

LABORATORIO 2b

Para el presente laboratorio se utilizara se requiere Contar con Turbo Assembler y la utilidad de DOS debug.

Parte I:

1. Copia el siguiente programa en cualquier editor y llámalo primer.asm:

```
STACK SEGMENT PARA STACK 'STACK'  
    DB 64 DUP ('STACK  ')  
STACK ENDS  
CSEG SEGMENT PARA PUBLIC 'CODE'  
START  PROC FAR  
    ASSUME CS:CSEG, SS:STACK  
  
    SUB AX,AX  
    MOV AX,18D  
    SUB AX,18D  
  
START  ENDP  
CSEG  ENDS  
END START
```

2. Compilalo usando Turbo assembler:

```
C:\Dirtra>TASM primer.asm
```

Con lo que obtendrás el archivo primer.obj, verifica haciendo:

```
C:\Dirtra> DIR *.obj
```

3. Luego enlazalo, para obtener un ejecutable:

```
C:\Dirtra>TLINK primer
```

Con lo que obtendrás el archivo primer.exe, verifica haciendo:

```
C:\Dirtra> DIR *.exe
```

4. Luego emplea la utilidad debug de la siguiente forma:

```
C:\Dirtra> DEBUG primer.exe
```

- Veras luego un guión, esto indica que debug está listo para trabajar, si empleas el comando "?" verás la ayuda de debug.
- Emplea el comando "u" (desensamblar) que observas?.

```
-u  
16F3:0000 2BC0      SUB  AX,AX  
16F3:0002 B81200      MOV  AX,0012  
16F3:0005 2D1200      SUB  AX,0012
```

- Luego ejecuta el comando "t" (trace):

```
-t  
  
AX=0000 BX=0000 CX=0208 DX=0000 SP=0200 BP=0000 SI=0000 DI=0000  
DS=16C3 ES=16C3 SS=16D3 CS=16F3 IP=0002 NV UP EI PL ZR NA PE NC  
16F3:0002 B81200      MOV  AX,0012
```

- Con lo anterior has ejecutado la primera instrucción (SUB AX;AX), y se observa el estado actual de los registros y la siguiente instrucción.
- Sgue ejecutando hasta la tercera instrucción.
- Ahora emplea el comando "d" (dump):

```
-d
16F3:0000 2B C0 B8 12 00 2D 12 00-1F 5D 4D CA 02 00 B8 CF +....-...]M.....
16F3:0010 20 45 55 8B EC 1E 8E D8-8B 46 06 1F 5D 4D CA 02 EU.....F..]M..
16F3:0020 00 B8 CF 20 45 55 8B EC-1E 8E D8 83 EC 06 56 57 ... EU.....VW
```

- Que indica lo anterior (16F3:0000)? Y los números exadecimales?.
- A continuación has lo siguiente:

```
-d 16d3:0000
16D3:0000 53 54 41 43 4B 20 20 20-53 54 41 43 4B 20 20 20 STACK STACK
16D3:0010 53 54 41 43 4B 20 20 20-53 54 41 43 4B 20 20 20 STACK STACK
16D3:0020 53 54 41 43 4B 20 20 20-53 54 41 43 4B 20 20 20 STACK STACK
16D3:0030 53 54 41 43 4B 20 20 20-53 54 41 43 4B 20 20 20 STACK STACK
16D3:0040 53 54 41 43 4B 20 20 20-53 54 41 43 4B 20 20 20 STACK STACK
16D3:0050 53 54 41 43 4B 20 20 20-53 54 41 43 4B 20 20 20 STACK STACK
16D3:0060 53 54 41 43 4B 20 20 20-53 54 41 43 4B 20 20 20 STACK STACK
16D3:0070 53 54 41 43 4B 20 20 20-53 54 41 43 4B 20 20 20 STACK STACK
```

- Que se observa?. El listado corresponde a la pila, si no lo obtienes, prueba con la dirección SS.
- Existe segmento de datos?

Parte II:

1. Compila el programa primer.asm usando Turbo assembler, pero con la siguiente directiva:

```
C:\Dirtra>TASM /Zi primer.asm
```

Luego:

```
C:\Dirtra>TLINK /v firasm
```

2. Ahora emplearemos Turbo debugger, que es una herramienta de depuración más completa que debug:

```
C:\Dirtra>TD primer.exe
```

The screenshot shows the Turbo Debugger interface. The main window displays assembly code for the 'firasm' module. The code starts at address 0000 with instructions: SUB AX, AX; MOV AX, 18D; SUB AX, 18D; add [bx+sil], al; add [bx+sil], al; add [bx+sil], al; sti; push dx. The registers window on the right shows: ax=0000, bx=0000, cx=0000, dx=0000, si=0000, di=0000, bp=0000, sp=0200, ds=153C, es=153C, ss=154C. The status bar at the bottom shows: F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu.

- El modo de trabajo en turbo debugger es el mismo que Turbo C o Borland C++, prueba las diversas opciones y trata de familiarizarte con el entorno
- Prueba realizar un trace (ejecución paso a paso). Emplea breakpoints, y muestra como cambian los diversos segmentos.
- Finalmente realiza la codificación de las instrucciones a lenguaje máquina. Sin usar el depurador luego comprueba tus resultados.

Parte III Modos de direccionamiento.

1. Utiliza nuevamente un editor de texto para ingresar el siguiente programa DIRS.ASM:

```
%TITLE "PROGRAMA DE DEMOSTRACION DE MODOS DE DIRECCIONAMIENTO
STACK SEGMENT PARA STACK 'STACK'
    DB 64 DUP ('STACK ')
STACK ENDS

DATA SEGMENT PARA PUBLIC 'DATA'
    DDDD DW 0
    DDDW DW 300
    DDDX DW 200
    DDDY DW 150
    DDDZ DW 125
    DDDQ DW 100
    DDDR DW 80
    DDDS DW 70
    DDDJ DW 60
    DDDU DW 50
DATA ENDS

CSEG SEGMENT PARA PUBLIC 'CODE'
START PROC FAR
    ASSUME CS:CSEG, DS:DATA, SS:STACK

    PUSH DS
    SUB AX,AX
    PUSH AX

    MOV AX,SEG DATA ; localiza la dir de Seg data
    MOV DS,AX ; Carga en DS la dir de segmento

    MOV AX,DDDW ; direccionamiento directo

    MOV BX, OFFSET DDDX ; Direccionamiento indirecto de registro
    MOV AX,[BX]

    MOV AX, [BX+2] ; Direccionamiento relativo a base

    MOV SI, 2 ; Direccionamiento indexado directo
    MOV AX,DDDZ[SI] ;

    MOV BX, OFFSET DDDW ; Direccionamiento indexado de base
    MOV SI,8
    MOV AX, [BX] [SI+2]

    RET

START ENDP
CSEG ENDS
END START
```

2. Una vez realizado, compila y enlaza con turbo assembler, puedes emplear debug o turbo debugger para la siguiente fase, si pretendes usar TD no olvides las directivas /Zi y /v.

```
E:\doc2003\curso leng bajo nivel\fuentes>debug dirs.exe
-t

AX=0000 BX=0000 CX=05BC DX=0000 SP=01FE BP=0000 SI=0000 DI=0000
DS=16C3 ES=16C3 SS=16D3 CS=16F5 IP=0001 NV UP EI PL NZ NA PO NC
16F5:0001 2BC0      SUB  AX,AX
-t

AX=0000 BX=0000 CX=05BC DX=0000 SP=01FE BP=0000 SI=0000 DI=0000
DS=16C3 ES=16C3 SS=16D3 CS=16F5 IP=0003 NV UP EI PL ZR NA PE NC
16F5:0003 50        PUSH AX
-t

AX=0000 BX=0000 CX=05BC DX=0000 SP=01FC BP=0000 SI=0000 DI=0000
DS=16C3 ES=16C3 SS=16D3 CS=16F5 IP=0004 NV UP EI PL ZR NA PE NC
16F5:0004 B8F316     MOV  AX,16F3
-d 16c3:0000
16C3:0000 CD 20 00 A0 00 9A F0 FE-1D F0 4F 03 17 10 8A 03  . .....O.....
16C3:0010 17 10 17 03 17 10 06 10-01 01 01 00 02 FF FF FF  .....
16C3:0020 FF FF FF FF FF FF FF FF-FF FF FF FF A3 16 64 3E  .....d>
16C3:0030 17 10 14 00 18 00 C3 16-FF FF FF FF 00 00 00 00  .....
16C3:0040 07 0A 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
16C3:0050 CD 21 CB 00 00 00 00 00-00 00 00 00 00 20 20 20  !!.....
16C3:0060 20 20 20 20 20 20 20 20-00 00 00 00 00 20 20 20  ....
16C3:0070 20 20 20 20 20 20 20 20-00 00 00 00 00 00 00 00  .....
-d 16f3:0000
16F3:0000 00 00 2C 01 C8 00 96 00-7D 00 64 00 50 00 46 00  ..,.....}.d.P.F.
16F3:0010 3C 00 32 00 00 00 00 00-00 00 00 00 00 00 00 00  <.2.....
16F3:0020 1E 2B C0 50 B8 F3 16 8E-D8 A1 02 00 BB 04 00 8B  .+.P.....
16F3:0030 07 8B 47 02 BE 02 00 8B-84 08 00 BB 02 00 BE 08  ..G.....
16F3:0040 00 8B 40 02 CB 00 00 00 00-00 00 00 00 00 00 00 00  ..@.....
16F3:0050 FB 52 03 04 48 00 00 00-0D 00 00 00 1E 00 00 00  .R..H.....
16F3:0060 00 00 00 00 0B 00 00 00-00 00 00 00 01 00 00 00  .....
16F3:0070 0B 00 00 00 00 00 00 00-0F 00 00 00 01 00 00 00  .....

-t
AX=16F3 BX=0000 CX=05BC DX=0000 SP=01FC BP=0000 SI=0000 DI=0000
DS=16C3 ES=16C3 SS=16D3 CS=16F5 IP=0007 NV UP EI PL ZR NA PE NC
16F5:0007 8ED8      MOV  DS,AX
-t
AX=16F3 BX=0000 CX=05BC DX=0000 SP=01FC BP=0000 SI=0000 DI=0000
DS=16F3 ES=16C3 SS=16D3 CS=16F5 IP=0009 NV UP EI PL ZR NA PE NC
16F5:0009 A10200     MOV  AX,[0002]          DS:0002=012C
-t
AX=012C BX=0000 CX=05BC DX=0000 SP=01FC BP=0000 SI=0000 DI=0000
DS=16F3 ES=16C3 SS=16D3 CS=16F5 IP=000C NV UP EI PL ZR NA PE NC
16F5:000C BB0400     MOV  BX,0004
-t
AX=012C BX=0004 CX=05BC DX=0000 SP=01FC BP=0000 SI=0000 DI=0000
DS=16F3 ES=16C3 SS=16D3 CS=16F5 IP=000F NV UP EI PL ZR NA PE NC
16F5:000F 8B07      MOV  AX,[BX]          DS:0004=00C8
```

- Analiza la conversión del código fuente *.asm hacia el proceso de tracing, y estudia detalladamente como funciona cada modo de direccionamiento, asegúrate de entenderlos todos, finaliza al encontrar la instrucción RET.
- Que indican los dumps que se realizaron?.