DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Practico 3

Forest Adventure

Organización del Computador 2

Integrante	LU	Correo electrónico
Hayon, Gabriel David	701/17	gabrielhayonort@gmail.com
Lopes Perera, Pablo	007/18	plopesperera99@gmail.com
Naftaly, Joaquin	816/17	joaquin.naftaly@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2610 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (++54 +11) 4576-3300
https://exactas.uba.ar

Índice

1. Juego *Forest Adventure*

El juego en sí consiste en dos pueblos de *Lemmings*, *Amalin* y *Betarote* que irán enviando sus Lemmings a recorrer el valle de *Titinita*, tratando de llegar al pueblo opuesto. Estos Lemmings saldrán en orden desde un punto predeterminado, con un máximo de 5 Lemmings en simultáneo por equipo. Si ya hay 5 Lemmings en juego, luego de cierto tiempo, el Lemming más antiguo se da por vencido y le deja su lugar a uno nuevo.

1.1. Propiedades del juego

Cada 401 ciclos de reloj, si algún equipo tiene menos de 5 Lemmings en juego, el sistema creará un nuevo Lemming para ese equipo, posicionándolo en la dirección inicial del mapa para su equipo. Cada 2001 ciclos de reloj, si algún equipo tiene 5 Lemmings en juego, el sistema desalojará al Lemming más antiguo y creará uno nuevo, posicionándolo en la dirección inicial del mapa para su equipo. Si por algún motivo la dirección inicial del mapa para un equipo está ocupada, y esto no le permite crear el Lemming, la creación falla y no será creado un Lemming. Una vez que son creados, los Lemmings pueden llamar a servicios del sistema para realizar algunas de las acciones que tienen disponibles, con el objetivo de llegar al extremo opuesto de la pantalla. El juego termina cuando algún Lemming llega al extremo opuesto del mapa (el jugador de la izquierda gana cuando alguno de sus Lemmings llega a cualquier celda en la última columna del mapa).

2. Pantalla

La pantalla es de tamaño `80x50`. En el mismo tenemos un mapa que ocupa `80x40`, dejando la primer linea de la pantalla en negro para mensajes del sistema. En la sección inferior hay un contador de puntos por cada jugador y el estado de los Lemmings (Ver Figura 1).

El valle de *Titinita* será modelado como un mapa rectangular de `80x40` celdas, y va a contar con ciertos obstáculos: paredes, lagos, y los bordes del valle.

Figura 1: Pantalla del juego

Como los Lemmings pueden explotar y crear puentes, estos reemplazaran en el mapa del juego una pared por una `X` y un lago por un `+` respectivamente. Los Lemmings pueden caminar por sobre puentes o paredes explotadas.

Como interfaz gráfica también se encuentra la ventana del Modo Debugging, con la cual se accede presionando la tecla `Y` (Ver 8.2).

2.1. Mapa

El mapa es representado por un arreglo 2-dimensional en una variable dentro del sistema.

Nota En los tests de la catedra el mapa es una variable `const`. En nuestra implementación del sistema, el mapa es una variable **no** `const`. Tener esto en cuenta al correr los tests.

3. Organización de memoria

Se utilizo el modelo de segmentación flat, con 2 niveles de protección. Los segmentos son de un tamaño de 817MiB.

3.1. GDT

En la GDT incluimos descriptores de segmento diferenciándolos por el DPL (nivel de permisos), 0 para nivel Kernel y 3 para nivel Usuario, por lo tanto teniendo dos segmentos de código, uno nivel 0 y otro nivel 3, y lo mismo para segmentos de datos. Dentro de la GDT también se encuentra un segmento para la memoria de vídeo, de DPL nivel 0.

Para los descriptores de la GDT se utiliza una estructura en `C` que facilita el armado de las mismas. Esta es una estructura que se corresponde con el descriptor de segmento especificado en el manual de Intel.

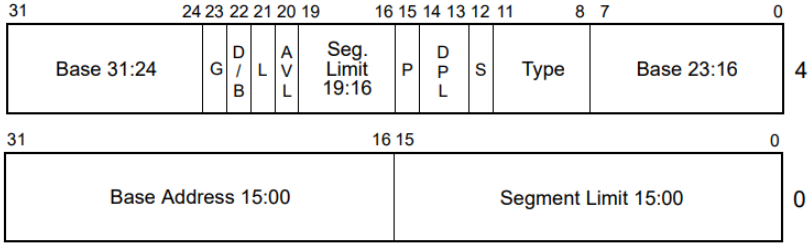

Figura 2: Descriptor de segmento especificado en el manual de Intel

La estructura en 2 se utiliza para crear los descriptores de segmento para el segmento de código nivel 0 y 3, para datos nivel 0 y 3, y para vídeo nivel 0.

Para la tarea inicial y la tarea idle (ambas en nivel 0) se crean segmentos que están ubicados en los índices 14 y 15, respectivamente. Luego, hay un índice dinámico de tarea que comienza en el 16, el cual se utiliza para las tareas *Amalin* y *Betarote*. Notar que cada equipo puede tener hasta 5 Lemmings en simultaneo. Por lo tanto el rango completo es desde la tarea 16 hasta la 26.

El cuadro 1 representa los índices de la tabla GDT implementada en el sistema.

Indice	Segmento	Tipo	DPL	G
0	Segmento Null	-	0	0
8	Código	Execute/Read	0	1
9	Código	Execute/Read	3	1
10	Datos	Read/Write	0	1
12	Datos	Read/Write	3	1
12	Vídeo	Read/Write	0	0
14	TSS Idle	32-bit TSS	0	0
15	TSS Initial	32-bit TSS	0	0
16-26	TSS Lemmings	32-bit TSS	0	0

Cuadro 1: Tabla de descriptores globales (GDT)

3.2. Pila

La base de la pila del kernel se encuentra en la posición `0x25000`.

3.3. Paginación y MMU

3.3.1. Kernel

Para la paginación del kernel se crea un mapa de memoria. El directorio de paginas del mismo se encuentra en la posición `0x25000` y la tabla de paginas en `0x26000`. Para el kernel se realiza *identity mapping* de las direcciones entre `0x00000000` y `0x003FFFFF`. Consecuentemente, el registro `CR3` apunta a la posición `0x25000`. Este mapeo se realiza con permisos de supervisor y read/write.

El sistema permite mapear paginas en una dirección física a una dirección virtual utilizando `CR3`. Esto se realiza con una función implementada en `C` que, en el caso que no exista directorio de la misma, busca la siguiente pagina libre (en el área libre de kernel) y realiza el proceso de mapeo. Esta función es utilizada en el sistema para realizar el mapeo de memoria para las tareas.

El sistema puede desmapear paginas dada una dirección virtual utilizando `CR3`. Este comportamiento genera perdida de memoria, ya que solo realiza un borrado lógico del mapeo creado, intercambiando el bit de `present` en la pagina del directorio por un `0`, en el caso que era `1`; si no, deja en el estado que se encuentra.

3.3.2. Memoria Compartida

El sistema les provee a los Lemmings de un mismo pueblo un método de comunicación mediante páginas compartidas, mapeadas bajo demanda. El rango virtual `0x400000 - 0x13FFFF` es reservado para tal fin. Cuando una tarea Lemming es lanzada, tiene esas páginas desmapeadas. Al acceder a una página dentro de ese rango, si es la primera vez que un Lemming de ese pueblo accede, se mapea una nueva página física. Si otro Lemming de ese pueblo ya había accedido a esa página, se mapea la misma página física, permitiendo que los Lemmings se comuniquen por medio de esta memoria compartida.
Esta funcionalidad es brindada en la rutina de atención de interrupciones de Page Fault.

3.3.3. Tareas

Cada tarea tiene su directorio y tabla de paginas, las mismas están ubicadas en el área libre de kernel. Estos mapeos se realizan con permisos de supervisor y de lectura/escritura, y los mismos se realizan con *identity mapping*.

Las tareas que ejecuta el sistema son o bien los jugadores o tareas Lemmings controladas por alguno de los jugadores. Cada tarea Lemming ocupa exactamente 12KiB de memoria, dos paginas son para la tarea y una para la pila de la tarea.

3.4. Descriptores de la TSS

Los descriptores de la TSS se utiliza una estructura en `C` que facilita el armado de las mismas. Esta es una estructura que se corresponde con el segmento de tareas especificado en el manual de Intel.

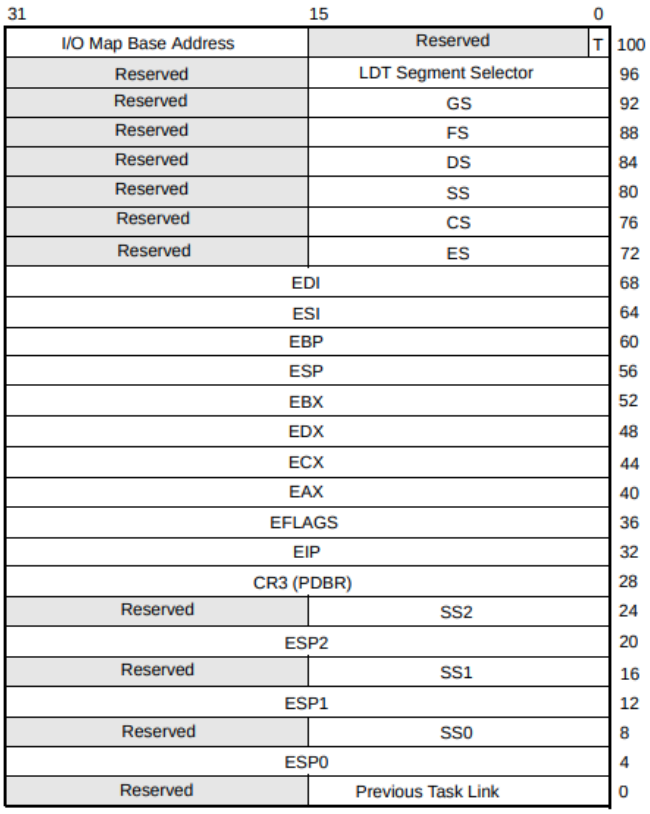

Figura 3: Task-State Segment especificado en el manual de Intel

4. Interrupciones

4.1. IDT

En el Cuadro 2 tenemos la tabla de interrupciones. Notar que las interrupciones 2 y 15 no están implementadas, ya que estas son reservadas por Intel [1, p. 6-3].

Índice	Interrupción	DPL
0	Divide Error	0
2	NMI Interrupt	0
3	Breakpoint	0
4	Overflow	0
5	BOUND Range Exceeded	0
6	Invalid Opcode (Undefined Opcode)	0
7	Device Not Available (No Math Coprocessor)	0
8	Double Fault	0
9	Coprocessor Segment Overrun (reserved)	0
10	Invalid TSS	0
11	Segment Not Present	0
12	Stack-Segment Fault	0
13	General Protection	0
14	Page Fault	0
16	x87 FPU Floating-Point Error (Math Fault)	0
17	Alignment Check	0
18	Machine Check	0
19	SIMD Floating-Point Exception	0
32	Reloj	0
33	Teclado	0
88	Interrupción 88	3
98	Interrupción 98	3
108	Interrupción 108	3

Cuadro 2: Tabla de entradas de la `IDT` implementadas

Las interrupciones `88,98` y `108` se implementan como syscalls (definidas mas adelante).

5. Syscalls

5.1. Servicios

El sistema dispone de tres servicios, que cada uno es atendido por una interrupción distinta. Las syscalls hacen un trigger de las interrupciones `88,98,108`. Las syscalls son: `move`, `explode` y `bridge`.

- Sycall `move` (`int 88`)

Parametros	Descripcion
`in EAX=direccion`	0: Arriba, 1: Derecha, 2: Abajo, 3: Izquierda
`out EAX=result`	0: Se desplazó sin problemas. 1: Pared de Ladrillo. 2: Agua. 3: Borde del mapa. 4: Lemming.

Toma como parámetro de entrada en EAX una dirección de movimiento. El sistema revisará si puede mover el Lemming en esa dirección o hay algún obstáculo que lo impida. En caso de que no haya ningún obstáculo, el servicio retornará 0 en EAX. En caso de que haya un obstáculo, el sistema le retornará un código al Lemming

en EAX para que este pueda decidir qué hacer al respecto. Un Lemming no podrá moverse pasando el borde del mapa, ni sobre agua, ni sobre una pared de ladrillos. Luego de ejecutar la interrupción, el Lemming pierde el turno, ejecutando la tarea `IDLE` hasta el siguiente clock.

- Syscall `explode` (`int 98`)

Parametros	Descripcion
`x`	Esta llamada a sistema no toma parámetros

El Lemming que invoca a esta llamada a sistema se auto destruye, destruyendo todas las paredes de ladrillo y a todos los Lemmings que se encuentren a su alrededor. Esta syscall no retorna. El Lemming que la invoca es desalojado del sistema.

- Syscall `bridge` (`int 108`)

Parametros	Descripcion
`in EAX=direccion`	0: Arriba, 1: Derecha, 2: Abajo, 3: Izquierda

Esta llamada a sistema puede ser usada por un Lemming para crear un puente. En EAX se indica la dirección con respecto a la posición del lemming en donde se desea crear el puente. En caso de que haya agua en esa posición, un puente será creado. Si no hay agua en esa posición, el puente no será creado. En cualquier caso, el Lemming que realizó la acción será desalojado del sistema.

Las tareas son ejecutadas una a continuación de la otra. Este proceso se repete indenidamente. No siempre habrá una tarea Lemming para ejecutar, en ese caso, se ejecuta la tarea Idle.

6. Tareas

Las tareas del sistema son jugadores de los equipos de Amalin y Betarote. Cada tarea va a intentar llegar hacia el extremo opuesto del mapa sorteando diferentes obstáculos (paredes y lagos). Para eso las tareas van a moverse, explotar o crear puentes mediante syscalls (ver 4.1). Si alguna de esas acciones no se puede hacer la tarea es desalojada. En caso de que se haya podido ejecutar, la tarea ejecuta la syscall y pierde su turno.

7. Scheduler

El scheduler implementado en el sistema es del tipo Round-Robin. El sistema va a correr tareas de forma concurrente, durante un tiempo fijo denominado quantum. Este será determinado por un tick de reloj. Un ejemplo del scheduler del sistema se puede ver en la Figura 4.

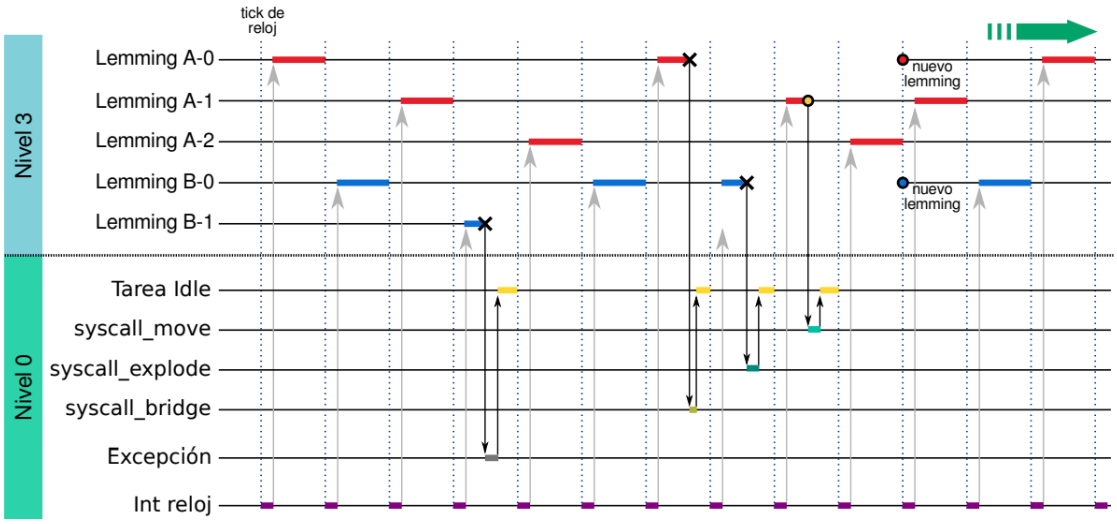

Figura 4: Esquema del scheduler

8. Estructuras del sistema

- `typedef enum direction_e`: Son sentidos de direccion para mover.
- `typedef enum move_result_e`: Resultados de la syscall move.
- `typedef struct str_page_directory_entry`: Entrada de la page directory
- `typedef struct str_page_table_entry`: Entrada de la table entry
- `typedef enum e_team`: Enum de los equipos
- `typedef struct str_coords`: Par ordenado de las coordenadas
- `typedef struct str_sched_task_entry`: Estructura para el sched que contiene info de las tareas
- `typedef struct str_sched_task_state`: Estructura para el sched que contiene estados de las tareas
- `typedef struct str_sched_lemmings_state`: Estructura para el sched que contiene estados de los lemmings
- `typedef struct str_error_state`: Estructura utilizada para guardar estados de error devueltos por una interrupcion
- `typedef struct str_debug_state`: Estructura de estados para modo debug
- `typedef struct str_game_state`: Estado del juego
- `typedef uint32_t vaddr_t`: Memoria Virtual
- `typedef uint32_t paddr_t`: Memoria Fisica

8.1. Desalojo de Tareas

Las tareas son desalojadas en uno de los siguientes casos

1. Pasan `2001` ciclos de clock, donde el Lemming mas viejo muere.
2. Cuando un Lemming explota, este es desalojado.
3. Cuando un Lemming que explota tiene otro Lemmings alrededor, estos también mueren.

4. Cuando un Lemming crea un puente.

5. Cuando genera cualquier otro tipo de excepción (Excepto page fault en algunos casos, ver 3.3.1).

8.2. Modo Debugging

La tecla `Y` activa el modo debug. En este modo se muestra en pantalla la primera excepción capturada por el procesador junto con un detalle de todo el estado del procesador. Una vez impresa en pantalla esta excepción, el juego se detendrá hasta presionar nuevamente la tecla `y` que mantendrá el modo debug pero borra la información presentada en pantalla por la excepción. La forma de detener el juego será instantánea. Al retomar el juego se esperará hasta el próximo ciclo de reloj en el que se decidirá cuál es la próxima tarea a ser ejecutada.

Referencias

[1] Intel Corporation *Intel 64 and IA-32 Architectures Software Developer's Manual Volume 3: System Programming* Intel Corporation, Santa Clara, California, U.S, 2016