PROYECTO FINAL ARQUITECTURA
CODIGO EMU8086
GRUPO #10

```
;---- Integrantes
; Diana Victores  9959-19-1471
; Josue Amaya     0901-19-12421
; Jonathan Xuya   0901-18-11371

include 'emu8086.inc'

org 100h

call ciclo

ret

   ciclo:
   call menuP
   call opcion

   cmp al,35h
    jg ciclo

   cmp al,30h
   je ciclo

      ret

   menuP PROC

   call clear

   ;Ubicar posiciones especificas en pantalla
   mov dh,10   ;fila
   mov dl,25  ;columna
   mov ah,2
   int 10h
   ;mensaje 1
   mov dx, offset cadena1
   mov ah,9
   int 21h

   ;Ubicar posiciones especificas en pantalla
   mov dh,11   ;fila
   mov dl,30  ;columna
   mov ah,2
   int 10h

   ;mensaje 2
   mov dx, offset cadena2
   mov ah,9
   int 21h

   ;mensaje 3
   mov dx, offset cadena3
   mov ah,9
   int 21h

   ;mensaje 4
   mov dx, offset cadena4
   mov ah,9
   int 21h

   ;mensaje 5
   mov dx, offset cadena5
   mov ah,9
   int 21h

   ;mensaje 6
   mov dx, offset cadena6
   mov ah,9
   int 21h

   ;mensaje 7
   mov dx, offset cadena7
   mov ah,9
   int 21h

   ;mensaje 8
   mov dx, offset cadena8
   mov ah,9
   int 21h

   ;Recibir opcion
   mov ah, 1
   mov dx,20
   int 21h

   ret
```

PROYECTO FINAL ARQUITECTURA
CODIGO EMU8086
GRUPO #10

```
    menuP ENDP
    opcion PROC

    cmp al,31h;     Compara el dato ingresado con el numero 1 ascii
    je sub_menu1 ;  Salta al sub menu1

    cmp al,32h ;    Compara el dato ingresado con el numero 2 ascii
    je sub_menu2 ;  Salta al sub menu2

    cmp al,33h;     Compara el dato ingresado con el numero 3 ascii
    je sub_menu3;   Salta al sub menu3

    cmp al,34h;     Compara el dato ingresado con el numero 4 ascii
    je sub_menu4;   Salta al sub menu4

    cmp al,35h;     Compara el dato ingresado con el numero 5 ascii
    je sub_menu5;    Salta al sub menu4

    ret
    opcion ENDP

          ;--- submenu ---
    sub_menu1:

    call clear
    call position

    mov dx, offset subcadenat1 ; Titulo de la seleccion
    mov ah,9
    int 21h

    mov dx, offset subcadena1_1 ; mesaje submenu1_1
    mov ah,9
    int 21h

    mov dx, offset subcadena1_2 ; mesaje submenu1_2
    mov ah,9
    int 21h

    mov dx, offset subcadena1_3 ; mesaje submenu1_3
    mov ah,9
    int 21h

    ;-- lectura / Recibir opcion
    mov ah, 1
    mov dx,20
    int 21h

    ;-- llamada a submenu 1
    call opcion_submenu1
    ret

     ; --- procedimiento proc
    opcion_submenu1 PROC
        cmp al, 31h ;compara el dato ingresado con el numero 1 ascii
        je Suma1        ;salta al submenu1

        cmp al, 32h ;compara el dato ingresado con el numero 2 ascii
```

PROYECTO FINAL ARQUITECTURA
CODIGO EMU8086
GRUPO #10

```
        je Resta1       ;salta al submenu1

        cmp al, 33h     ;compara el dato ingresado con el numero 3 ascii
        je Multi1       ;salta al submenu1

        ret
    opcion_submenu1 ENDP


    sub_menu2:

    call clear

    call position

    mov dx, offset subcadenat2 ; Titulo de la seleccion
    mov ah,9
    int 21h

    mov dx, offset subcadena2_1 ; mesaje submenu2_1
    mov ah,9
    int 21h


     ;-- lectura / Recibir opcion
    mov ah, 1
    mov dx,20
    int 21h

    ;-- llamada a submenu2
    call opcion_submenu2
    ret

     ; --- procedimiento proc
    opcion_submenu2 PROC
        cmp al, 31h     ;compara el dato ingresado con el numero 1 ascii
        je palindroma   ;salta al submenu1

        ret
     opcion_submenu2 ENDP


    sub_menu3:
    call clear
    call position

    mov dx, offset subcadenat3 ; Titulo de la seleccion
    mov ah,9
    int 21h

    mov dx, offset subcadena3_1 ; mesaje submenu3_1
    mov ah,9
    int 21h

      ;-- lectura / Recibir opcion
    mov ah, 1
    mov dx,20
    int 21h

    ;-- llamada a submenu2
    call opcion_submenu3

    ret

    opcion_submenu3 PROC
        cmp al, 31h     ;compara el dato
```

PROYECTO FINAL ARQUITECTURA
CODIGO EMU8086
GRUPO #10

```
ingresado con el numero 1
ascii
        je Juego_Snake
;salta al juego snake

        ret
     opcion_submenu3
ENDP



    sub_menu4:

    call clear
    call position

    mov dx, offset
subcadenat4 ; Titulo de la
seleccion
    mov ah,9
    int 21h

    mov dx, offset
subcadena4_1 ; mesaje
submenu4_1
    mov ah,9
    int 21h

       ;-- lectura / Recibir
opcion
    mov ah, 1
    mov dx,20
    int 21h

    ;-- llamada a submenu4
    call opcion_submenu4
    ret

     ; --- procedimiento proc
    opcion_submenu4
PROC
        cmp al, 31h
;compara el dato
ingresado con el numero 1
ascii
        je Fibonacci
;salta al submenu1

            ret
       opcion_submenu4
ENDP



    sub_menu5:
    call clear
    call position

    mov dx, offset
subcadenat5 ; Titulo de la
seleccion
    mov ah,9
    int 21h
    ret


    clear:
    mov ah,00h ; limpiar
pantalla
    mov al,03h
    int 10h
    ret

    position:
    ;Ubicar posiciones
especificas en pantalla
    mov dh,10   ;fila
    mov dl,25  ;columna
    mov ah,2
    int 10h
    ret


ret

cadena1 db "Arquitectura
de computadoras II ","$"
cadena2 db "Segundo
semestre 2022 ","$"
cadena3  db 10,13,"1.
Operaciones Basicas","$"
```

PROYECTO FINAL ARQUITECTURA
CODIGO EMU8086
GRUPO #10

```
cadena4  db 10,13,"2.
Operaciones de
Cadenas","$"
cadena5  db 10,13,"3.
Juego","$"
cadena6  db 10,13,"4.
Operaciones de
Recurrencia","$"
cadena7  db 10,13,"5.
Salir","$"
cadena8  db
10,13,"Ingrese una opcion
:","$"


  ;--- OPERACIONES
BASICAS ----------------------
--------------------------
subcadenat1  db
"Operaciones Basicas","$"
subcadena1_1 db 10,13,
"1. Suma ","$"
subcadena1_2 db 10,13,
"2. Resta ","$"
subcadena1_3 db 10,13,
"3. Multiplicacion ","$"


  ;---- PALINDROMA -------
----------------------------------
--------------
subcadenat2  db
"Operaciones De
Cadenas","$"
subcadena2_1 db 10,13,
"1. Determinar si una
cadena es palindroma ","$"

  ;------ JUEGOS ------------
----------------------------------
-------------
subcadenat3  db " Juegos
","$"
subcadena3_1 db 10,13,
"1. Juego Snake ","$"

; --------- FIBONACCI -------
----------------------------------
-------------

subcadenat4  db
"Operaciones De
Recurrencia ","$"
subcadena4_1 db 10,13,
"1. Serie de Fibonacci ","$"


subcadenat5  db "
Saliendo ... ","$"
ret

;--- Librerias a utilizar ---
DEFINE_SCAN_NUM
DEFINE_PRINT_STRING
DEFINE_PRINT_NUM
DEFINE_PRINT_NUM_U
NS
DEFINE_PTHIS
DEFINE_CLEAR_SCREE
N
DEFINE_GET_STRING

;------- Suma ----------------
----------------------------------
---

Suma1:
suma db 2 dup (?)
;declaracion de variablede
suma
                ;acumulara los
datos

call clear_screen
gotoxy 30,1
call pthis
db '***Suma***',0
gotoxy 0,1
call pthis
```

PROYECTO FINAL ARQUITECTURA
CODIGO EMU8086
GRUPO #10

```
db 'Ingrese el primer
numero: ',0
call scan_num
mov suma[0],cl
gotoxy 0,2
call pthis
db 'Ingrese el segundo
numero: ',0
call scan_num
mov suma[1],cl
xor ax, ax ;Limpieza de
operacion
add al,suma[0]
add al,suma[1]
gotoxy 0,3
call pthis
db 'El resultado de la
suma es: ',0
call print_num
ret

;--- Resta ----------------------
------------------------------- ---
----------
Resta1:
resta db 2 dup (?)
;declaracion de variablede
resta
                    ;acumulara
los datos

call clear_screen
gotoxy 30,1
call pthis
db '***Resta***',0
gotoxy 0,4
call pthis
db 'Ingrese el primer
numero: ',0
call scan_num
mov resta[0],cl
gotoxy 0,5
call pthis
db 'Ingrese el segundo
numero: ',0
call scan_num
mov resta[1],cl
xor ax, ax ;Limpieza de
operacion
mov al,resta[0]
sub al,resta[1]
gotoxy 0,6
call pthis
db 'El resultado de la resta
es: ',0
call print_num
ret

;--- Multiplicacion ------------
-----------------------------------
-------
Multi1:

multi db 2 dup (?)
;declaracion de variablede
multiplicacion
                    ;acumulara
los datos

call clear_screen
gotoxy 30,1
call pthis
db '***Multiplicacion***',0
gotoxy 0,7
call pthis
db 'Ingrese el primer
numero: ',0
call scan_num
mov multi[0],cl
gotoxy 0,8
call pthis
db 'Ingrese el segundo
numero: ',0
call scan_num
mov multi[1],cl
xor ax, ax
;Limpieza de operacion
add al, multi[0]
mul multi[1]
gotoxy 0,9
call pthis
```

PROYECTO FINAL ARQUITECTURA
CODIGO EMU8086
GRUPO #10

```
db 'El resultado de la
multiplicacion es: ',0
call print_num
ret

;------------------- palindroma
----------------------------

palindroma:

call clear_screen

jmp inicio()
palabra db 50h dup(?)
aux1 db 0
aux2 dw 0

mensajes1:
primero db 'Ingrese
palabra: '
;lz = $ - cod
db 1Ah, '$'
exito:
exitomsj db 'La palabra
ingresada es palindroma.'
;lz = $ - cod
db 7Fh,'$'
fallo:
fallomsj db 'La palabra
ingresada no es
palindroma.'
;lz = $ - cod
db 58h,'$'

inicio():
mov ah,9
mov dx,offset primero
int 21h


mov bl,0dh ;tecla enter
mov si,00d ;inicio del
contador string
mov ah,1

lectura():
int 21h   ;lectura de
caracter
mov palabra[si],al
;almacenamos en el vector
string
inc si    ;incremento
variable SI
cmp al,bl ;comparamos el
caracter ingresado con el
enter
jne lectura()

mov di,si
dec di ;la variable si se
pasa en dos posiciones,
ya que almacena
dec di ; el enter y el retorn,
decrementa dos veces

mov si,00d

mov aux2,di
;almacenamos el largo en
una variable

compara():

mov al,palabra[si]
cmp palabra[di],al ;si son
distintos no es palindroma
jne no_palindromo()

cmp si,aux2
je palindromo()

inc si
dec di

jmp compara()

no_palindromo():
mov aux1,1
jmp imprime()
mensaje1():
;mov ah,0eh
mov al,0ah
```

PROYECTO FINAL ARQUITECTURA
CODIGO EMU8086
GRUPO #10

```
int 10h

mov dx,offset fallomsj
mov ah,9
int 21h
jmp fin()

palindromo():
jmp imprime()
mensaje2():
;mov ah,0eh
mov al,0ah
int 10h

mov dx,offset exitomsj
mov ah,9
int 21h
jmp fin()

imprime():
mov di,aux2
mov ah,0eh
mov al,0ah
int 10h
mov si,-01d
for():
mov ah,0eh
inc si
mov al,palabra[si]
int 10h
cmp si,di
jne for()

mov al,aux1 ;devuelve
cmp al,1
je mensaje1() ;vuelve
donde corresponda


jmp sub_menu2

fin():

ret
;end
```

```
;----------------------juego ----
--------------------------------

Juego_Snake:
name "snake"

call clear_screen

; saltar sobre la sección de
datos:
jmp     start

; ------ data section ------

s_size  equ     7

; las coordenadas de la
serpiente
; (de la cabeza a la cola)
; queda byte bajo, byte alto
; es arriba - [arriba,
izquierda]
snake dw s_size dup(0)

tail    dw      ?


; ENUMS para estado de
crecimiento
  NO_CHANGE       = 0
  BIGGER      = 1
  SMALLER   = 2

   NSEOI_OCW2 =
00100001b
  PC_PIC        = 20h


; constantes de direccion
; (codigos clave de bios):
left    equ     4bh
right   equ     4dh
up      equ     48h
down    equ     50h

cur_dir db      right
```

PROYECTO FINAL ARQUITECTURA
CODIGO EMU8086
GRUPO #10

```
wait_time dw    0

start:



; variables utilizadas para
random
  food_x    DB 1

; coordinadas de la
siguiente comida
  food_y    DB 1
                          ;
  attribute DB 13

  char      DB 41h

  food_type DB 1
; tipo de proxima comida,

;0: '-' (hace que la
serpiente sea mas
pequena), o.w: ABC char

; variables para alimentos
actuales
  cur_food_x        DB 0
  cur_food_y        DB 0
  cur_food_type     DB 0
  cur_food_char     DB 0
  cur_food_attribute DB 0

  grow_state  DB
NO_CHANGE ; opciones:
SIN CAMBIO, MAS
GRANDE o MAS
PEQUENO

  ezer_word   DW 0
  ezer_byte   DB 0
  ezer_byte2  DB 0
  direction_for_next_cycle
DB UP           ; contiene la
direccion para el proximo
                          ;ciclo en
bucle_principal
  one_before DB 0
          ; bandera para usar
en 'erase_tail' (ver alll)


   .code
  mov ax,@data
  mov ds,ax

  ent: up;llamar a
imprimir_comida
  ;llamar a
change_key_stroke_interr
upt ; cambia la direccion
de la rutina responsable
de la respuesta a la
pulsación de tecla
                          ;
para realizar
'update_direction'
  ;bucle principal:
    ; mov bucle_contador,0
       ;llamar acciones;
realiza todas las tareas
para una serpiente
movemadtes en la
pantalla, verifica la
colisión, etc.

; ocultar cursor de texto:
mov     ah, 1
mov     ch, 2bh
mov     cl, 0bh
int     10h

game_loop:

; === seleccione la
primera pagina de video
mov     al, 0 ; numero de
pagina.
mov     ah, 05h
int     10h
```

PROYECTO FINAL ARQUITECTURA
CODIGO EMU8086
GRUPO #10

```
; === mostrar nueva cabeza:
mov     dx, snake[0]


; poner el cursor en dl,dh
mov     ah, 02h
int     10h


; imprime '*' en la ubicacion:
mov     al, '*'
mov     ah, 09h
mov     bl, 0eh
; atributo.
mov     cx, 1
; caracter unico
int     10h

; === mantener la cola:
mov     ax, snake[s_size * 2 - 2]
mov     tail, ax

call    move_snake

; === ocultar cola vieja: solo la cabeza del gusanito cursor
mov     dx, tail

; poner el cursor en dl,dh solo la cabeza del cursor gusanito
mov     ah, 02h
int     10h


; imprime '' en la ubicación: necesario para que no se cicle
mov     al, ' '
mov     ah, 09h
mov     bl, 0eh
; atributo.
mov     cx, 1   ; caracter unico
int     10h

check_for_key:
; === verifique los comandos del jugador:
mov     ah, 01h
int     16h
jz      no_key

mov     ah, 00h
int     16h

cmp     al, 1bh    ; esc - key?
je      stop_game  ;

mov     cur_dir, ah

no_key:
 ; === espera unos momentos aquí:
; obtener el número de tics de reloj
; (alrededor de 18 por segundo)
; desde medianoche en cx:dx
mov     ah, 00h
int     1ah
cmp     dx, wait_time
jb      check_for_key
add     dx, 4
mov     wait_time, dx


; === bucle de juego eterno:
jmp     game_loop

 stop_game:

; mostrar el cursor hacia atrás:
mov     ah, 1
```

PROYECTO FINAL ARQUITECTURA
CODIGO EMU8086
GRUPO #10

```
mov     ch, 0bh
mov     cl, 0bh
int     10h

ret
   move_snake proc near

; establecer es en el
segmento de información
de bios:
mov     ax, 40h
mov     es, ax

; punto di a la cola
  mov   di, s_size * 2 - 2
 ; mover todas las partes
del cuerpo
  ; (el ultimo simplemente
se va)
  mov   cx, s_size-1
move_array:
  mov   ax, snake[di-2]
  mov   snake[di], ax
  sub   di, 2
  loop  move_array

  cmp     cur_dir, left
  je    move_left
cmp     cur_dir, right
  je    move_right
cmp     cur_dir, up
  je    move_up
cmp     cur_dir, down
  je    move_down

jmp     stop_move
; sin direccion.

move_left:
  mov   al, b.snake[0]
  dec   al
  mov   b.snake[0], al
  cmp   al, -1
  jne   stop_move
  mov   al, es:[4ah]    ; col
numero.
  dec   al
  mov   b.snake[0], al  ;
volver a la derecha.
  jmp   stop_move

  move_right:

  mov   al, b.snake[0]
  inc   al
  mov   b.snake[0], al
  cmp   al, es:[4ah]    ; col
numero.
  jb    stop_move
  mov   b.snake[0], 0  ;
volver a la izquierda.
  jmp   stop_move

  move_up:
  mov   al, b.snake[1]
  dec   al
  mov   b.snake[1], al
  cmp   al, -1
  jne   stop_move
  mov   al, es:[84h]    ; row
numero -1.
  mov   b.snake[1], al  ;
volver al fondo.
  jmp   stop_move

  move_down:
  mov   al, b.snake[1]
  inc   al
  mov   b.snake[1], al
  cmp   al, es:[84h]    ; row
numero -1.
  jbe   stop_move
  mov   b.snake[1], 0  ;
volver a la cima.
  jmp   stop_move

;contador_de_bucles,0
  stop_move:
  ret
move_snake endp

ret
```

PROYECTO FINAL ARQUITECTURA
CODIGO EMU8086
GRUPO #10

```
;----------------------Serie Fibobacci ----------------------

Fibonacci:
call clear_screen

PUSH    AX

        MOV     AH, 0Eh
        INT     10h
        POP     AX
ENDM

JMP start2       ; jump to start label
msg1 db "porfavor ingrese una valor para la cadena $" , 0Dh,0Ah, 24h ; define la variable
num1 dw ?

start2:  LEA     DX, msg1  ; carga direccion dx.
        MOV     AH, 09h
        INT     21h

CALL SCAN_NUM   ; obtenga el numero firmado de varios dígitos del teclado y almacene el resultado en el registro cx:
MOV num1,cx
putc 0Dh
putc 0Ah

CMP CX, 1
JLE lessthan

greater_or_equal:          ; si cx no es menor que 1  / se hace comparacion
    CMP CX, 25             ; compara cx con 25
    JLE lessthan_or_equal
; si cx es menor que 25 salta a less_or_equal

    greater__or__equal:
        JMP restart        ; salta a restart label
        msg2 db "por favor ingrese el número adecuado en el rango de [1,25] $" , 0Dh,0Ah, 24h  ; define variable


        restart:  LEA     DX, msg2  ; direccion del mensaje dx
                  MOV     AH, 09h
                  INT     21h

                  MOV     AH, 0
                  INT     16h       ; espera tecla de usuario
                  putc 0Dh
                  putc 0Ah
                  JMP start2
; salta a etiqueta inicio

lessthan:
    CMP     CX, 0         ; comparar cx con 0
    JNZ     restart2      ; si cx no es igual a 0 salta a reiniciar
    JE stop               ; si cx es igual a 0 salta a la etiqueta en stop

    stop:
    MOV AH, 4CH
    MOV AL, 01
;retorno
    INT 21H

    JMP restart2
```

PROYECTO FINAL ARQUITECTURA
CODIGO EMU8086
GRUPO #10

```
   msg3 db "por favor
ingrese el número
adecuado en el rango de
[1,25] $" , 0Dh,0Ah, 24h

; define variable

   restart2:
            CMP     CX, 1
; compara cx con uno
            JE      faboo
            LEA     DX, msg3
; cargue la direccion
efectiva de msg en dx.
            MOV     AH, 09h
            INT     21h

            MOV     AH, 0
            INT     16h
            putc 0Dh
            putc 0Ah
            JMP start2

   lessthan_or_equal:
; etiqueta menor que o
igual
       MOV BX, 1
       MOV AX , 0
       CALL PRINT_NUM
; llame a la etiqueta
print_num (imprima el
valor en ax)
       MOV AX , 1
       CALL PRINT_NUM
       SUB CX, 1
; resta 1 de cx
       fabo:
             ADD AX,BX
; agregue ax a bx y
almacene el resultado en
ax
             MOV [SI],AX
             MOV AX,BX
             MOV BX,[SI]
             CALL
PRINT_NUM
             INC SI          ;
incrementa si

       LOOP fabo R
;bucle fabo numero de
bucles igual al valor de cx
       putc 0Dh
       putc 0Ah
       JMP start2

       faboo:
            MOV AX , 0
            CALL
PRINT_NUM
            MOV AX , 1
            CALL
PRINT_NUM
            putc 0Dh
            putc 0Ah
            JMP start2


; obtiene el numero
FIRMADO de varios
dígitos del teclado y
almacena el resultado en
;el registro CX:
SCAN_NUM2       PROC
NEAR
       PUSH    DX
       PUSH    AX
       PUSH    SI

       MOV     CX, 0

       ; reset flag:
       MOV
CS:make_minus, 0

next_digit:

       ;obtener caracteres
del teclado
       ; into AL:
       MOV     AH, 00h
       INT     16h
```

PROYECTO FINAL ARQUITECTURA
CODIGO EMU8086
GRUPO #10

```
        ; imprimirlo
        MOV     AH, 0Eh
        INT     10h



        CMP     AL, '-'
        JE      set_minus

      ; verifique la tecla
ENTER:
        CMP     AL, 0Dh
        JNE     not_cr
        JMP     stop_input
not_cr:

        CMP     AL, 8
; retrocedo
        JNE
backspace_checked
        MOV     DX, 0
; eliminar el último dígito
por
        MOV     AX, CX
; division:
        DIV     CS:ten
; AX = DX:AX / 10 (DX-
rem).
        MOV     CX, AX
        PUTC    ' '
; clear position.
        PUTC    8
; retroceder de nuevo.
        JMP     next_digit
backspace_checked:

        ;permite solo digitos
        CMP     AL, '0'
        JAE     ok_AE_0
        JMP
remove_not_digit
ok_AE_0:
        CMP     AL, '9'
        JBE     ok_digit
remove_not_digit:
        PUTC    8         ;
retroceder
        PUTC    ' '       ;borrar
el último digito no
ingresado
        PUTC    8
;retroceder de nuevo.
        JMP     next_digit ;
esperar a la siguiente
entrada.
ok_digit:


        ; multiplica CX por 10
(primera vez el resultado
es cero)
        PUSH    AX
        MOV     AX, CX
        MUL     CS:ten    ;
DX:AX = AX*10
        MOV     CX, AX
        POP     AX

        ; compruebe si el
numero es demasiado
grande. 16 bits
        CMP     DX, 0
        JNE     too_big

        ; convertir de código
ASCII
        SUB     AL, 30h



        MOV     AH, 0
        MOV     DX, CX
        ADD     CX, AX
        JC      too_big2   ;
salta si el número es
demasiado grande.

        JMP     next_digit

set_minus:
        MOV
CS:make_minus, 1
        JMP     next_digit
```

PROYECTO FINAL ARQUITECTURA
CODIGO EMU8086
GRUPO #10

```
too_big2:
        MOV     CX, DX
        MOV     DX, 0
too_big:
        MOV     AX, CX
        DIV     CS:ten     ;
reverse last DX:AX =
AX*10, make AX = DX:AX
/ 10
        MOV     CX, AX
        PUTC    8          ;
retroceso.
        PUTC    ' '        ;
        PUTC    8          ;
retroceder de nuevo.
        JMP     next_digit ;
espere Entrar/Retroceso.

stop_input:

        CMP
CS:make_minus, 0
        JE      not_minus
        NEG     CX
not_minus:

        POP     SI
        POP     AX
        POP     DX
        RET
make_minus      DB      ?
SCAN_NUM2       ENDP

; este procedimiento
imprime el numero en AX
usado con
PRINT_NUM_UNS
; para imprimir numeros
con signo:
PRINT_NUM2      PROC
NEAR
        PUSH    DX
        PUSH    AX

        CMP     AX, 0
        JNZ     not_zero
        PUTC    '0'
        JMP     printed
not_zero:
        ; el SIGNO de
verificación de AX se
convierte en absoluto si es
negativo
        CMP     AX, 0
        JNS     positive
        NEG     AX

        PUTC    '-'
positive:
        CALL
PRINT_NUM_UNS
printed:
        POP     AX
        PUTC    ','
        POP     DX
        RET
PRINT_NUM2      ENDP
PRINT_NUM_UNS2
PROC    NEAR
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX

        ;bandera para evitar
la impresión de ceros
antes del número:
        MOV     CX, 1

        ; (result of "/ 10000"
siempre es menor o igual
a 9).
        MOV     BX, 10000
        CMP     AX, 0
        JZ      print_zero

begin_print:

        ; verifique el divisor
(si es cero, vaya a
end_print):
```

PROYECTO FINAL ARQUITECTURA
CODIGO EMU8086
GRUPO #10

```
        CMP     BX,0
        JZ      end_print
        CMP     CX, 0
        JE      calc
        CMP     AX, BX
        JB      skip
calc:
        MOV     CX, 0

        MOV     DX, 0
        DIV     BX

       ; AH siempre es
CERO, por lo que se
ignora
        ADD     AL, 30h
        PUTC    AL
        MOV     AX, DX  ;
obtener el resto de la
última div.
skip:
        PUSH    AX
        MOV     DX, 0
        MOV     AX, BX
        DIV     CS:ten  ; AX =
DX:AX / 10   (DX=resto).
        MOV     BX, AX
        POP     AX

        JMP     begin_print
print_zero:
        PUTC    '0'

end_print:

        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
PRINT_NUM_UNS2
ENDP

ten DW 10

ret
```