Universidad Autónoma de Baja California

Facultad de Ciencias Químicas e Ingeniería



**ORGANIZACIÓN DE LAS COMPUTADORAS Y LENGUAJE ENSAMBLADOR**

**Practica 7**

**Estructura de control en lenguaje ensamblador para el procesador 8086**

**Docente:** Sanchez Herrera Mauricio Alonso

**Alumno:** Gómez Cárdenas Emmanuel Alberto

**Matricula:** 1261509

Contenido

[TEORIA 3](#_Toc71142025)

[Directivas del leguaje ensamblador 3](#_Toc71142026)

[DB (DEFINE BYTE) 3](#_Toc71142027)

[DW (DEFINE WORD) 3](#_Toc71142028)

[DD (DEFINE DOUBLE WORD) 3](#_Toc71142029)

[DQ (DEFINE QUAD WORD) 3](#_Toc71142030)

[DT (DEFINE TEN BYTES) 3](#_Toc71142031)

[SEGMENT 3](#_Toc71142032)

[ENDS (END SEGMENT) 3](#_Toc71142033)

[PROC (PROCEDIMIENTO) 4](#_Toc71142034)

[ENDP (END PROCEDURE) 4](#_Toc71142035)

[EQU (EQUATE o EQUIPARA) 4](#_Toc71142036)

[ALIGN 4](#_Toc71142037)

[ASSUME 4](#_Toc71142038)

[ORG (ORIGEN) 4](#_Toc71142039)

[EXTRN 4](#_Toc71142040)

[OFFSET 4](#_Toc71142041)

[PTR (APUNTADOR) 4](#_Toc71142042)

[DESARROLLO 5](#_Toc71142043)

[PARTE 1 Hola Mundo 5](#_Toc71142044)

[PARTE 2 6](#_Toc71142045)

[1. IF\_THEN 6](#_Toc71142046)

[3. CASE 7](#_Toc71142047)

[4. FOR 8](#_Toc71142048)

[5. WHILE\_DO 9](#_Toc71142049)

[6. DO\_WHILE 9](#_Toc71142050)

[CONCLUSIONES 10](#_Toc71142051)

[REFERENCIAS 10](#_Toc71142052)

[ANEXOS 10](#_Toc71142053)

[A. FORMATO 10](#_Toc71142054)

[1. IF-THEN 11](#_Toc71142055)

[2. IF-THEN-ELSE 12](#_Toc71142056)

[3. SWITCH CASE 13](#_Toc71142057)

[4. FOR 15](#_Toc71142058)

[5. WHILE 16](#_Toc71142059)

[6. DO\_WHILE 17](#_Toc71142060)

# TEORIA

## Directivas del leguaje ensamblador

Las directivas son comandos que afectan al compilador (también llamadas pseudo operaciones que controlan el proceso de ensamblado) y no al microprocesador por lo que estas no generan código objeto. Son utilizadas para definir segmentos símbolos, subrutinas, para generar memoria y entre otras cosas.

### DB (DEFINE BYTE)

Es usada para declarar una variable de tipo byte, o para reservar una o más locaciones de memoria de tipo byte en memoria.

### DW (DEFINE WORD)

Es usada para declarar una variable de tipo word, o para reservar una o más locaciones de memoria de tipo word en memoria.

### DD (DEFINE DOUBLE WORD)

Es usada para declarar una variable de tipo double word, las cuales pueden ser accedidas como una palabra de tipo double

### DQ (DEFINE QUAD WORD)

Es usada para decirle al ensamblador que declare una variable de 4 palabras de largo o que reserve 4 palabras de almacenamiento en memoria.

### DT (DEFINE TEN BYTES)

Es usada para declarar una variable de 10 bytes de largo o para reservar 10 bytes de almacenamiento en memoria.

### SEGMENT

Es usada para indicar el inicio de un segmento lógico. Le precede el nombre que se le quiera dar al segmento. Ej. CODE SEGMENT le indica al ensamblado el inicio de un segmento lógico llamado CODE.

### ENDS (END SEGMENT)

Es usada para indicar el final de un segmento lógico. Las directivas SEGMENT y END son usadas para encerrar un segmento lógico.

### PROC (PROCEDIMIENTO)

Es usada para identificar el inicio de un procedimiento. Le precede el nombre que se le quiera dar al procedimiento. Después de la directiva PROC el termino NEAR o FAR es usado para especificar el tipo de procedimiento.

### ENDP (END PROCEDURE)

Es usada para indicar el final de un procedimiento al ensamblador. Las directivas PRCO y ENDP son usadas para encerrar un procedimiento.

### EQU (EQUATE o EQUIPARA)

Esta directiva es usada para darle un nombre o etiqueta a algún valor o símbolo, cada vez que el ensamblador encuentre la etiqueta en el programa, esta es reemplazada con el valor o símbolo equiparado a ella. Se usa para definir constantes dentro del programa.

### ALIGN

El Array de memoria es almacenado en límites de palabras. Ej. ALIGN 2 significa que se almacena en direcciones pares.

### ASSUME

Le dice al ensamblador que nombres han sido elegidos para el segmento de código, de datos y de la pila. Ej. ASSUME CS: CODE2

### ORG (ORIGEN)

Cambia el offset de dirección inicial en el segmento de datos. Esta directiva permite fijar el contador de locación a cualquier valor desea en cualquier punto del programa.

### EXTRN

Es usada para indicarle al ensamblador que el nombre o etiqueta que le procede están en otro modulo ensamblador.

### OFFSET

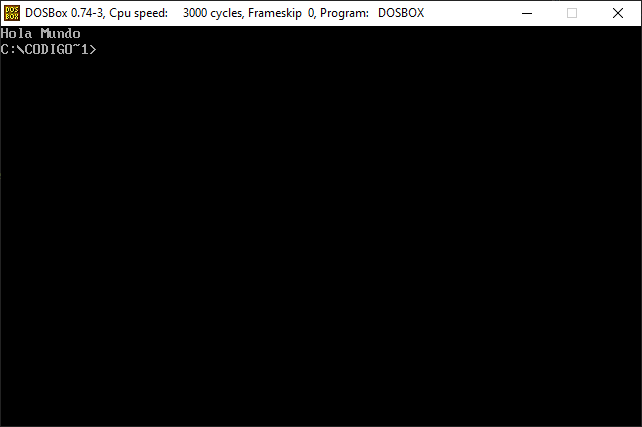
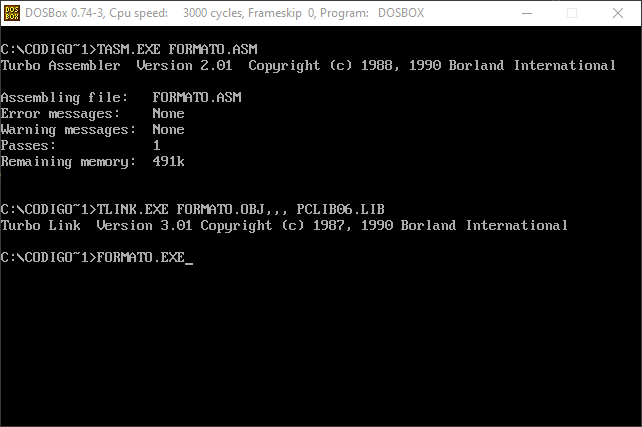
Es un operador que le indica al ensamblador que determine el desplazamiento de una variable.

### PTR (APUNTADOR)

Es un apuntador usado para asignar un tipo específico a una variable o etiqueta. Es necesario utilizarse cuando el tipo del operando es incierto.

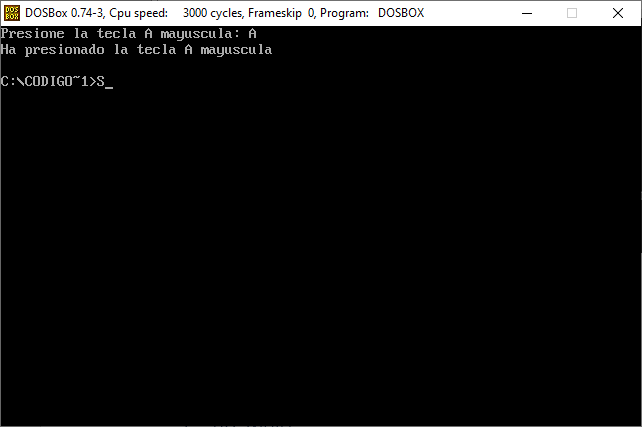
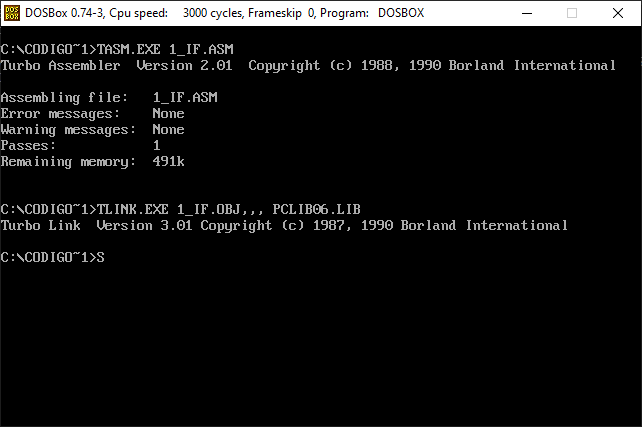
DESARROLLO

## PARTE 1 Hola Mundo

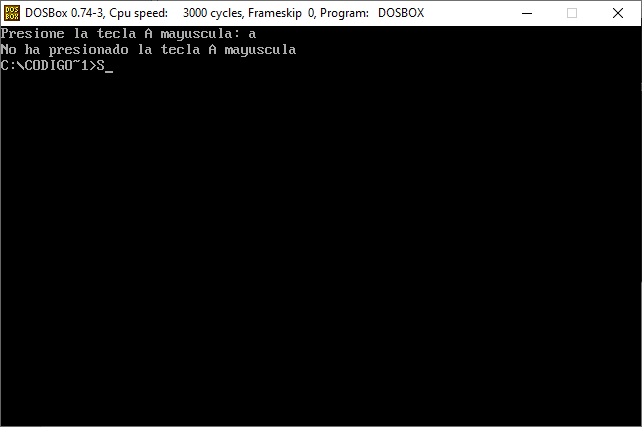
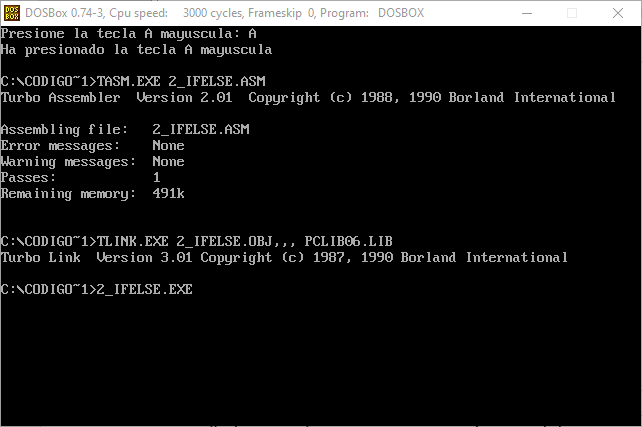


## PARTE 2

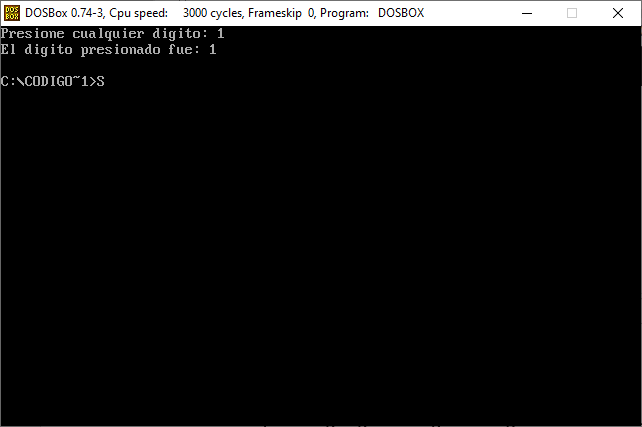
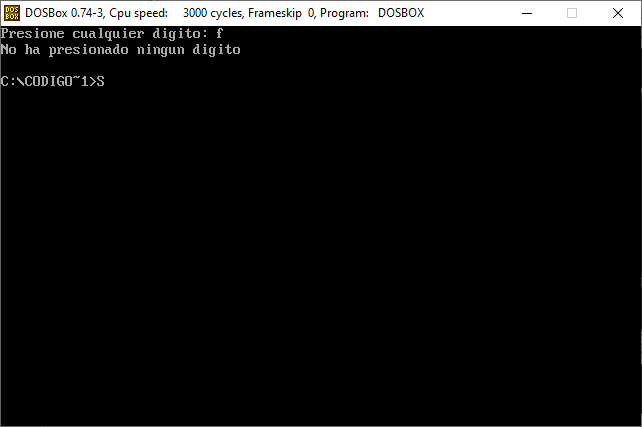
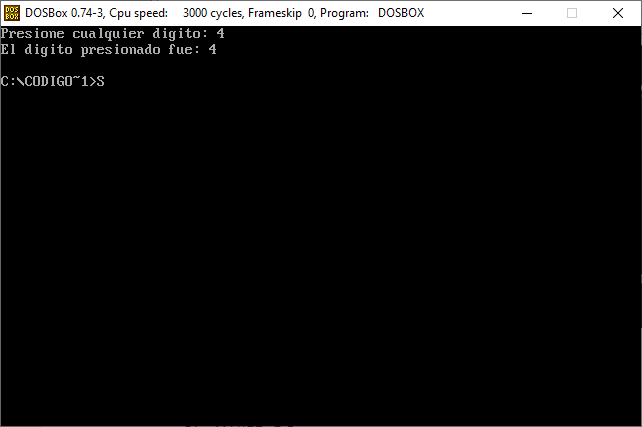
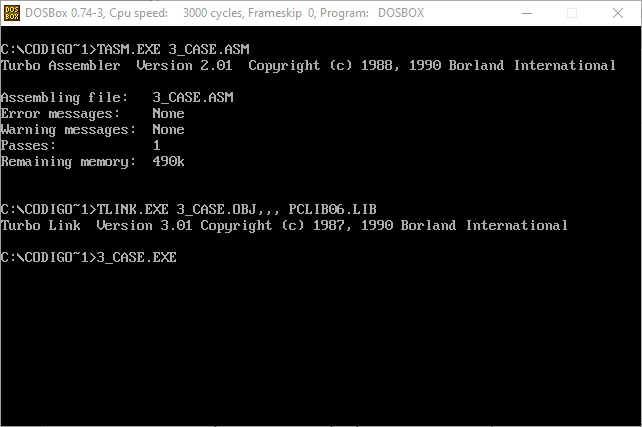
### IF\_THEN



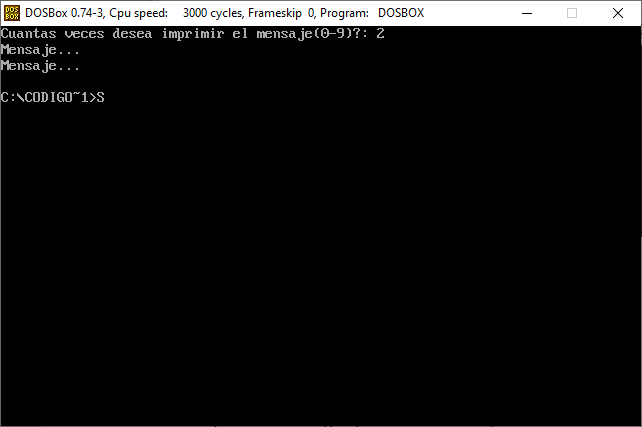
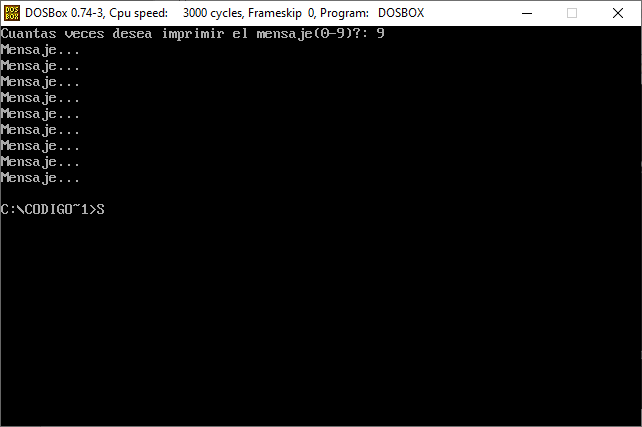
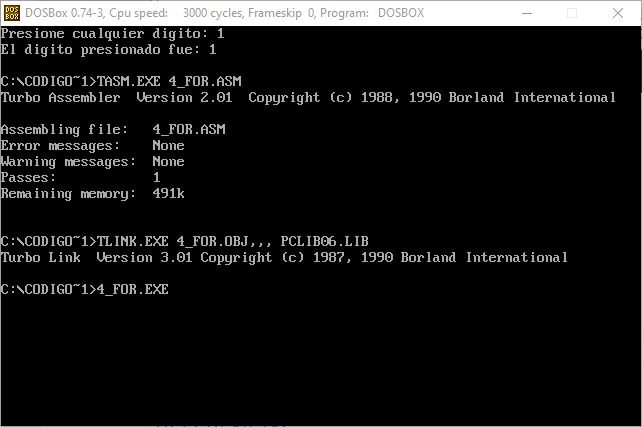
1. IF\_THEN\_ELSE



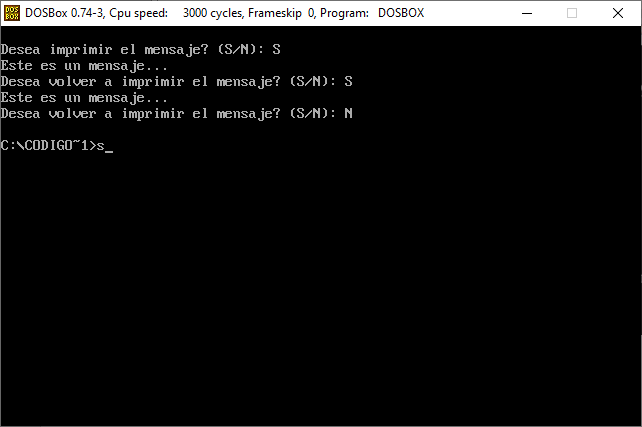
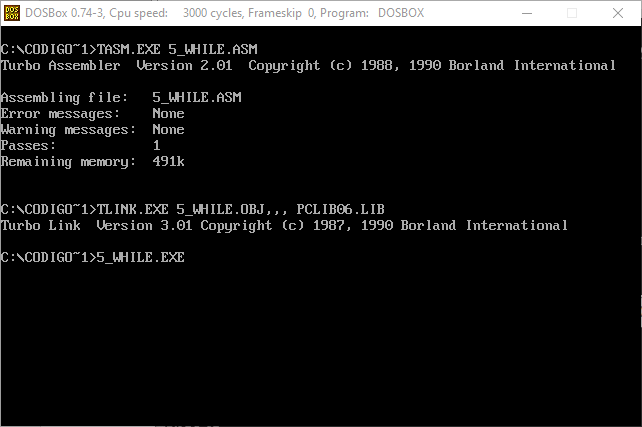
### CASE



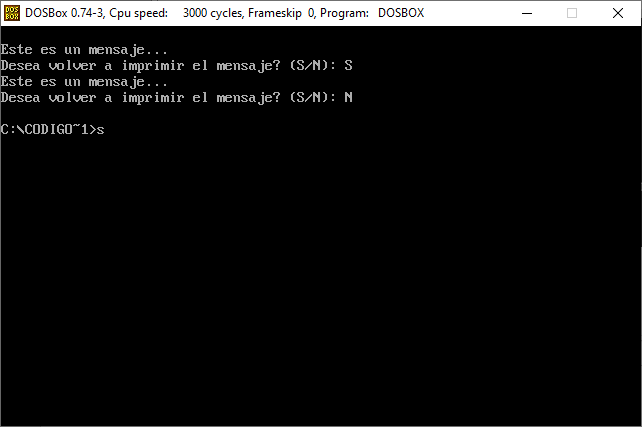
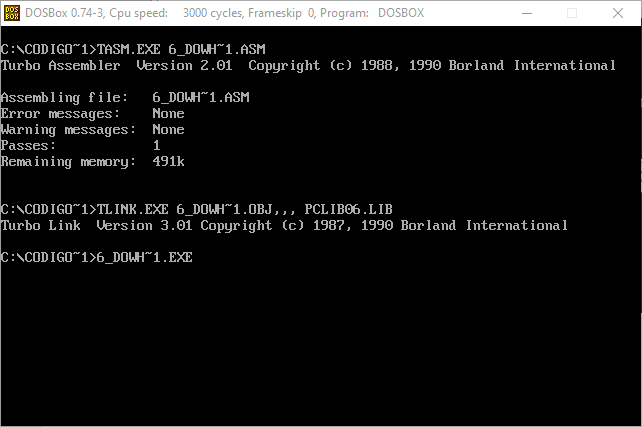
### FOR



### WHILE\_DO



### DO\_WHILE



# CONCLUSIONES

Programar las estructuras de control básicas nos ayudó a entender aún más el concepto de lo que es el lenguaje ensamblador, estas estructuras de control son la parte más básica de cualquier lenguaje de programación y con estas se pueden crear casi cualquier tipo de estructuras, procedimientos y hasta programas complejos.

# REFERENCIAS

*2 Assembly Language Programming*. Cs.unm.edu. (2020). Retrieved from https://www.cs.unm.edu/~maccabe/classes/341/labman/node2.html.

# ANEXOS

## FORMATO

MODEL small

.STACK 100h

;----- Insert INCLUDE "filename" directives here

;----- Insert EQU and = equates here

INCLUDE procs.inc

LOCALS

.DATA

    mens db 'Hola Mundo',0

.CODE

;-----   Insert program, subrutine call, etc., here

Principal   PROC

            mov ax,@data    ;Inicializar DS al la direccion

            mov ds,ax       ; del segmento de datos (.DATA)

            call clrscr

            mov  dx,offset mens

            call puts

            call getch

            mov ah,04ch      ; fin de programa

            mov al,0             ;

            int 21h              ;

Principal   ENDP

; incluir procedimientos

; ejemplo:

; funcionX  PROC ; < -- Indica a TASM el inicio del un procedimiento

;               ;

;               ; < --- contenido del procedimiento

;           ret

;           ENDP; < -- Indica a TASM el fin del procedimiento

        END

## IF-THEN

MODEL small

.STACK 100h

INCLUDE procs.inc

LOCALS

.DATA

    ask   db  'Presione la tecla A mayuscula: ',0

    message   db  'Ha presionado la tecla A mayuscula',0

.CODE

Principal   PROC

            mov ax,@data   ;Inicializar DS al la dirección

            mov ds,ax      ; del segmento de datos (.DATA)

            mov bl,41h     ;Introduce el valor ascii hex de la A mayúscula

            call clrscr

            mov  dx,offset ask

            call puts

            call getchar   ;Espera una tecla del usuario

            call println   ;Imprime un salto de línea

            cmp al,bl      ;Compara el valor introducido con bl

            JNE @@EndIF    ;Si no son iguales se sale

@@IfThen:   mov dx,offset message

            call puts

            call println   ;Imprime un salto de línea

@@EndIF:    mov ah,04ch      ; fin de programa

            mov al,0             ;

            int 21h              ;

Principal   ENDP

println     PROC ; Funcion para imprimir un salto de linea

            push ax        ;push para guardar su valor en la pila

            mov al,10      ;Salto de línea y

            call putchar

            mov al,13      ;Retorno de carro

            call putchar

            pop ax     ;pop para obtener el valor del registro antes de modificar

            ret

println     ENDP; < -- Indica a TASM el fin del procedimiento

        END

## IF-THEN-ELSE

MODEL small

.STACK 100h

INCLUDE procs.inc

LOCALS

.DATA

    ask db 'Presione la tecla A mayuscula: ',0

    messageIF db 'Ha presionado la tecla A mayuscula',0

    messageIFElse db 'No ha presionado la tecla A mayuscula',0

    endASK db 'Presione cualquier tecla para salir...',0

.CODE

Principal   PROC

            mov ax,@data    ;Inicializar DS al la direccion

            mov ds,ax       ; del segmento de datos (.DATA)

            mov bl,41h     ;Introduce el valor ascii hex de la A mayuscula

            call clrscr

            mov  dx,offset ask

            call puts

            call getchar   ;Espera una tecla del usuario

            call println   ;Imprime un salto de linea

            cmp al,bl      ;Compara el valor introducido con bl

            JNE @@Else    ;Si son iguales imprime un mensaje

@@messageIF:

            mov dx,offset messageIF

            JMP @@EndIF

@@Else:     mov dx,offset messageIFElse

@@EndIF:    call puts

            mov ah,04ch      ; fin de programa

            mov al,0             ;

            int 21h              ;

Principal   ENDP

println     PROC ; Funcion para imprimir un salto de linea

            push ax        ;push para guardar su valor en la pila

            mov al,10      ;Salto de línea y

            call putchar

            mov al,13      ;Retorno de carro

            call putchar

            pop ax     ;pop para obtener el valor del registro antes de modificar

            ret

println     ENDP; < -- Indica a TASM el fin del procedimiento

        END

## SWITCH CASE

MODEL small

.STACK 100h

INCLUDE procs.inc

LOCALS

.DATA

    ask db 'Presione cualquier digito: ',0

    defaultMessage db 'No ha presionado ningun digito',0

    scaseMessage db 'El digito presionado fue: ',0

.CODE

;-----   Insert program, subrutine call, etc., here

Principal   PROC

            mov ax,@data    ;Inicializar DS al la direccion

            mov ds,ax       ; del segmento de datos (.DATA)

            call clrscr

            mov  dx,offset ask

            call puts

            call getchar   ;Espera una tecla del usuario

            call println   ;Imprime un salto de linea

            mov bl,30h     ;Compara si es char '0'

            cmp al,bl      ;Compara el valor introducido con bl

            JE @@SCASE     ;JMP al case indicado

            mov bl,31h     ;Compara si es char '1'

            cmp al,bl

            JE @@SCASE

            mov bl,32h     ;Compara si es char '2'

            cmp al,bl

            JE @@SCASE

            mov bl,33h     ;Compara si es char '3'

            cmp al,bl

            JE @@SCASE

            mov bl,34h     ;Compara si es char '4'

            cmp al,bl

            JE @@SCASE

            mov bl,35h     ;Compara si es char '5'

            cmp al,bl

            JE @@SCASE

            mov bl,36h     ;Compara si es char '6'

            cmp al,bl

            JE @@SCASE

            mov bl,37h     ;Compara si es char '7'

            cmp al,bl

            JE @@SCASE

            mov bl,38h     ;Compara si es char '8'

            cmp al,bl

            JE @@SCASE

            mov bl,39h     ;Compara si es char '9'

            cmp al,bl

            JE @@SCASE

            JMP @@DEFAULT

@@SCASE:    mov dx,offset scaseMessage

            call puts

            mov al,bl      ;Se copia el valor de bl a al

            call putchar   ;imprime el valor almacenado en al

            call println   ;Imprime un salto de linea

            JMP @@EndSwitchCase

@@DEFAULT:  mov dx,offset defaultMessage

            call puts

            call println

            JMP @@EndSwitchCase

@@EndSwitchCase:

            mov ah,04ch    ; fin de programa

            mov al,0       ;

            int 21h        ;

Principal   ENDP

println     PROC ; Funcion para imprimir un salto de linea

            push ax        ;push para guardar su valor en la pila

            mov al,10      ;Salto de línea y

            call putchar

            mov al,13      ;Retorno de carro

            call putchar

            pop ax     ;pop para obtener el valor del registro antes de modificar

            ret

println     ENDP; < -- Indica a TASM el fin del procedimiento

        END

## FOR

MODEL small

.STACK 100h

INCLUDE procs.inc

LOCALS

.DATA

    ask db 'Cuantas veces desea imprimir el mensaje(0-9)?: ',0

    xcptn db 'Por favor, seleccionar un numero valido',0

    message db 'Mensaje...',0

.CODE

;-----   Insert program, subrutine call, etc., here

Principal   PROC

            mov ax,@data    ;Inicializar DS al la direccion

            mov ds,ax       ; del segmento de datos (.DATA)

            mov bl,30h

            call clrscr

            mov  dx,offset ask

            call puts

            call getchar

            mov cl,al

            cmp cl,3Fh

            JG @@Exception

@@StartFOR: call println

            cmp cl,bl

            JE @@EndFor

            dec cl

            mov dx,offset message

            call puts

            JMP @@StartFOR

@@Exception:

            call println   ;Imprime un salto de linea

            mov dx,offset xcptn

            call puts

@@EndFor:   mov ah,04ch    ; fin de programa

            mov al,0       ;

            int 21h        ;

Principal   ENDP

println     PROC ; Funcion para imprimir un salto de linea

            push ax        ;push para guardar su valor en la pila

            mov al,10      ;Salto de línea y

            call putchar

            mov al,13      ;Retorno de carro

            call putchar

            pop ax     ;pop para obtener el valor del registro antes de modificar

            ret

println     ENDP; < -- Indica a TASM el fin del procedimiento

        END

## WHILE\_DO

MODEL small

.STACK 100h

INCLUDE procs.inc

LOCALS

.DATA

    ask db 'Desea imprimir el mensaje? (S/N): ',0

    askAgain db 'Desea volver a imprimir el mensaje? (S/N): ',0

    message db 'Este es un mensaje...',0

.CODE

;-----   Insert program, subrutine call, etc., here

Principal   PROC

            mov ax,@data    ;Inicializar DS al la direccion

            mov ds,ax       ; del segmento de datos (.DATA)

            mov bl,4Eh     ;Valor ascii del char 'n'

            call clrscr

            call println   ;Imprime un salto de linea

            mov  dx,offset ask

            call puts

            call getchar

            mov cl,al

            call println

@@While:    cmp cl,bl      ;While

            JE @@EndWhile

            JMP @@StartWhile

@@StartWhile:

            mov  dx,offset message

            call puts

            call println   ;Imprime un salto de linea

            mov  dx,offset askAgain

            call puts

            call getchar

            mov cl,al

            call println

            jmp @@While

@@EndWhile: mov ah,04ch    ; fin de programa

            mov al,0       ;

            int 21h        ;

Principal   ENDP

println     PROC ; Funcion para imprimir un salto de linea

            push ax        ;push para guardar su valor en la pila

            mov al,10      ;Salto de línea y

            call putchar

            mov al,13      ;Retorno de carro

            call putchar

            pop ax     ;pop para obtener el valor del registro antes de modificar

            ret

println     ENDP; < -- Indica a TASM el fin del procedimiento

        END

## DO\_WHILE

MODEL small

.STACK 100h

INCLUDE procs.inc

LOCALS

.DATA

    ask db 'Desea volver a imprimir el mensaje? (S/N): ',0

    message db 'Este es un mensaje...',0

.CODE

;-----   Insert program, subrutine call, etc., here

Principal   PROC

            mov ax,@data    ;Inicializar DS al la direccion

            mov ds,ax       ; del segmento de datos (.DATA)

            mov bl,4Eh     ;Valor ascii del char 'n'

            call clrscr

            call println   ;Imprime un salto de linea

@@StartDo:  mov  dx,offset message

            call puts      ;Do

            call println   ;Imprime un salto de linea

            mov  dx,offset ask

            call puts

            call getchar

            mov cl,al

            call println

            cmp cl,bl      ;While

            JE @@EndWhile

            JMP @@StartDo

@@EndWhile: mov ah,04ch    ; fin de programa

            mov al,0       ;

            int 21h        ;

Principal   ENDP

println     PROC ; Funcion para imprimir un salto de linea

            push ax        ;push para guardar su valor en la pila

            mov al,10      ;Salto de línea y

            call putchar

            mov al,13      ;Retorno de carro

            call putchar

            pop ax     ;pop para obtener el valor del registro antes de modificar

            ret

println     ENDP; < -- Indica a TASM el fin del procedimiento

        END