DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico III

23 de octubre de 2015 — Organizacion del computador II

Te voy a dar un Byte

Integrante	LU	Correo electrónico
Gonzalez Benitez, Albertito Juan	324/14	gonzalezjuan.ab@gmail.com
Lew, Axel Ariel	225/14	axel.lew@hotmail.com
Noli Villar, Juan Ignacio	174/14	juaninv@outlook.com

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

Índice

1. Introducción:

En este trabajo practico procederemos a realizar un sistema operativo que pueda correr un juego.

2. Ejercicio 1:

2.1. Introducción:

En este ejercicio vamos a realizar la Tabla de Descriptores Globales (GDT). Se pide que realizemos lo siguiente :

a) Que la tabla gdt tenga 4 segmentos, dos para código de nivel 0 y 3; y otros dos para datos de nivel 0 y 3. Estos segmentos deben direccionar los primeros 500MB de memoria. Por último se pide no usar las primeras siete posiciones de la gdt, ya que se consideran utilizadas.

b) Pasar a modo protegido y setear la pila del kernel en la direccion 0x27000.

c) Agregar a la gdt un segmento adicional y escribir una rutina qe utilice este nuevo segmento para pintar la esquina superior izquierda de la pantalla.

d) Limpiar la pantalla y pintar el área del mapa (sugerido el color gris) junto con las barras inferiores para los jugadores.

2.2. Ítem a: Setear la GDT

Para este ítem completamos el archivo gdt.c proporcionado por la catedra. En el mismo la GDT es representada mediante un array de 30 posiciones. Cada posicion tiene la siguiente estructura.

```
[GDT_IDX_NULL_DESC] = (gdt_entry) {
    (unsigned short)   0x0000,      /* limit[0:15]  */
    (unsigned short)   0x0000,      /* base[0:15]   */
    (unsigned char)    0x00,        /* base[23:16]  */
    (unsigned char)    0x00,        /* type         */
    (unsigned char)    0x00,        /* s            */
    (unsigned char)    0x00,        /* dpl          */
    (unsigned char)    0x00,        /* p            */
    (unsigned char)    0x00,        /* limit[16:19] */
    (unsigned char)    0x00,        /* avl          */
    (unsigned char)    0x00,        /* l            */
    (unsigned char)    0x00,        /* db           */
    (unsigned char)    0x00,        /* g            */
    (unsigned char)    0x00,        /* base[31:24]  */
},
```

Figura 1: Este descriptor corresponde a la primer entrada de la gdt

Como la primer posicion de la tabla GDT debe ser corresponder a una entrada nula, llenamos la primer posicion como muestra la imagen debajo.

```
[GDT_IDX_NULL_DESC] = (gdt_entry) {
    (unsigned short)    0x0000,       /* limit[0:15]  */
    (unsigned short)    0x0000,       /* base[0:15]   */
    (unsigned char)     0x00,         /* base[23:16]  */
    (unsigned char)     0x00,         /* type         */
    (unsigned char)     0x00,         /* s            */
    (unsigned char)     0x00,         /* dpl          */
    (unsigned char)     0x00,         /* p            */
    (unsigned char)     0x00,         /* limit[16:19] */
    (unsigned char)     0x00,         /* avl          */
    (unsigned char)     0x00,         /* l            */
    (unsigned char)     0x00,         /* db           */
    (unsigned char)     0x00,         /* g            */
    (unsigned char)     0x00,         /* base[31:24]  */
},
```

Figura 2: Este descriptor corresponde a la primer entrada de la gdt

Luego, creamos los 4 segmentos que se piden a partir de la posicion 8 de la GDT, ya que por enunciado, no se deben tocar las primeras 7 posiciones de la table de descriptores. Mostramos en las imagenes de abajo como creamos un descriptor de datos y otro de codigos.

```
[GDT_IDX_NULL_DESC+8] = (gdt_entry) {
    (unsigned short)    0xF400,       /* limit[0:15]  */
    (unsigned short)    0x0000,       /* base[0:15]   */
    (unsigned char)     0x00,         /* base[23:16]  */
    (unsigned char)     0x0A,         /* type         */
    (unsigned char)     0x01,         /* s            */
    (unsigned char)     0x00,         /* dpl          */
    (unsigned char)     0x01,         /* p            */
    (unsigned char)     0x01,         /* limit[16:19] */
    (unsigned char)     0x00,         /* avl          */
    (unsigned char)     0x00,         /* l            */
    (unsigned char)     0x01,         /* db           */
    (unsigned char)     0x01,         /* g            */
    (unsigned char)     0x00,         /* base[31:24]  */
},
```

Figura 3: Este descriptor corresponde al segmento de datos de nivel 0

```
[GDT_IDX_NULL_DESC+9] = (gdt_entry) {
    (unsigned short)    0xF400,       /* limit[0:15]  */
    (unsigned short)    0x0000,       /* base[0:15]   */
    (unsigned char)     0x00,         /* base[23:16]  */
    (unsigned char)     0x02,         /* type         */
    (unsigned char)     0x01,         /* s            */
    (unsigned char)     0x00,         /* dpl          */
    (unsigned char)     0x01,         /* p            */
    (unsigned char)     0x01,         /* limit[16:19] */
    (unsigned char)     0x00,         /* avl          */
    (unsigned char)     0x00,         /* l            */
    (unsigned char)     0x01,         /* db           */
    (unsigned char)     0x01,         /* g            */
    (unsigned char)     0x00,         /* base[31:24]  */
},
```

Figura 4: Este descriptor corresponde al segmento de código de nivel 0

Los otros dos que faltan son exactamente iguales, solo que en la linea correspondiente al nivel (dpl) ponemos 0x03, ya que corresponde al nivel 3 de prioridad.

Los descriptores de segmentos tienen la siguiente forma:

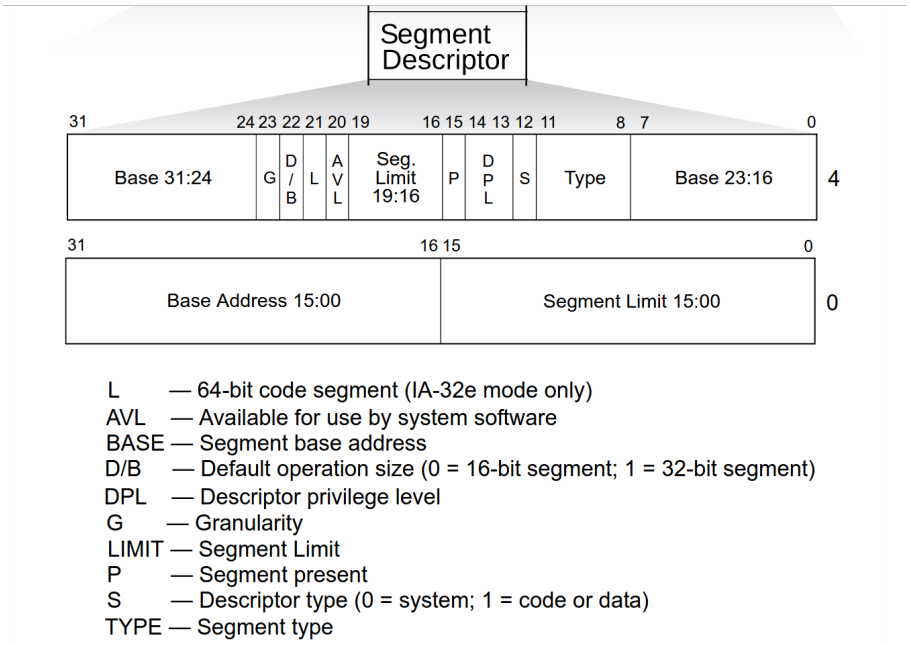

Figura 5: Este descriptor corresponde a la primer entrada de la gdt

Completamos nuestros descriptores como marcan las figuras 3 y 4 de esa forma porque:

Base: En el tp se pide que los descriptores direcciones los primeros 500mb de memoria. Por ende la base corresponde a la direccion 0x00000000.

G: Para poder direccionar 500mb, no nos alcanza la cantidad de bits que hay para el limite, por ende necesitamos activar la granularidad para poder abarcar más memoria, ya que cuando esta activada la posicion que indica el limite se multiplica por 4kb.

Límite: Como esta activada la granularidad, podemos abarcar 500mb de memoria, el limite correspondiente a 500mb con la granularidad activada es 0x0F400.

Type: Aqui se indica si el descriptor es de código/datos, a los correspondientes a datos les pusimos que eran de tipo 0x02 (segmento de datos de escritura/lectura) y a los de codigo que eran de tipo 0x0A (segmentdo de código de escritura/lectura).

S: Con este bit se decide si es un segmento de sistema (s=0) o si son de código/data (s=1), por ende a este bit le corresponde un 0x1.

Dpl: En esta seccion se declara el privilegio del segmento, a los que eran de nivel 0 les corresponde un 0x00 y a los de nivel 3 un 0x03

P: Este es el bit de present. Cuando es '1' el segmento correspondiente esta presente en la memoria RAM. Si es '0', el segmento esta en la memoria virtual. Por ende lo seteamos en 1.

Avl: Es el bit correspondiente a Available. Como no lo vamos a tener en cuenta lo dejamos en 0.

L: Indica si el codigo es de 64bits o de 32. Como trabajamos en 32bits dejamos este bit en 0.

D/B: Este bit define el tamaño de las operaciones en las que va a trabajar el procesador. De nuevo, como nos encontramos trabajando en 32bits, el tamaño de las operaciones debe ser de 32, por eso lo seteamos en 1.

2.3. Ítem B: Setear la GDT