

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
Directivas del lenguaje ensamblador	3
DB (DEFINE BYTE)	3
DW (DEFINE WORD)	3
DD (DEFINE DOUBLE WORD)	3
DQ (DEFINE QUAD WORD)	3
DT (DEFINE TEN BYTES).....	3
SEGMENT	3
ENDS (END SEGMENT)	3
PROC (PROCEDIMIENTO)	4
ENDP (END PROCEDURE)	4
EQU (EQUATE o EQUIPARA).....	4
ALIGN	4
ASSUME.....	4
ORG (ORIGEN)	4
EXTRN.....	4
OFFSET	4
PTR (APUNTADOR).....	4
DESARROLLO.....	5
PARTE 1 IF_THEN.....	5
PARTE 2.....	6
1. IF_THEN.....	6
2. IF_THEN_ELSE.....	7
3. CASE	7
4. FOR	8
5. WHILE_DO	9
6. DO_WHILE	10
CONCLUSIONES.....	11
REFERENCIAS.....	11
ANEXOS	12
1. IF_THEN	13

2.	IF_ELSE.....	14
3.	SWITCH CASE	14
4.	FOR	17
5.	WHILE_DO	18
6.	DO_WHILE	19

TEORIA

Directivas del lenguaje 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 PROC 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

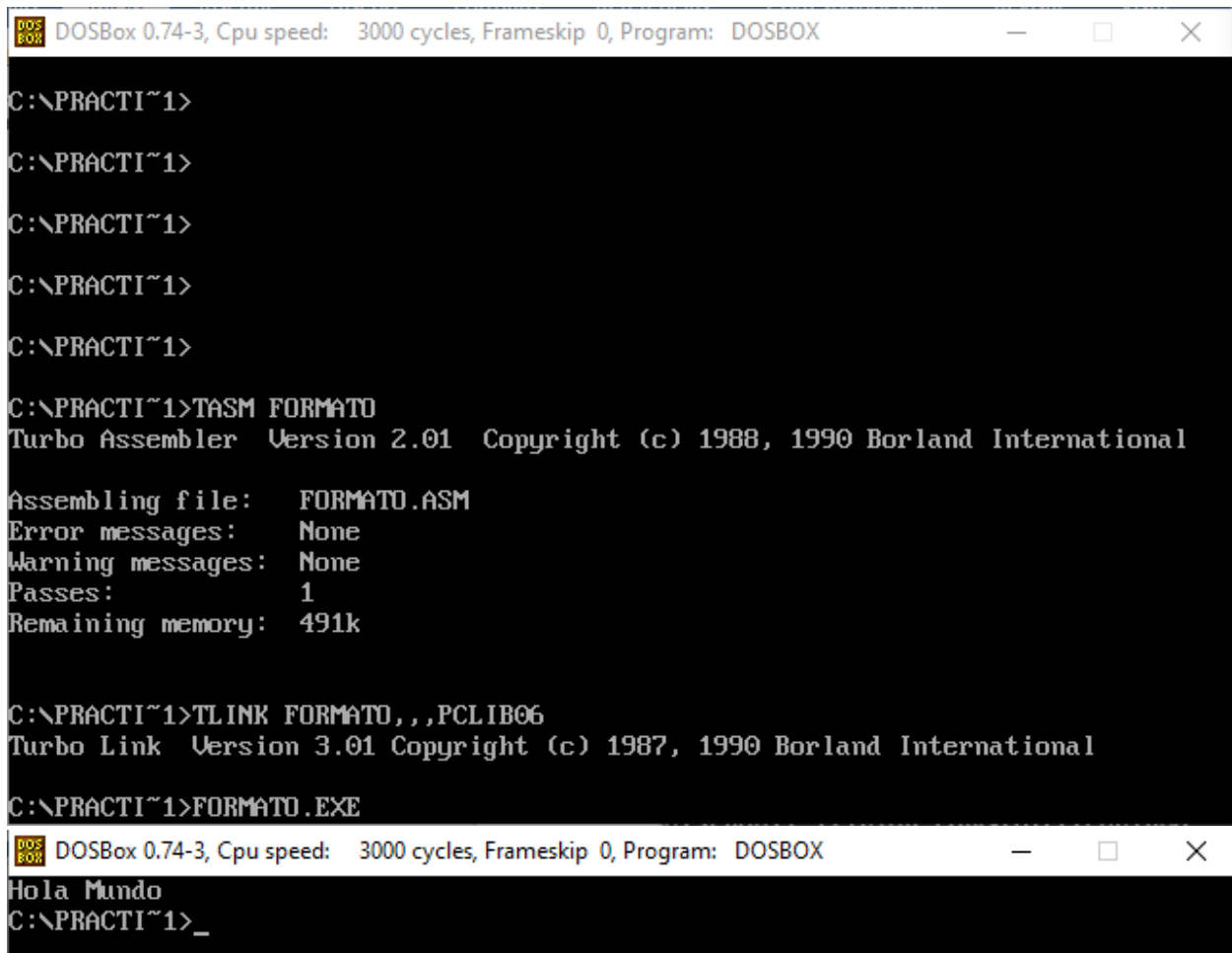
Es un operador que le indica al ensamblador que determine el desplazamiento de una variable.

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



The screenshot shows a DOSBox 0.74-3 window with a black background and white text. The window title bar reads "DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX". The command prompt shows the following sequence of commands and output:

```
C:\PRACTI~1>
C:\PRACTI~1>
C:\PRACTI~1>
C:\PRACTI~1>
C:\PRACTI~1>
C:\PRACTI~1>TASM FORMATO
Turbo Assembler Version 2.01 Copyright (c) 1988, 1990 Borland International

Assembling file:   FORMATO.ASM
Error messages:   None
Warning messages: None
Passes:           1
Remaining memory: 491k

C:\PRACTI~1>TLINK FORMATO,,,PCLIB06
Turbo Link Version 3.01 Copyright (c) 1987, 1990 Borland International

C:\PRACTI~1>FORMATO.EXE
```

Below the DOSBox window, the text "Hola Mundo" is displayed, followed by a new command prompt line:

```
C:\PRACTI~1>_
```

PARTE 2

1. IF_THEN

```
C:\PRACTI~1>TASM IF_THEN
Turbo Assembler Version 2.01 Copyright (c) 1988, 1990 Borland International

Assembling file:    IF_THEN.ASM
Error messages:     None
Warning messages:   None
Passes:             1
Remaining memory:   491k

C:\PRACTI~1>TLINK.EXE IF_THEN,,,PCLIB06
Turbo Link Version 3.01 Copyright (c) 1987, 1990 Borland International

C:\PRACTI~1>IF_THEN.EXE

Presione la tecla A mayuscula: S

C:\PRACTI~1>s

Presione la tecla A mayuscula: A
Ha presionado la tecla A mayuscula

C:\PRACTI~1>s_
```


2. IF_THEN_ELSE

```
C:\PRACTI~1>TASM IF_ELSE
Turbo Assembler Version 2.01 Copyright (c) 1988, 1990 Borland International

Assembling file:    IF_ELSE.ASM
Error messages:     None
Warning messages:   None
Passes:             1
Remaining memory:   491k
```

```
C:\PRACTI~1>TLINK.EXE IF_ELSE,,,PCLIB06
Turbo Link Version 3.01 Copyright (c) 1987, 1990 Borland International
```

```
C:\PRACTI~1>IF_ELSE.EXE
```

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

```
Presione la tecla A mayuscula: S
No ha presionado la tecla A mayuscula
C:\PRACTI~1>s
```

```
Presione la tecla A mayuscula: A
Ha presionado la tecla A mayuscula
```


```
C:\PRACTI~1>s_
```

3. CASE

```
C:\PRACTI~1>TASM SCASE
Turbo Assembler Version 2.01 Copyright (c) 1988, 1990 Borland International

Assembling file:    SCASE.ASM
Error messages:     None
Warning messages:   None
Passes:             1
Remaining memory:   490k
```

```
C:\PRACTI~1>TLINK.EXE SCASE,,,PCLIB06
Turbo Link Version 3.01 Copyright (c) 1987, 1990 Borland International
```

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

```
Presione cualquier digito: D
No ha presionado ningun digito
```

```
C:\PRACTI~1>_
```

```
DOS
BOX DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
Presione cualquier digito: 2
El digito presionado fue: 2

C:\PRACTI~1>
```

```
DOS
BOX DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
Presione cualquier digito: 6
El digito presionado fue: 6

C:\PRACTI~1>
```

4. FOR

```
C:\PRACTI~1>TASM T_FOR
Turbo Assembler Version 2.01 Copyright (c) 1988, 1990 Borland International

Assembling file: T_FOR.ASM
Error messages: None
Warning messages: None
Passes: 1
Remaining memory: 491k
```

```
C:\PRACTI~1>TLINK.EXE T_FOR,,,PCLIB06
Turbo Link Version 3.01 Copyright (c) 1987, 1990 Borland International

C:\PRACTI~1>T_FOR.EXE
```

```
DOS
BOX DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
Cuantas veces desea imprimir el mensaje(0-9)? : 9
Mensaje...
Mensaje...
Mensaje...
Mensaje...
Mensaje...
Mensaje...
Mensaje...
Mensaje...
Mensaje...

C:\PRACTI~1>s
```

```
DOS
BOX DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
Cuantas veces desea imprimir el mensaje(0-9)? : 5
Mensaje...
Mensaje...
Mensaje...
Mensaje...
Mensaje...

C:\PRACTI~1>s
```


5. WHILE_DO

```
C:\PRACTI~1>TASM WHILE_DO
Turbo Assembler Version 2.01 Copyright (c) 1988, 1990 Borland International

Assembling file:   WHILE_DO.ASM
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  491k

C:\PRACTI~1>TLINK.EXE WHILE_DO,,,PCLIB06
Turbo Link Version 3.01 Copyright (c) 1987, 1990 Borland International

C:\PRACTI~1>WHILE_DO.EXE_
```

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

Desea imprimir el mensaje? (S/N): N

C:\PRACTI~1>

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

Desea imprimir el mensaje? (S/N): S
Este es un mensaje...
Desea volver a imprimir el mensaje? (S/N): S
Este es un mensaje...
Desea volver a imprimir el mensaje? (S/N): S
Este es un mensaje...
Desea volver a imprimir el mensaje? (S/N): S
Este es un mensaje...
Desea volver a imprimir el mensaje? (S/N): S
Este es un mensaje...
Desea volver a imprimir el mensaje? (S/N): S
Este es un mensaje...
Desea volver a imprimir el mensaje? (S/N): S
Este es un mensaje...
Desea volver a imprimir el mensaje? (S/N): S
Este es un mensaje...
Desea volver a imprimir el mensaje? (S/N): N

C:\PRACTI~1>_

6. DO_WHILE

```
C:\PRACTI~1>TASM DO_WHILE
Turbo Assembler Version 2.01 Copyright (c) 1988, 1990 Borland International

Assembling file:    DO_WHILE.ASM
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  491k
```

```
C:\PRACTI~1>TLINK.EXE DO_WHILE,,,PCLIB06
Turbo Link Version 3.01 Copyright (c) 1987, 1990 Borland International

C:\PRACTI~1>DO_WHILE.EXE_
```

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

```
Este es un mensaje...
Desea volver a imprimir el mensaje? (S/N): S
Este es un mensaje...
Desea volver a imprimir el mensaje? (S/N): S
Este es un mensaje...
Desea volver a imprimir el mensaje? (S/N): S
Este es un mensaje...
Desea volver a imprimir el mensaje? (S/N): S
Este es un mensaje...
Desea volver a imprimir el mensaje? (S/N): S
Este es un mensaje...
Desea volver a imprimir el mensaje? (S/N): S
Este es un mensaje...
Desea volver a imprimir el mensaje? (S/N): S
Este es un mensaje...
Desea volver a imprimir el mensaje? (S/N): S
Este es un mensaje...
Desea volver a imprimir el mensaje? (S/N): N

C:\PRACTI~1>
```

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 estructuras, procedimientos y hasta programas más complejos.

REFERENCIAS

2 Assembly Language Programming. Cs.unm.edu. (2020). Retrieved 5 December 2020, from <https://www.cs.unm.edu/~maccabe/classes/341/labman/node2.html>.

ANEXOS

1. Hola Mundo

```

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, subroutine 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           ;

    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

```

2. IF_THEN

```

MODEL small
.STACK 100h
INCLUDE procs.inc
LOCALS
.DATA
ask db 'Presione la tecla A mayuscula: ',0
IfThen db '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 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
    JE @@IfThen      ;Si son iguales imprime un mensaje
    JMP @@EndIF      ;Si no son iguales, imprime otro mensaje
@@IfThen: mov dx,offset IfThen
    call puts
    call println     ;Imprime un salto de línea
@@EndIF:  mov ah,04ch    ; fin de programa
    mov al,0         ;
    int 21h          ;
    ENDP

println PROC ; Función para indicar un salto de línea
    ;Debido que la función modifica el registro A se hace
    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
ENDP; < -- Indica a TASM el fin del procedimiento
END

```

3. IF_ELSE

```

MODEL small
.STACK 100h
INCLUDE procs.inc
LOCALS
.DATA
ask db 'Presione la tecla A mayuscula: ',0
IfThen db 'Ha presionado la tecla A mayuscula',0
IfThenElse db 'No ha presionado la tecla A mayuscula',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
    JE @@IfThen ;Si son iguales imprime un mensaje
    JMP @@Else ;Si no son iguales, imprime otro mensaje
@@IfThen: mov dx,offset IfThen
    call puts
    call println ;Imprime un salto de linea
    JMP @@EndIF
@@Else: mov dx,offset IfThenElse
    call puts
@@EndIF: mov ah,04ch ; fin de programa
    mov al,0 ;
    int 21h ;
    ENDP

println PROC ; Funcion para indicar un salto de linea
    ;Debido que la función modifica el registro A se hace
    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
ENDP; < -- Indica a TASM el fin del procedimiento
END

```

4. 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, subroutine 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,b1
        JE @@SCASE

        mov bl,39h      ;Compara si es char '9'
        cmp al,b1
        JE @@SCASE
        JMP @@DEFAULT

@@SCASE:
        mov dx,offset scaseMessage
        call puts
        mov al,b1      ;Se copia el valor de b1 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
        ENDP

println PROC ; Funcion para indicar un salto de linea
                ;Debido que la función modifica el registro A se hace
        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
ENDP; < -- Indica a TASM el fin del procedimiento
END

```


5. FOR

```

MODEL small
.STACK 100h
INCLUDE procs.inc
LOCALS
.DATA
ask db 'Cuántas veces desea imprimir el mensaje(0-9)? : ',0
xcptn db 'Por favor, seleccionar un numero valido',0
message db 'Mensaje...',0
.CODE
;----- Insert program, subroutine 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 ;
    ENDP

println PROC ; Funcion para indicar un salto de linea
    ;Debido que la función modifica el registro A se hace
    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
ENDP; < -- Indica a TASM el fin del procedimiento
END

```

6. 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
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         ;
               ENDP

println PROC ; Funcion para indicar un salto de linea
               ;Debido que la función modifica el registro A se hace
    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
ENDP; < -- Indica a TASM el fin del procedimiento
END

```

7. 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, subroutine 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

@@StartWhile:
    mov dx,offset message
    call puts        ;Do
    call println     ;Imprime un salto de línea
    mov dx,offset ask
    call puts
    call getchar
    mov cl,al
    call println

    cmp cl,bl        ;While
    JE @@EndWhile
    JMP @@StartWhile

@@EndWhile: mov ah,04ch    ; fin de programa
            mov al,0
            int 21h
            ENDP

println PROC ; Funcion para indicar un salto de linea
            ;Debido que la función modifica el registro A se hace
            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
ENDP; < -- Indica a TASM el fin del procedimiento
END

```