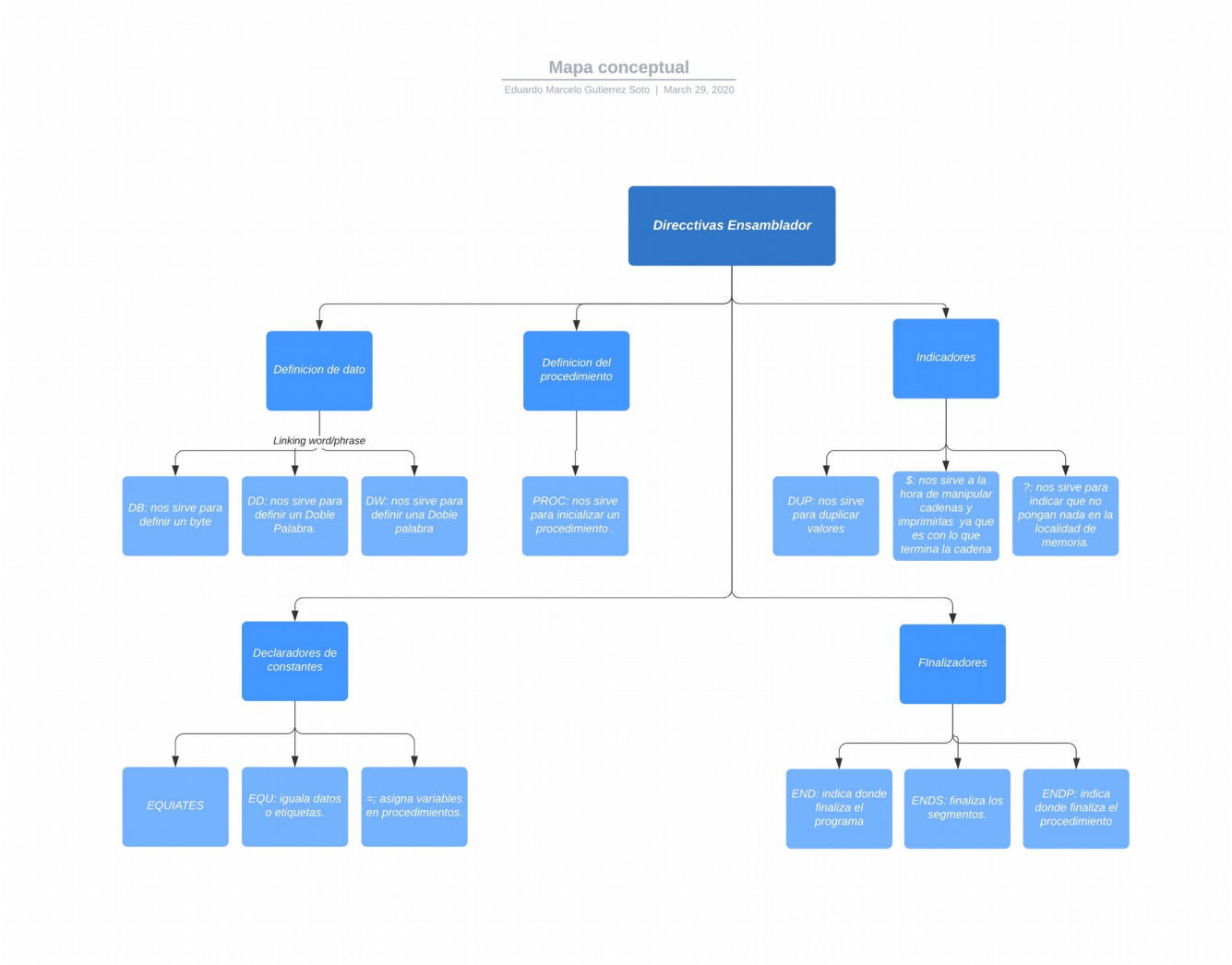
Universidad Autónoma de Baja California
Facultad de Ciencias Químicas e Ingeniería

Alumno: Eduardo Marcelo Gutiérrez Soto
Profesora: Evangelina Lara Camacho
Practica 6 Estructuras de Control de Programa

Teoría

Mapa conceptual sobre las directivas:

- DB,DW Y DD.
- DUP, ?,$.
- Equates EQU y =.
- PROC.
- END,ENDP y ENDS



Responder las siguientes preguntas

Que es un modelo de memoria? Un modelo de memoria es el que define la manera de como un programa se almacena en la memoria del sistema.

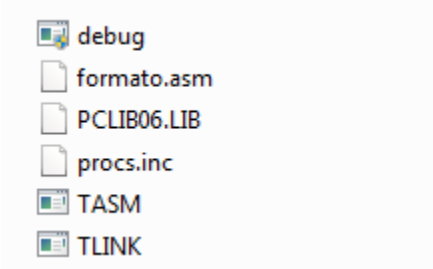
Cuales son las diferencias entre los archivos .EXE y .COM? Que los archivos .COM se ejecutan mas rapido que los archivos .EXE y los .COM son diferentes por que todos los datos y codigo se calzan en un solo segmento u se usa como origen la 100h y para la mayoría de las aplicaciones se usa el formato .EXE y el modelo de memoria small

Tabla de modelos de memoria

Modelo de memoria	Descripción
TIny	Se combinan datos y codigo en el mismo segmento, debe ser menor que 64k. Este modelo permite crear archivos .COM el cual se origina en la localidad 100h
Small	Contiene dos segmentos separados el de codigo de 64k y el de datos de 64k.
Medium	Contiene un segmento de datos y cualquier numero de segmentos de codigo.
Compact	Contiene un segmento de codigo y cualquier segmetno de datos.
Large	Permite cualquier numero de segmento de datos y de codigo.
Huge	Igual que large, pero los segmentos de datos pueden tener mas de 64k.

Desarrollo de la parte practica

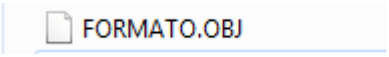
1:



2: Ensable del formato.asm

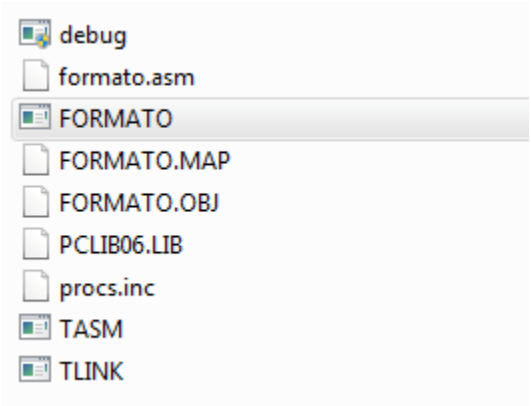
```
C:\>tasm FORMATO.ASM
Turbo Assembler  Version 2.01  Copyright (c) 1988, 1990 Borland International

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

Crecion del formato.obj

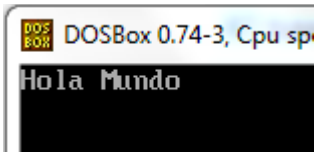


3: Encadenamiento del archivo formato.asm y creación del .EXE

```
C:\>tlink FORMATO,,,PCLIB06
Turbo Link  Version 3.01 Copyright (c) 1987, 1990 Borland International
```



4: Ejecución del programa .exe



en la ejecución podemos ver que el resultado es un hola mundo.

5: Crear un ejemplo de las siguientes sentencias de programación.

A) IF-THEN.

```
.MODEL SMALL
    .STACK 100H
    INCLUDE PROCS.INC
.DATA
    MENS DB 10,13,"PROGRAMA DE LA SENTENCIA IF-THEN QUE DETECTA EL NUMERO 3:",0
    MENS2 DB 10,13,"ES NUMERO 3",0
.CODE
    PROGRAMA PROC
    MOV AX,@DATA
    MOV DS,AX
    ;INICIALIZACION DEL SEGMENTO DE DATOS
    ;CODIGO DE UN IF-THEN
    MOV DX,OFFSET MENS      ;MENSAJE DE INICIO DEL PROGRAMA
    CALL PUTS               ;PROCEDIMINETO PARA DESPLEGAR EN PANTALLA
    CALL GETCHAR            ;PROCEDIMIENTO PARA CAPTURAR UN CARACTER
    CMP AL,51               ;COMPARACION CON EL CARACTER ESPERADO
    JZ @@IF_THEN            ;ES EL CARACTER ESPERADO BRINCA A LA ETIQUETA
    JMP @@FIN               ;NO ES EL DATO ESPERADO, SALIMOS DEL PROGRAMA

    @@IF_THEN:              ;ETIQUETA DEL IF-THEN
    MOV DX,OFFSET MENS2     ;MENSAJE DE VERIFICACION
    CALL PUTS               ;DESPLIEGUE EN PANTALLA

    @@FIN:                  ;SALIMOS DEL PROGRAMA

    ;FINALIZACION DEL PROGRAMA
    MOV AH,04CH
    MOV AL,0
    INT 21H
    ENDP
END
```

```
C:\>IF-THEN

PROGRAMA DE LA SENTENCIA IF-THEN QUE DETECTA EL NUMERO 3 Y SI NO LO ES SE SALE:3

ES NUMERO 3
C:\>_
```

B) IF-THEN-ELSE.

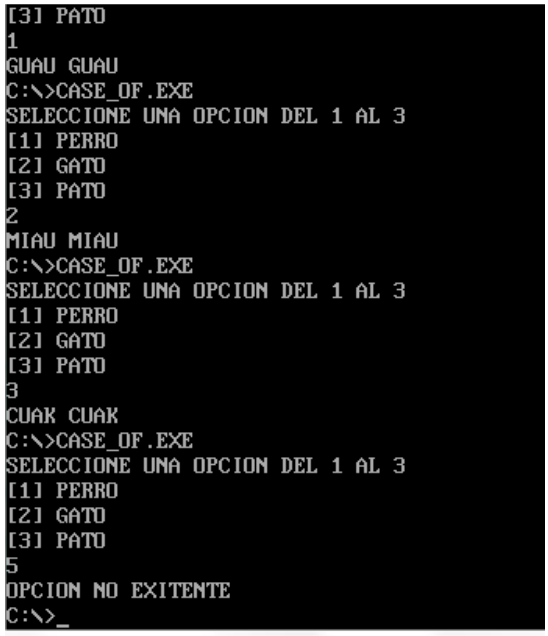
```
.MODEL SMALL
    .STACK 100H
    INCLUDE PROCS.INC
.DATA
    MENS DB 10,13,"PROGRAMA DE LA SENTENCIA IF-THEN ELSE INGRESE N#3:",0
    MENS2 DB 10,13,"ES NUMERO 3",0
    MENS3 DB 10,13,"NO ES EL NUMERO ESPERADO",0
.CODE
    PROGRAMA PROC
    MOV AX,@DATA
    MOV DS,AX
    ;INICIALIZACION DEL SEGMENTO DE DATOS
    ;CODIGO DE UN IF-THEN
    MOV DX,OFFSET MENS      ;MENSAJE DE INICIO DEL PROGRAMA
    CALL PUTS               ;PROCEDIMINETO PARA DESPLEGAR EN PANTALLA
    CALL GETCHAR            ;PROCEDIMIENTO PARA CAPTURAR UN CARACTER
    CMP AL,51               ;COMPARACION CON EL CARACTER ESPERADO
    JZ @@IF_THEN            ;ES EL CARACTER ESPERADO BRINCA A LA ETIQUETA
    ;ELSE
    MOV DX,OFFSET MENS3     ;CASO DEFAULT
    CALL PUTS
    JMP @@FIN               ;NO ES EL DATO ESPERADO, SALIMOS DEL PROGRAMA

    @@IF_THEN:              ;ETIQUETA DEL IF-THEN
    MOV DX,OFFSET MENS2     ;MENSAJE DE VERIFICACION
    CALL PUTS               ;DESPLIEGUE EN PANTALLA

    @@FIN:                  ;SALIMOS DEL PROGRAMA

    ;FINALIZACION DEL PROGRAMA
    MOV AH,04CH
    MOV AL,0
    INT 21H
    ENDP
END
```

Resultado

```
PROGRAMA DE LA SENTENCIA IF-THEN ELSE INGRESE N#3:3
ES NUMERO 3
C:\>IF-ELSE

PROGRAMA DE LA SENTENCIA IF-THEN ELSE INGRESE N#3:J
NO ES EL NUMERO ESPERADO
C:\>
```

C) CASE OF.

Resultado




```
.DATA
    MENS DB "SELECCIONE UNA OPCION DEL 1 AL 3",10,13
         DB "[1] PERRO",10,13
         DB "[2] GATO",10,13
         DB "[3] PATO",10,13,0
    MENS2 DB 10,13,"OPCION NO EXITENTE",0
    MENS3 DB 10,13,"GUAU GUAU",0
    MENS4 DB 10,13,"MIAU MIAU",0
    MENS5 DB 10,13,"CUAK CUAK",0
.CODE
    SIWTCH PROC
    MOV AX,@DATA
    MOV DS,AX

            MOV DX,OFFSET MENS
            CALL PUTS
            CALL GETCHAR    ;LLAMAMOS AL PROCEDIMIENTO PARA CAPTURAR UN VALOR
            CMP AL,49       ;CASO 1 EL NUMERO 49 REPRESENTA EL VALOR 1
            JNZ @@CASO_2    ;SI NO ES CERO VAMOS AL CASO SIGUIENTE
                            ;SI LA OPCION ES VALIDA SE EJECUTA EL CASO Y SALIMOS DEL PROGRAMA
            MOV DX,OFFSET MENS3
            CALL PUTS       ;LLAMAMOS EL PROCEDIMIENTO PUTS PARA DESPLEGAR UN MENSAJE EN PANTALLA
            JMP @@FIN       ;SI ES LA CONDICION ESPERADA SALIMOS DEL PROGRAMA

@@CASO_2:   CMP AL,50   ;CASO 2 EL NUMERO 50 REPRESENTA EL VALOR 2
            JNZ @@CASO_3    ;SI NO ES CERO VAMOS AL CASO SIGUIENTE
            MOV DX,OFFSET MENS4
            CALL PUTS
            JMP @@FIN

@@CASO_3:    CMP AL,51  ;CASO 3 EL NUMERO 51 REPRESENTA EL VALOR 3
            JNZ @@DEFAULT   ;SI NO ES CERO VAMOS AL CASO DEFAULT
            MOV DX,OFFSET MENS5
            CALL PUTS
            JMP @@FIN
@@DEFAULT:
            MOV DX,OFFSET MENS2
            CALL PUTS

@@FIN:
```

D) FOR.

```
.MODEL SMALL
    .STACK 100H
    INCLUDE PROCS.INC
.DATA
    MENS DB "PROGRAMA QUE EJECUTA UN CICLO FOR",10,13
         DB "INGRESE EL NUMERO DE REPETICIONES CON UN MAXIMO DE 9",10,13
         DB "SE IMPRIMIRA UN CARACTER N VECES",10,13,0

.CODE
    CICLO_FOR PROC
    MOV AX,@DATA
    MOV DS,AX

    MOV DX,OFFSET MENS
    CALL PUTS

    CALL GETCHAR ;NUMERO DE REPETICIONES
    MOV AH,AL    ;MOVEMOS EL NUMERO DE VECES A AH
    SUB AH,30H   ;AJUSTAMOS EL VALOR PARA TENER EL VALOR REAL
    MOV AL,33    ;MOVEMOS EL VALOR A IMPRIMIR
    ;FOR
    MOV CX,0     ;INICIALIZAMOS NUESTRO CONTADOR EN CERO
@@FOR:
    CMP CL,AH    ;COMPARAMOS AH CON CL SI ES IGUAL
    JAE @@FIN
    CALL PUTCHAR;IMPRIMIMOS EL CARACTER
    INC CL       ;INCREMENTAMOS CL EN 1
    JMP @@FOR
@@FIN:

    MOV AH,04CH
    MOV AL,0
    INT 21H
    ENDP
END
```

Resultado

```
C:\>FOR
PROGRAMA QUE EJECUTA UN CICLO FOR
INGRESE EL NUMERO DE REPETICIONES CON UN MAXIMO DE 9
SE IMPRIMIRA UN CARACTER N VECES
3!!!
C:\>
```

E) WHILE-DO.

WHILE-DO.asm | DO-WHILE.asm

```
;creacion de un programa que ejecute el ciclo while-do en ensamblador
.MODEL SMALL
    .STACK 100H
    INCLUDE PROCS.INC
.DATA
    MENS DB 10,13,"CICLO WHILE QUE SE EJECUTA MIENTRAS EL NUMERO SEA 4:",0
    MENS2 DB 10,13,"ES NUMERO 4",0
.CODE
    PROGRAMA PROC
    MOV AX,@DATA
    MOV DS,AX
    @@WHILE:
    MOV DX,OFFSET MENS  ;MENSAJE DE INICIO
    CALL PUTS           ;PROCEDIMIENTO PARA DESPLEGAR EL MENSAJE
    CALL GETCHAR        ;CAPTURAMOS EL VALOR 4
    CMP AL,52           ;SI EL NUMERO ES 4 LA CONDICION SE CUMPLE Y LUEGO SE EJECUTA
    JNZ @@FIN           ;SI NO ES EL NUMERO ESPERADO SE SALE DEL CICLO
    MOV DX,OFFSET MENS2 ;MENSAJE QUE VERIFICA QUE ENTRO AL WHILE
    CALL PUTS           ;PROCEDIMIENTO DE PUTS QUE DESPLIEGA UNA CADENA DE TEXTO
    JMP @@WHILE         ;SI EL NUMERO ES EL ESPERADO SE EJECUTA DE NUEVO EL CICLO
    @@FIN:
    MOV AH,04CH
    MOV AL,0
    INT 21H
    ENDP
END
```

Resultado

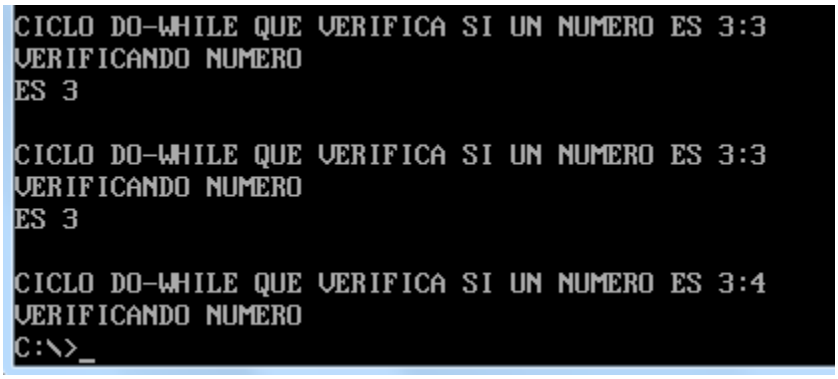
```
C:\>while-do

CICLO WHILE QUE SE EJECUTA MIENTRAS EL NUMERO SEA 4:4
ES NUMERO 4
CICLO WHILE QUE SE EJECUTA MIENTRAS EL NUMERO SEA 4:4
ES NUMERO 4
CICLO WHILE QUE SE EJECUTA MIENTRAS EL NUMERO SEA 4:4
ES NUMERO 4
CICLO WHILE QUE SE EJECUTA MIENTRAS EL NUMERO SEA 4:6
C:\>
```

F) DO-WHILE.

WHILE-DO.asm | DO-WHILE.asm

```
;creacion de un programa que realiza el ciclo do while
.MODEL SMALL
    .STACK 100H
    INCLUDE PROCS.INC
.DATA
    MENS DB 10,13,"CICLO DO-WHILE QUE VERIFICA SI UN NUMERO ES 3:",0
    MENS2 DB 10,13,"ES 3",10,13,0
    MENS3 DB 10,13,"VERIFICANDO NUMERO",0,13,0
.CODE
    PROGRAMA PROC
    MOV AX,@DATA
    MOV DS,AX

    @@SALTO:            ;ETIQUETA PARA INICIAR EL CICLO OTRA VEZ
    MOV DX,OFFSET MENS  ;MENSAJE DE INICIO DEL PROGRAMA
    CALL PUTS           ;PROCEDIMIENTO QUE IMPRIMIE EN PANTALLA
    CALL GETCHAR        ;PROCEDIMIENTO QUE CAPTURA UN CARACTER
    ;INICIO DEL DO-WHILE
    MOV DX,OFFSET MENS3 ;MENSAJE DE VERIFICACION DEL CICLO
    CALL PUTS           ;PROCEDIMIENTO QUE LO IMPRIME
    CMP AL,51           ;COMPARACION DE LA CONDICION
    JNE @@FIN           ;SALTO DE QUE SI EL NUMERO NO ES EL ESPERADO SE TERMINE
    MOV DX,OFFSET MENS2 ;MENSAJE DE QUE SE REALIZO EL CICLO CORRECTAMENTE
    CALL PUTS           ;LO DESPLEGAMOS EN PANTALLA
    JMP @@SALTO         ;VOLVEMOS A INICIAR EL CICLO
    @@FIN:


    MOV AH,04CH
    MOV AL,0
    INT 21H
    ENDP
END
```

Resultado



Ejercicio de metales:

```
C:\>EJE2

INGRESE VALORES PARA LAS SIGUIENTES CXARACTERISTICAS DEL METAL FICTICIO

DUREZA:
55

CONTENIDO DE CARBON:
8

MALEABILIDAD DEL METAL:
99

GRADO 10
C:\>
```

El código es demasiado largo para ponerlo en el reporte pero se hizo uso de un nuevo procedimiento para la creación de cadenas y la convercion de cadenas a valor entero Atoi.

Ejercicio del ciclo

```
C:\>FOR_1
1
22
333
4444
55555
666666
7777777
88888888
999999999
```

Código

WHILE-DO.asm | DO-WHILE.asm | eje2.asm | FOR_1.ASM

```
1   .MODEL SMALL
2   .STACK 100H
3       INCLUDE PROCS.INC
4       LOCALS
5   .DATA
6           MENS DB 10,13,"",0
7   .CODE
8
9   PROC MAIN
10          MOV AX,@DATA
11          MOV DS,AX
12                              ; FOR(I=0; I<10; I++)
13          MOV CL,1            ; I=1;
14  @@F_I:  CMP CL,9            ;   I<10;
15          JA @@END_FI         ;
16
17          MOV CH,1
18  @@F_J:  CMP CH,CL
19          JA @@END_J
20
21          MOV AL,30H
22          ADD AL,CL
23          CALL PUTCHAR
24          INC CH
25          JMP @@F_J
26          @@END_J:
27
28          MOV DX,OFFSET MENS
29          CALL PUTS
30
31          INC CL
32          JMP @@F_I
33          @@END_FI:
34
35          MOV AH,04CH
36          MOV AL,0H
37          INT 21H
38  ENDP
39
40  END
```

Assembly language source file

Conclusión

En el desarrollo de la practica hicimos uso de las sentencia de C pero en su versión en ensamblador las que son los ciclos, los cuales fueron el ciclo FOR, WHILE, DO-WHILE, CASE y condicionales como IF e ELSE IF.

Dificultades

El ultimo ejercicio no lo logre hacer a la perfeccion ya que en C tenia varios problemas de como implementar lo que hice a su versión en ensamblador.