Trabajo Práctico III

subtitulo del trabajo

Organización del Computador II

Segundo Cuatrimestre de 2014

Integrante	LU	Correo electrónico
Nombre	XXX/XX	`mail`
Nombre	XXX/XX	`mail`

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

Resumen

En el presente trabajo se describe la problemática de ...

Índice

Figura 1: Descripcion de la figura

1. Objetivos generales

El objetivo de este Trabajo Práctico es ...

2. Contexto

Titulo del parrafo Bla bla bla bla. Esto se muestra en la figura 1.

```
struct Pepe {

    ...

};
```

3. Enunciado y solucion

3.1. Ejercicio 1

En el ejercicio 1 se completa e inicializa la GDT, con cuatro segmentos, 2 de codigo(uno del kernel y el otro del usuario), y dos de datos(uno del kenrnel y otro del usuario), dirrecionando los primeros 878MB de memoria.(Se dejan las primeras 3 posiciones de la gdt libre, una or ser la nula y las otras dos por reestricciones del tp).El primer indice que deben usar para declarar los segmentos es el 4(contando desde 0).

Para completar la gdt, agregamos los descriptores de segmento a la GDT, modificando el archivo gdt.c. Alli, describimos los segmentos completando structuras de descriptores y descriptor de gst(str_gdt_entry, y str_gdt_descriptor, definidas en gdt.h), Alli, al nulo se le pone todo 0 y a los otros se le pone: Se las flatea, poniendoles a todas la misma direccion base (0x00) y de limite se coloca el tamano-1 /0x400 +0x3FF(se pone asi, porqu en realidad vamos a poner el de granularidad en 1, para que la cuenta nos de el tamano pedido menos 1. el tipo es read/write(0x02) en los de data, y (execute/read) en los de codigo. El s es 1 x ser de datos o codigos, la dpl es 3 o 0 dependiendo si es un segmento de usuario o del kernel. el p es 1, (???), el l esta en 0 x estar en 32 bytes. lel db esta en 1 por el mismo motivo. Tambien hay un segmento de video(del kernel) cuya direccion de entrada esla pedida por la catedra(y limite de acuerdo al tamano de la pantalla pedida.)), el resto es como un segmento de codigo definido antes. Utilizando estos defines

(imagen de toma de pantalla)

se dispuso asi en la gdt.c.

(imagen de toma de pantalla) (imagen de toma de pantalla) (imagen de toma de pantalla)

En el kernel.asm, se pasa a modo protegido, para hacerlo se pone la directiva BITS 16, (para que el linker sepa que se interpreta en 16 bits las direcciones) Luego , se desabilitan ls interrupciones(cli), se cambia el modo de video(interrupcion)(se va a modo 3h y luego se setea) Se da mensaje de bienvenida, pero como no es parte del ejerciocio no lo describo(por ahora) Se habilita la A20 (con una funcion) y se carga la gdt con la funcion lgdt(y la direccion de la tabla(y tam))/ Luego se pasa a modo protegido seteando el bit PE del registro CR0. y se salta a la siguiente instruccion, (desde seg de codigo de la gdt), que es la siguiente (en el medio, se usa la direcctiva BITS 32, para que reconozca que trabajamos con 32 bites). Se establecen los selecotores de segmento (ds, es, fs, gs, ss)(todos apuntan al segmento de datos del kernel, porque es el codigo del kernel) Con estos segmentos seteados, podemos establecer la pila moviendo la base a los reg ebp y esp (bdireccion pedida por catedra) Se imprime otro mensaje(luego describo), y se inicializa la pantalla, para ell llamamos a la funcion inicializar_pantalla, que pushea ds, mueve eax al segmento de video,(y a la primera posicion), y mientras el contador ecx loopee(tiene la cantidad de words a pintar de gris), avanzamos eax y por cada 2 bytes llenamos el lugar del color querido (01110000b ; 0111 = grey sin bright, 0000 = black sin bright). Luego popeamos ds y salimos. (porque dejamos el espacio???)

Terminamos ejercicio 1

3.2. Ejercicio 2

Tenemos que completar las entradas de la IDT para asociar las diferentes rutinas a todas las llamadas del procesador. La IDT (Interrupt Descriptor Table), es la que almacena los decriptores de interrupciones(desriptores de sistema que pueden ser de tres tipos, trap, interrupt o task, nosotros vamos a usar interrupt solamente). Para completar la IDT llamamos desde el kernel.asm a idt_inicializar.Esta funcion contiene los descriptores que seran puestos en la GDT (lo que hacemos utilizando la instruccion lidt [IDT_DESC], donde IDT_DESK el limite y la base de la IDT.) En las entradas de la IDT(IDT_ENTRY), colocamos el offset de la direccion de la rutina de atencion de interrupciones (definidas como isr(numero de interrupcion)), y las rutinas de interrupcion consisten en imprimir infinitamente el texto el nombre de la excepcion que el procesador genero, en la pantalla(en la parte superior derecha); en los bits correspondientes del descriptor de interrupciones, pusimos los atributos (presente, le dimos prioridad 0 porque son excepciones del procesador, aclaramos el tipo(interrupt) y que sea de 32bits)EL segsel es el 0x18 por ser el tercero de la gdt, donde esta el descriptor del segmento donde se encuentran las rutinas de atencion de interrupciones. Cuando se produce una interrupcion, el procesador busca en la istr la idt, va a la Idt y se va a la puerta de esa interrupcion, se usa el segmento para ir al segmento en la gdt donde esta el segmento, se le suma a la base el offset de la interrupcion para ir a la rutina correspondiente, y se la ejecuta.

Terminamos el ejercicio 2

3.3. Ejercicio 3

Vamos a limpiar el buffer de video para que se vea como lo indica la figura 7. Para ello, creamos la funcion imprimir_pantalla en screen.c. La funcion imprimir_pantalla dibuja los bordes (negro), los marcadores(la parte roja y azul de la pantalla correspondiente a cada jugador), los relojes(que luego de deberemos actualizar para que respondan a la interrupcion que genere una tarea siendo ejecutada), e imprime las vidas. Aqui tambien se utiliza la funcion print_int_sinattr, que como el nombre lo indica imprime en el lugar buscado del buffer de video el numero sin atributos,(agarramos, los que estan en actualizar vidas.)Luego tendremos que hacer esto actualizando las vidas con interrupciones. Cada vez que imprimimos excepto esta usamos la funcion print dada. Para la medicion de los margenes utilizamos defines(fijarse en la seccion defines)

Ahora vamos a definir la estructura de paginacion, y posteriormente, vamos a activar la paginacion. Para definir lo primero, se inicializa el manejador de memoria llamando a mmu_inicializar,que nos coloca en las posicion del principio de las paginas libres(0x10000).A partir de ahora, las paginas seran dadas desde esta direccion de memoria. Luego, inicializamos el directorio del kernel. Lo que hacemos es (mediante

mmu_inicializar_dir_kernel) es: Primero, nos guardamos la direccion del directorio de paginas del kernel(0x27000) y la direccion de las tablas de paginas (0x28000). Luego, para el numero de entradas del directorio paginas que requerimos para el identity mappig de nuestro tp (lease, una), se completa las entradas del directorio con la direccion de la tabla de paginas definida anteriormente y los atributos de estar presente y de poder ser leida y/o escrita(los ultimos bits en 3).Luego, como necesitamos todas las paginas que nos provee una tabla de paginas,(1024), llenamos cada entrada con la direccion que queremos mapear(0x00000000 a 0x003FFFFF), y los atributos de ser leida a la entrada, escrita y presente). Luego avanzamos ptabla la cantidad de entradas de la pagina.(en este caso no se necesita, pero si se requisiese otra tabla de paginas en el directorio, que el ptabla apunte a la direccion de la proxima tabla.) Luego, se completa las entradas no validas del direcctorio de paginas con 0 para que no haya problemas. Por ultimo, como vamos a cambiar la cr3 para inicializar la paginacion, flusheamos la tlb. Luego movemos la direccion del directorio cargada en cr3, y habilitamos la paginacion seteando el bit 31 de cr0.

Imprimimos el nobre de grupo usando la funcion imprimir_texto.

Terminamos el ejercicio 3

3.4. Ejercicio 4

En este ejercicio se tiene que definir los generadores de las estructuras que nos van a permitir realizar la paginacion. En

4. Defines

5. Conclusiones y trabajo futuro