Practica 8
Cargador de DOS

Introducción

La función del cargador es copiar de disco a RAM la imagen de un programa y realizar las últimas reloccalizaciones antes de iniciar la ejecución del programa. Grosso modo podemos decir que el cargador ejecuta la siguiente secuencia de pasos: *i*) Lee el encabezado del ejecutable para verificar que el archivo a cargar sea realmente un ejecutable, *ii*) Determina el tamaño de la imagen a cargar y solicita esa cantidad de memoria al SO, *iii*) Copia la imagen del disco a la RAM (la carga propiamente dicha), *iv*) Ejecuta las relocalizaciones necesarias para ajustar el código a la dirección asignada y v) Transfiere el control al inicio del programa. El cargador de DOS, que es el que usaremos en esta práctica, es un cargador relocalizante que maneja relocalización estática.

Actividades

Para esta práctica vamos a tomar como ejemplo el siguiente programa en C:

```
/* Archivo: tams.c */
#include<stdio.h>
int main(void)
{
      printf("Tamanho de los tipos basicos.\n");
      printf("Tipo char = %d bytes\n",sizeof(char));
      printf("Tipo entero corto  = %d bytes\n",sizeof(short));
      printf("Tipo entero = %d\n",sizeof(int));
      printf("Tipo entero largo = %d bytes\n",sizeof(long));
      printf("Tipo flotante simple = %d\n",sizeof(float));
      printf("Tipo flotante doble = %d\n",sizeof(double));
      return 0;
}
```

Después de compilarlo y ligarlo obtenemos la interpretación de su encabezado (Despliegue 1) y el vaciado del archivo (Despliegue 2):

```
Turbo Dump  Version 3.1 Copyright (c) 1988, 1992 Borland International
                    Display of File TAMS.EXE

DOS File Size                                      1ACAh  (  6858. )
Load Image Size                                    18CAh  (  6346. )
Relocation Table entry count                       0001h  (     1. )
Relocation Table address                           003Eh  (    62. )
Size of header record      (in paragraphs)         0020h  (    32. )
Minimum Memory Requirement (in paragraphs)         000Dh  (    13. )
Maximum Memory Requirement (in paragraphs)         FFFFh  ( 65535. )
File load checksum                                 0000h  (     0. )
Overlay Number                                     0000h  (     0. )

Borland TLINK Version 5.00

Initial Stack Segment  (SS:SP)                 0191:0080
Program Entry Point    (CS:IP)                 0000:0000

Relocation Locations (1 Entry)

    0000:0001
```

Despliegue 1: Interpretación de la cabecera del archivo ejecutable obtenido con `tdump tams.exe`

```
Turbo Dump  Version 3.1 Copyright (c) 1988, 1992 Borland International
                    Display of File TAMS.EXE

000000: 4D 5A CA 00 0E 00 01 00  20 00 0D 00 FF FF 91 01 MZ...... .......
000010: 80 00 00 00 00 00 00 00  3E 00 00 00 01 00 FB 50 ........>......P
000020: 6A 72 00 00 00 00 00 00  00 00 00 00 00 00 00 00 jr..............
000030: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 01 00 ................
000040: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
000050: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
000060: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
000070: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
000080: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
000090: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
0000A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
0000B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
0000C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
0000D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
0000E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
0000F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
000100: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
      :           :                :                       :
      :           :                :                       :
000200: BA 4E 01 2E 89 16 BE 02  B4 30 CD 21 8B 2E 02 00 .N.......0.!....
000210: 8B 1E 2C 00 8E DA A3 92  00 8C 06 90 00 89 1E 8C ..,.............
000220: 00 89 2E A8 00 E8 84 01  A1 8C 00 8E C0 33 C0 8B .............3..
000230: D8 8B F8 B9 FF 7F FC F2  AE E3 61 43 26 38 05 75 ..........aC&8.u
000240: F6 80 CD 80 F7 D9 89 0E  8A 00 B9 01 00 D3 E3 83 ................
000250: C3 08 83 E3 F8 89 1E 8E  00 8C DA 2B EA 8B 3E 34 ...........+..>4
000260: 03 81 FF 00 02 73 07 BF  00 02 89 3E 34 03 81 C7 .....s.....>4...
000270: 2C 04 72 28 03 3E D6 02  72 22 B1 04 D3 EF 47 3B ,.r(.>..r"....G;
000280: EF 72 19 83 3E 34 03 00  74 07 83 3E D6 02 00 75 .r..>4..t..>...u
000290: 0E BF 00 10 3B EF 77 07  8B FD EB 03 E9 0A 02 8B ....;.w.........
0002A0: DF 03 DA 89 1E A0 00 89  1E A4 00 A1 90 00 2B D8 ..............+.
0002B0: 8E C0 B4 4A 57 CD 21 5F  D3 E7 FA 8E D2 8B E7 FB ...JW.!_........
0002C0: 33 C0 2E 8E 06 BE 02 BF  EA 03 B9 2C 04 2B CF FC 3..........,.+..
0002D0: F3 AA 83 3E AC 02 14 76  47 80 3E 92 00 03 72 40 ...>...vG.>...r@
0002E0: 77 07 80 3E 93 00 1E 72  37 B8 01 58 BB 02 00 CD w..>...r7..X....
0002F0: 21 72 2A B4 67 8B 1E AC  02 CD 21 72 20 B4 48 BB !r*.g.....!r .H.
      :           :                :                       :
      :           :                :                       :
0016D0: 59 4F 83 C6 10 0B FF 75  EC 5F 5E C3 00 00 00 00 YO.....u._^.....
0016E0: 00 00 00 00 42 6F 72 6C  61 6E 64 20 43 2B 2B 20 ....Borland C++
0016F0: 2D 20 43 6F 70 79 72 69  67 68 74 20 31 39 39 31 - Copyright 1991
001700: 20 42 6F 72 6C 61 6E 64  20 49 6E 74 6C 2E 00 4E  Borland Intl..N
001710: 75 6C 6C 20 70 6F 69 6E  74 65 72 20 61 73 73 69 ull pointer assi
001720: 67 6E 6D 65 6E 74 0D 0A  44 69 76 69 64 65 20 65 gnment..Divide e
001730: 72 72 6F 72 0D 0A 41 62  6E 6F 72 6D 61 6C 20 70 rror..Abnormal p
001740: 72 6F 67 72 61 6D 20 74  65 72 6D 69 6E 61 74 69 rogram terminati
001750: 6F 6E 0D 0A 00 00 00 00  00 00 00 00 00 00 00 00 on..............
001760: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
001770: 00 00 00 00 00 00 00 00  00 00 2C 04 2C 04 00 00 ..........,.,...
001780: 00 00 00 00 00 00 00 00  00 00 54 61 6D 61 6E 68 ..........Tamanh
001790: 6F 20 64 65 20 6C 6F 73  20 74 69 70 6F 73 20 62 o de los tipos b
0017A0: 61 73 69 63 6F 73 2E 0A  00 54 69 70 6F 20 63 68 asicos...Tipo ch
0017B0: 61 72 20 3D 20 25 64 20  62 79 74 65 73 0A 00 54 ar = %d bytes..T
0017C0: 69 70 6F 20 65 6E 74 65  72 6F 20 63 6F 72 74 6F ipo entero corto
0017D0: 20 20 3D 20 25 64 20 62  79 74 65 73 0A 00 54 69   = %d bytes..Ti
0017E0: 70 6F 20 65 6E 74 65 72  6F 20 3D 20 25 64 0A 00 po entero = %d..
0017F0: 54 69 70 6F 20 65 6E 74  65 72 6F 20 6C 61 72 67 Tipo entero larg
001800: 6F 20 3D 20 25 64 20 62  79 74 65 73 0A 00 54 69 o = %d bytes..Ti
001810: 70 6F 20 66 6C 6F 74 61  6E 74 65 20 73 69 6D 70 po flotante simp
```

```
001820: 6C 65 20 3D 20 25 64 0A  00 54 69 70 6F 20 66 6C le = %d..Tipo fl
001830: 6F 74 61 6E 74 65 20 64  6F 62 6C 65 20 3D 20 25 otante doble = %
001840: 64 0A 00 00 00 00 45 03  45 03 45 03 00 00 09 02 d.....E.E.E.....
001850: 00 00 00 00 00 00 00 00  00 00 6C 01 00 00 0A 02 ..........l.....
001860: 01 00 00 00 00 00 00 00  00 00 7C 01 00 00 02 02 ..........|.....
001870: 02 00 00 00 00 00 00 00  00 00 8C 01 00 00 43 02 ..............C.
      :           :                :                       :
      :           :                :                       :
001A50: 14 14 14 14 14 0D 14 14  14 14 14 14 14 14 14 14 ................
001A60: 10 0A 0F 0F 0F 08 0A 14  14 06 14 12 0B 0E 14 14 ................
001A70: 11 14 0C 14 14 0D 14 14  14 14 14 14 14 00 70 72 ..............pr
001A80: 69 6E 74 20 73 63 61 6E  66 20 3A 20 66 6C 6F 61 int scanf : floa
001A90: 74 69 6E 67 20 70 6F 69  6E 74 20 66 6F 72 6D 61 ting point forma
001AA0: 74 73 20 6E 6F 74 20 6C  69 6E 6B 65 64 0D 0A 00 ts not linked...
001AB0: 00 00 00 00 00 00 0D 00  00 00 00 00 C4 0A C9 0A ................
001AC0: C9 0A C9 0A 00 02 ED 04  00 00 00 00 00 00 00 00 ................
```

Despliegue 2: Vaciado del archivo ejecutable obtenido con `tdump -h tams.exe`

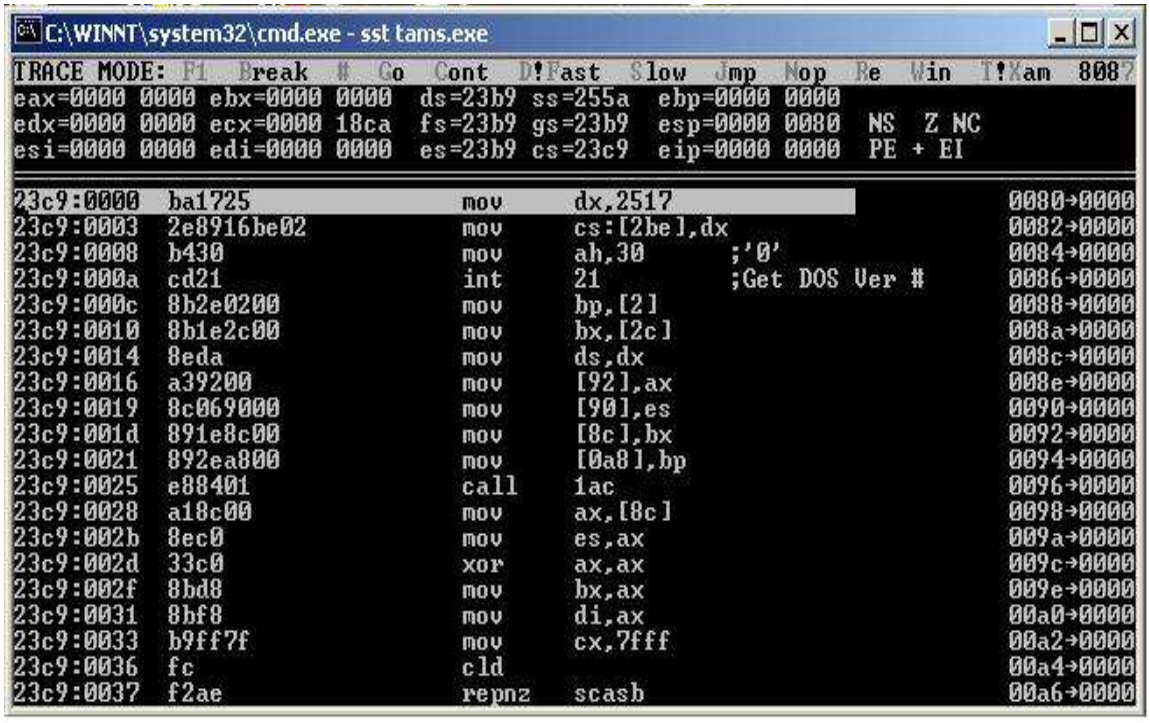
De la interpretación del encabezado (Listado 1) vemos que este ejecutable sólo necesita una relocalización y que ésta se ubica en `0000:0001`. Esta dirección es relativa a la imagen del archivo que empieza al final del encabezado, que ocupa 20 párrafos, esto es 20 bloques de 16 bytes, por lo que la ubicalización de la relocalización dentro del ejecutable es `0020:0001` que corresponde al siguiente renglón del vaciado (Listado 2):

```
000200: BA 4E 01 2E 89 16 BE 02  B4 30 CD 21 8B 2E 02 00 .N.......0.!....
```

Usando el programa `debug` puedes comprobar que la instrucción en lenguaje máquina `BA4E01` corresponde es la instrucción `mov dx,014e`.

Para comprobar la relocalización que realiza el cargador vamos a utilizar el depurador SST.

Para esto, desde una ventana de DOS tecleamos `sst tams1.exe` y en la pantalla que aparece tecleamos `t`, para entrar al modo de trazado. Esto despliega algo parecido a lo siguiente en una parte de la pantalla:



que nos indica que, en esta ocasión, la dirección de inicio del programa es `23c9:0000` y que la primera instrucción ya relocalizada es `mov dx,2517`. Esto es, la instrucción `mov dx,014e` fue modificada por el cargador para ser ahora `mov dx,2517.` La diferencia de `2517h` con `014e` es `23C9h`, que resulta ser el segmento de la dirección donde se cargó el programa. Esto nos muestra que el cargador de DOS sólo necesita saber la dirección de carga del programa para poder parchar la imagen ejecutable y relocalizar estáticamente el programa en memoria para prepararlo para su ejecución.

Actividades

1.- Usando los archivos fuente de la práctica de ligadores (archivos `a.c` y `m.c`): *i*) genera el ejecutable `a.exe` tecleando:

```
tcc a.c m.c<enter>
```

ii) Usando `tdump` determina las relocalizaciones que debe realizar el cargador al ejecutar el programa `a.exe` y reporta el resultado de la(s) relocalización(es) que requiere este ejecutable:

Relocalización en .EXE	Bytes en el archivo ejecutable	Corresponde a la instrucción*
_____:_____	_____	_____
_____:_____	_____	_____
_____:_____	_____	_____

*Nota: Usa `debug` para desensamblar estos bytes.

2.- Usando el depurador `sst`, carga y despliega el código de `a.exe` tecleando:

```
sst a.exe
```

y obtén los datos siguientes:
Dirección de carga del programa: _____:_____
Primera instrucción en archivo ejecutable: _____
Primera instrucción del programa en memoria: _____
Diferencia entre los datos de las instrucciones: _____

3.- Comentarios y conclusiones

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____