** dx,0  FIRST: **cmp** dx,5 ; LOOP LIMIT 5 **jge** SHOWRESULT ; IF DX >= 5 then SHOW OUTPUT   **mov** cx,0 ; STARTING WIH 0  SECOND:  **cmp** cx,dx ; CHECK CX & DX  **jg** TEMP ; IF Greater then dl+1 and FIRST LOOP   **mov** si,dx ; Else, get DX into SI     **mov** bh,a[si] ; Current i of ARRAY = A[i]    **mov** si,cx ; FOR THE NEXT ITEM     **mov** bl,a[si] ; Current next(cx) of ARRAY = A[j]      **cmp** bh,bl ; CHECK IF BH <= BL ; Then A[i] <= A[j], Need to Change Position  **jle** CHANGEPOS ; Go to Change Position   **inc** cx ; Increase CX to check with the next-next item ++  **jmp** SECOND ; REPEAT  TEMP:  **inc** dl  **jmp** FIRST  CHANGEPOS:  **mov** si,dx ; DX = Current  **mov** a[si],bl ; BL = Value Store    **mov** si,cx ; CX = Next One  **mov** a[si],bh ; BH = Prev Value Store  **inc** cx  **jmp** SECOND   SHOWRESULT:  **mov** cx,5 ; RESET  **mov** ah,2  **mov** si,0  PRINTOUTPUT:  **mov** dl,a[si] ; PRINT OUTPUT   **add** dl,30h  **int** 21h     ; mov AX, B ; add DX, 30h ; int 21h    **add** si,1  **loop** PRINTOUTPUT   **mov** cl, 5 **mov** si, 4 output:  **mov** ah, 2  **mov** dl, a[si]  **add** dl, 30h  **int** 21h     **sub** si, 1  **loop** output  ;exit to DOS exit:  **MOV** AX,4C00H **INT** 21H  MAIN ENDP  END MAIN |
| --- |

**#max between 3 num**

| .model small  macro max n1,n2,n3  mov bx,n1  mov cx,n2  mov dx,n3  cmp bx,cx  jg n1gn2  jmp n2gn1  n2gn1:  cmp cx,dx  jg n2g  jmp n3g  n1gn2:  cmp bx,dx  jg n1g  jmp n3g  n1g:  mov ax,n1  jmp max  n3g:  mov ax,n3  jmp max  n2g:  mov ax,n2  jmp max  max:  mov dx,ax  add dx,30h  mov ah,2  int 21h  endm  .data  A dw 9  B dw 5  C dw 7  .stack 3000h  .code  mov ax, @data  mov ds, ax  mov es, ax  max A,B,C  exit:  MOV AX,4C00H  INT 21H |
| --- |

**#bin to dec**

| .MODEL SMALL  .STACK 100H  .DATA  arr db "1234"  len db 4  p db 0  s dw 0  .CODE  MAIN PROC  ;iniitialize DS  MOV AX,@DATA  MOV DS,AX  mov es,ax  ; enter your code here  mov bx,0  mov ah,1  for:  int 21h  cmp al,0dh  je end  sub al,30h  shl bx,1  or bl,al  jmp for  end:  mov ah,2  mov dl,bl  add dl,30h  int 21h  ;exit to DOS  exit:  MOV AX,4C00H  INT 21H  MAIN ENDP  END MAIN |
| --- |

**#convert 4 digit string to num**

| **.MODEL SMALL**  **macro power b,p**  **cmp p,0**  **je edge:**  **mov al,1**  **mov bl,b**  **mov bh,0**  **loop:**  **cmp bh,p**  **je multi**  **mul bl**  **inc bh**  **jmp loop**  **edge:**  **mov ax,1**  **jmp multi**  **endm**  **.STACK 100H**  **.DATA**  **arr db "1234"**  **len db 4**  **p db 0**  **s dw 0**  **.CODE**  **MAIN PROC**  **;iniitialize DS**  **MOV AX,@DATA**  **MOV DS,AX**  **mov es,ax**  **; enter your code here**  **and dx,0**  **and cx,0**  **and bx,0**  **mov cl,len**  **sub cl,1**  **mov si,0**  **start:**  **mov bl,10**  **and ax,0**  **cmp cl,-1**  **je exit**  **mov bh,arr[si]**  **sub bh,30h**  **mov p,bh**  **power bl,cl**  **multi:**  **mov bl,p**  **mov bh,0**  **mul bx**  **mov dx,s**  **add dx,ax**  **mov s,dx**  **inc si**  **dec cl**  **jmp start**  **;exit to DOS**  **exit:**  **MOV AX,4C00H**  **INT 21H**  **MAIN ENDP**  **END MAIN** |
| --- |

**#(x^n)**

| .MODEL SMALL  macro power b,p  cmp p,0  je edge:  mov al,1  mov bl,b  mov bh,0  loop:  cmp bh,p  je exit  mul bl  inc bh  jmp loop  edge:  mov ax,1  jmp exit  endm  .STACK 100H  .DATA  bin db "111$"  ARR DB 50 DUP(?)  .CODE  MAIN PROC  ;iniitialize DS  MOV AX,@DATA  MOV DS,AX  mov es,ax  ; enter your code here  and dx,0  and cx,0  and ax,0  and bx,0  ;;  mov bl,2  mov cl,3  power bl,cl  ;exit to DOS  exit:  mov dx,ax  add dx,30h  mov ah,2  int 21h  MOV AX,4C00H  INT 21H  MAIN ENDP  END MAIN |
| --- |

**#parenthesis match**

| .model small  macro parenthesis str  mov dx,"@"  push dx  and dx,0  mov si,0  loop:  mov bl,str[si]  cmp bl,"\"  je check  cmp bl,"("  je append  cmp bl,"{"  je append  cmp bl,"["  je append  cmp bl,")"  je see(  cmp bl,"}"  je see{  cmp bl,"]"  je seet  inc si  jmp loop  append:  mov dl,bl  and dh,0  push dx  inc si  jmp loop  see(:  pop dx  cmp dl,"("  jne nomatch  inc si  jmp loop  see{:  pop dx  cmp dl,"{"  jne nomatch  inc si  jmp loop  seet:  pop dx  cmp dl,"["  jne nomatch  inc si  jmp loop  check:  pop dx  cmp dl,"@"  je match  jne nomatch  match:  lea dx,right  mov ah,9  int 21h  jmp exit  nomatch:  lea dx,wrong  mov ah,9  int 21h  jmp exit    endm  .data    wrong db "no match$"  right db "match$"    ARR DB "10+(3+5)}\$"  .stack 300h  .code  mov ax, @data  mov ds, ax  mov es, ax  parenthesis arr  exit:  MOV AX,4C00H  INT 21H |
| --- |

**#factorial**

.model small

factorial macro var1

mov cx, var1

mov ax, 1

fact:

mov bl,cx

mul bl

loop fact

endm

.stack 100h

.data

a db 5

b db 5 dup(?)

.code

main proc

mov ax, @data

mov ds, ax

factorial a

**#Neg-positive identify :**

INCLUDE 'EMU8086.INC '

.Model Small

.stack 100h

.data

.code

main proc

mov ax,@data

mov ds,ax

;asking for an input

PRINT "PLEASE ENTER A NUMBER "

MOV AH,1

INT 21H

PRINTN

MOV BX, AX ;RANDOM NUMBER

CMP BX,0

JL IF

JGE ELSE

IF:

PRINTN "YOUR NUMBER IS NEGETIVE"

JMP END

ELSE:

PRINTN "YOUR NUMBER IS POSITIVE"

JMP END

;

;showing output with message in new line

;showing output in same line two output

END:

mov ah,4ch

int 21h

main endp

end main

**#Array sorting :**

include 'emu8086.inc'

.model small

.data

arr db 5 dup(?)

.code

main proc

mov ax,@data

mov ds,ax

print "Enter 5 Number in Array:"

mov cx, 5

mov bx,offset arr

mov ah, 1

inputs:

int 21h

mov [bx], al

inc bx

loop inputs

mov cx, 5

dec cx

OuterLoop:

mov bx, cx

mov si, 0

CompLoop:

mov al,arr[si]

mov dl,arr[si-1], al

cmp al,dl

jc noSwap

mov arr[si], dl

mov arr[si+1], al

noSwap:

inc si

dec bx

jnz CompLoop

loop OuterLoop

mov ah,2

mov dl,10

int 21h

mov dl, 13

int 21h

print "After Sorting Array:"

mov cx,5

mov bx, offset arr

Outputs:

mov dl,[bx]

mov ah, 2

int 21h

mov dl,32

mov ah, 2

int 21h

inc bx

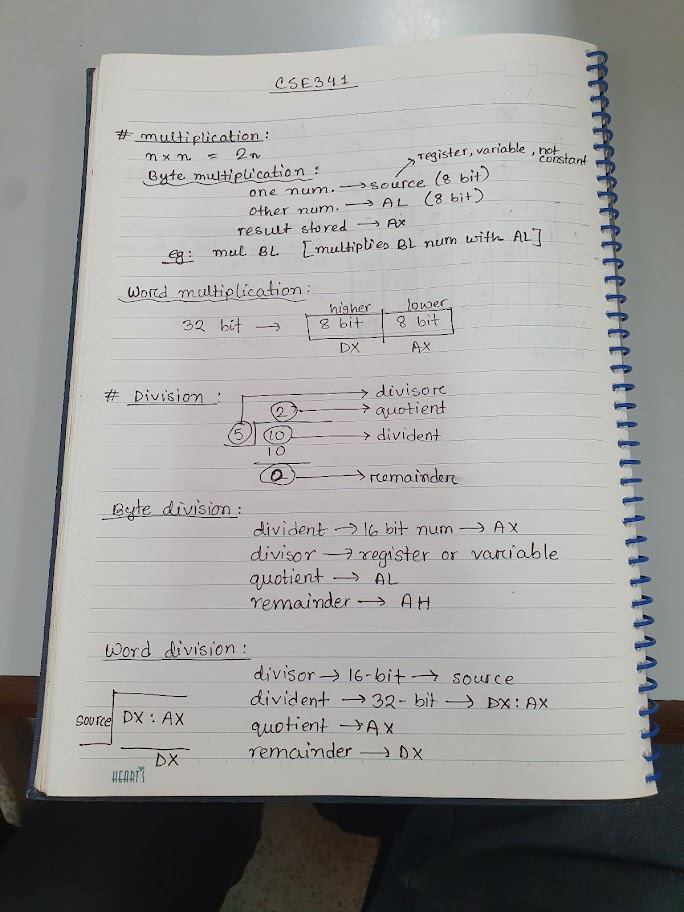
loop Outputs

main endp

ret

**#Leap year :**

| include 'emu8086.inc' |  |
| --- | --- |
|  |  |
|  | mov cx, 04h |
|  |  |
|  |  |
|  | inp: mov ah, 01h |
|  | int 21h |
|  |  |
|  | sub al, 30h |
|  | mov bl, al |
|  | mov ax, dx |
|  | mov dx, 10h |
|  | mul dx |
|  |  |
|  | mov bh, 00h |
|  | add ax, bx |
|  | mov dx, ax |
|  | loop inp |
|  |  |
|  |  |
|  | mov ax, dx |
|  | mov dx, 0000h |
|  | mov bx, 64h |
|  | mul bx |
|  | mov bx, 100h |
|  | div bx |
|  | mov cx, ax |
|  | mov dx, 0000h |
|  |  |
|  | mov bx, 400h |
|  | div bx |
|  | jz yes |
|  |  |
|  | mov ax, cx |
|  | mov bx, 04h |
|  | mov dx, 0000h |
|  | div bx |
|  | jnz no |
|  |  |
|  | mov ax, cx |
|  | mov dx, 0000h |
|  | mov bx, 100h |
|  | div bx |
|  | jnz no |
|  | jz yes |
|  |  |
|  | no: |
|  | print 'Not a Leap Year' |
|  | jmp exit |
|  |  |
|  | yes: |
|  | print 'Leap Year' |
|  |  |
|  | exit: |
|  |  |
|  | ret |



name "hex-bin"

org 100h

; load binary value:

; (hex: 5h)

mov al, 00000101b

; load hex value:

mov bl, 0ah

; load octal value:

; (hex: 8h)

mov cl, 10o

; 5 + 10 = 15 (0fh)

add al, bl

; 15 - 8 = 7

sub al, cl

; print result in binary:

mov bl, al

mov cx, 8

print: mov ah, 2 ; print function.

mov dl, '0'

test bl, 10000000b ; test first bit.

jz zero

mov dl, '1'

zero: int 21h

shl bl, 1

loop print

; print binary suffix:

mov dl, 'b'

int 21h

; wait for any key press:

mov ah, 0

int 16h

ret