DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico III

TronTank

Organización del Computador II

Primer Cuatrimestre de 2014

Integrante	LU	Correo electrónico
Tomas Shaurli	671/10	tshaurli@gmail.com
Sebastian Aversa	379/11	sebastianaversa@gmail.com
Fernando Gabriel Otero	424/11	fergabot@gmail.com

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 realiza la configuración de un kernel de sistema operativo de 32bits con sus elementos indispensables (GDT, IDT, TSS, paginación, etc) y el manejo de tareas.

Índice

1. Objetivos generales

Este trabajo práctico consistió en un conjunto de ejercicios para el parendizaje de los conceptos de System Programming
Se implementó para ello un sistema mínimo en base a los archivos aportados por la cátedra, el cual será capaz de manejar exactamente 8 tareas a nivel de usuario. El sistema será capaz de capturar cualquier problema que puedan generar las tareas y tomar las acciones necesarias para quitar a la tarea del sistema.
Los ejercicios de este trabajo práctico proponen utilizar los mecanismos que posee el procesador para la programación desde el punto de vista del sistema operativo enfocados en dos aspectos: el sistema de protección y la ejecución concurrente de tareas.

2. Kernel.asm

2.1. Básico:

En el kernel lo primero que hacemos es deshabilitar las interrupciones con **CLI** y cambiamos el modo de video. Luego, iremos habilitando y creando los diversos componentes, llamando a las funciones destinadas para ése fin.

Esas funciones son listadas al comienzo del código:

```
extern GDT_DESC
extern IDT_DESC
extern idt_inicializar
extern mmu_inicializar
extern mmu_inicializar_dir_kernel
extern pintar
extern pintarTablero
extern deshabilitar_pic
extern resetear_pic
extern habilitar_pic
```

2.2. Preparando el modo protegido:

Lo segundo que tenemos que hacer es habilitar A20 en el controlador del teclado para tener acceso a direcciones superiores a los 2^{20} bits

Luego, cargamos la **GDT** con *LGDT*, pasandole el descriptor de la GDT que ya tiene el formato adecuado

```
    ; Habilitar A20
    CALL habilitar_A20
    ; Cargar la GDT
    LGDT [GDT_DESC]
```

Ahora, con la GDT cargada y A20 habilitado, podemos pasar a modo protegido seteando el bit PE de CR0 (bit 0) en 1.

Al finalizar debemos hacer un *far jump* a la etiqueta de modo protegido. Es importante notar que el far jump es la unica forma de modificar el CS, operación necesaria para que no haya errores.

```
; Setear el bit PE del registro CR0 (esto pasa a modo protegido)
    MOV EAX,CR0
    OR EAX,1
    MOV CR0,EAX
    ; Saltar a modo protegido
    JMP (9*0x08):modoProtegido
```

2.3. Modo protegido:

Ya estamos en modo protegido. Ahora seteamos los selectores de segmentos haciendo uso de la GDT.

```
    BITS 32
    modoProtegido:
        ;CODIGO
    ; Establecer selectores de segmentos
    XOR EAX, EAX
    MOV AX, 1011000b ;1011b == 11d (index de la GDT) | 0 (0 -> GDT / 1 -> LDT) |
                                                    | 00 (NIVEL DE PRIVILEGIO)
    MOV DS, AX
    MOV ES, AX
    MOV GS, AX
    MOV SS, AX
    MOV AX, 1101000b
    MOV FS, AX
```

Luego, establecemos la base de la pila, llamamos a *pintar()* que limpia la pantalla y pinta el area que representa el sector *el_mapa*, llamamos a idt_inicializar y, una vez inicializada, le pasamos el descriptor de la **IDT** a *LIDT*

```
    ; Establecer la base de la pila
    MOV ESP, 0x27000
    MOV EBP, ESP
    CALL pintar
    CALL idt_inicializar
    LIDT [IDT_DESC]

    xor eax, eax
    xor edi, edi
    div edi
    CALL pintarTablero
```

2.4. Paginación:

mmu_inicializar_dir_kernel MOV CR3, EAX ;cargo en CR3 la direccion del page directory
MOV EAX, CR0 OR EAX, 0x80000000 ;habilitamos paginacion MOV CR0, EAX

```
CALL mmu_inicializar_dir_kernel
MOV EAX, 0x27000
MOV CR3, EAX ;cargo en CR3 la direccion del page directory
MOV EAX, CR0
    OR EAX, 0x80000000 ;habilitamos paginacion
MOV CR0, EAX
```

```
    ; Imprimir mensaje de bienvenida

    ; Inicializar pantalla

    ; Inicializar el manejador de memoria
```

```
    ; Inicializar el directorio de paginas
    CALL mmu_inicializar

    CALL deshabilitar_pic
    CALL resetear_pic
    CALL habilitar_pic
    STI
```

RESTO

3. Kernel.asm

3.1. Modificaciones:

En el kernel lo primero que hacemos es deshabilitar las interrupciones con **cli** y cambiamos el modo de video. Luego, iremos habilitando y creando los diversos componentes, llamando a las funciones destinadas para ése fin.
Esas funciones son listadas al comienzo del código:

```
extern GDT_DESC
extern IDT_DESC
extern idt_inicializar
extern mmu_inicializar
extern mmu_inicializar_dir_kernel
extern pintar
extern pintarTablero
extern deshabilitar_pic
extern resetear_pic
extern habilitar_pic
```

```
    ; Habilitar A20
    CALL habilitar_A20
    ; Cargar la GDT
    LGDT [GDT_DESC]
```

```
; Setear el bit PE del registro CR0 (esto pasa a modo protegido)
    MOV EAX,CR0
    OR EAX,1
    MOV CR0,EAX
    ; Saltar a modo protegido
    JMP (9*0x08):modoProtegido

    BITS 32
```

```
    modoProtegido:
        ;CODIGO

    ; Establecer selectores de segmentos
    XOR EAX, EAX
    MOV AX, 1011000b ;1011b == 11d (index de la GDT) | 0 (0 -> GDT / 1 -> LDT) | 00 (NIVEL DE PRIVILEGIO)
    MOV DS, AX
    MOV ES, AX
    MOV GS, AX
    MOV SS, AX
    MOV AX, 1101000b
    MOV FS, AX
    ; Establecer la base de la pila
    MOV ESP, 0x27000
    MOV EBP, ESP
    CALL pintar
    CALL idt_inicializar
LIDT [IDT_DESC]

    xor eax, eax
    xor edi, edi
    div edi
CALL pintarTablero
```

```
CALL mmu_inicializar_dir_kernel
MOV EAX, 0x27000
MOV CR3, EAX ;cargo en CR3 la direccion del page directory
MOV EAX, CR0
    OR EAX, 0x80000000 ;habilitamos paginacion
MOV CR0, EAX
    ; Imprimir mensaje de bienvenida

    ; Inicializar pantalla

    ; Inicializar el manejador de memoria

    ; Inicializar el directorio de paginas
    CALL mmu_inicializar

    CALL deshabilitar_pic
    CALL resetear_pic
    CALL habilitar_pic
    STI
```

4. Kernel.asm

4.1. Modificaciones:

En el kernel lo primero que hacemos es deshabilitar las interrupciones con **cli** y cambiamos el modo de video. Luego, iremos habilitando y creando los diversos componentes, llamando a las funciones destinadas para ése fin.
Esas funciones son listadas al comienzo del código:

```
extern GDT_DESC
extern IDT_DESC
extern idt_inicializar
extern mmu_inicializar
extern mmu_inicializar_dir_kernel
extern pintar
extern pintarTablero
extern deshabilitar_pic
extern resetear_pic
extern habilitar_pic
```

```
    ; Habilitar A20
    CALL habilitar_A20
    ; Cargar la GDT
    LGDT [GDT_DESC]
```

```
; Setear el bit PE del registro CR0 (esto pasa a modo protegido)
    MOV EAX,CR0
    OR EAX,1
    MOV CR0,EAX
    ; Saltar a modo protegido
    JMP (9*0x08):modoProtegido

    BITS 32
```

```
    modoProtegido:
        ;CODIGO

    ; Establecer selectores de segmentos
    XOR EAX, EAX
    MOV AX, 1011000b ;1011b == 11d (index de la GDT) | 0 (0 -> GDT / 1 -> LDT) | 00 (NIVEL DE PRIVILEGIO)
    MOV DS, AX
    MOV ES, AX
    MOV GS, AX
    MOV SS, AX
    MOV AX, 1101000b
    MOV FS, AX
    ; Establecer la base de la pila
    MOV ESP, 0x27000
    MOV EBP, ESP
    CALL pintar
    CALL idt_inicializar
LIDT [IDT_DESC]

    xor eax, eax
    xor edi, edi
    div edi
CALL pintarTablero
```

```
CALL mmu_inicializar_dir_kernel
MOV EAX, 0x27000
MOV CR3, EAX ;cargo en CR3 la direccion del page directory
MOV EAX, CR0
    OR EAX, 0x80000000 ;habilitamos paginacion
MOV CR0, EAX
    ; Imprimir mensaje de bienvenida

    ; Inicializar pantalla

    ; Inicializar el manejador de memoria

    ; Inicializar el directorio de paginas
    CALL mmu_inicializar

    CALL deshabilitar_pic
    CALL resetear_pic
    CALL habilitar_pic
    STI
```

5. Kernel.asm

5.1. Modificaciones:

En el kernel lo primero que hacemos es deshabilitar las interrupciones con **cli** y cambiamos el modo de video. Luego, iremos habilitando y creando los diversos componentes, llamando a las funciones destinadas para ése fin.
Esas funciones son listadas al comienzo del código:

```
extern GDT_DESC
extern IDT_DESC
extern idt_inicializar
extern mmu_inicializar
extern mmu_inicializar_dir_kernel
extern pintar
extern pintarTablero
extern deshabilitar_pic
extern resetear_pic
extern habilitar_pic
```

```
    ; Habilitar A20
    CALL habilitar_A20
    ; Cargar la GDT
    LGDT [GDT_DESC]
```

```
; Setear el bit PE del registro CR0 (esto pasa a modo protegido)
    MOV EAX,CR0
    OR EAX,1
    MOV CR0,EAX
    ; Saltar a modo protegido
    JMP (9*0x08):modoProtegido

    BITS 32
```

```
    modoProtegido:
        ;CODIGO

    ; Establecer selectores de segmentos
    XOR EAX, EAX
    MOV AX, 1011000b ;1011b == 11d (index de la GDT) | 0 (0 -> GDT / 1 -> LDT) | 00 (NIVEL DE PRIVILEGIO)
    MOV DS, AX
    MOV ES, AX
    MOV GS, AX
    MOV SS, AX
    MOV AX, 1101000b
    MOV FS, AX
    ; Establecer la base de la pila
    MOV ESP, 0x27000
    MOV EBP, ESP
    CALL pintar
    CALL idt_inicializar
LIDT [IDT_DESC]

    xor eax, eax
    xor edi, edi
    div edi
CALL pintarTablero
```

```
CALL mmu_inicializar_dir_kernel
MOV EAX, 0x27000
MOV CR3, EAX ;cargo en CR3 la direccion del page directory
MOV EAX, CR0
    OR EAX, 0x80000000 ;habilitamos paginacion
MOV CR0, EAX
    ; Imprimir mensaje de bienvenida

    ; Inicializar pantalla

    ; Inicializar el manejador de memoria

    ; Inicializar el directorio de paginas
    CALL mmu_inicializar

    CALL deshabilitar_pic
    CALL resetear_pic
    CALL habilitar_pic
    STI
```

6. Kernel.asm

6.1. Modificaciones:

En el kernel lo primero que hacemos es deshabilitar las interrupciones con **cli** y cambiamos el modo de video. Luego, iremos habilitando y creando los diversos componentes, llamando a las funciones destinadas para ése fin.
Esas funciones son listadas al comienzo del código:

```
extern GDT_DESC
extern IDT_DESC
extern idt_inicializar
extern mmu_inicializar
extern mmu_inicializar_dir_kernel
extern pintar
extern pintarTablero
extern deshabilitar_pic
extern resetear_pic
extern habilitar_pic
```

```
    ; Habilitar A20
    CALL habilitar_A20
    ; Cargar la GDT
    LGDT [GDT_DESC]
```

```
; Setear el bit PE del registro CR0 (esto pasa a modo protegido)
    MOV EAX,CR0
    OR EAX,1
    MOV CR0,EAX
    ; Saltar a modo protegido
    JMP (9*0x08):modoProtegido

    BITS 32
```

```
    modoProtegido:
        ;CODIGO

    ; Establecer selectores de segmentos
    XOR EAX, EAX
    MOV AX, 1011000b ;1011b == 11d (index de la GDT) | 0 (0 -> GDT / 1 -> LDT) | 00 (NIVEL DE PRIVILEGIO)
    MOV DS, AX
    MOV ES, AX
    MOV GS, AX
    MOV SS, AX
    MOV AX, 1101000b
    MOV FS, AX
    ; Establecer la base de la pila
    MOV ESP, 0x27000
    MOV EBP, ESP
    CALL pintar
    CALL idt_inicializar
LIDT [IDT_DESC]

    xor eax, eax
    xor edi, edi
    div edi
CALL pintarTablero
```

```
CALL mmu_inicializar_dir_kernel
MOV EAX, 0x27000
MOV CR3, EAX ;cargo en CR3 la direccion del page directory
MOV EAX, CR0
    OR EAX, 0x80000000 ;habilitamos paginacion
MOV CR0, EAX
    ; Imprimir mensaje de bienvenida

    ; Inicializar pantalla

    ; Inicializar el manejador de memoria

    ; Inicializar el directorio de paginas
    CALL mmu_inicializar

    CALL deshabilitar_pic
    CALL resetear_pic
    CALL habilitar_pic
    STI
```

7. Kernel.asm

7.1. Modificaciones:

En el kernel lo primero que hacemos es deshabilitar las interrupciones con **cli** y cambiamos el modo de video. Luego, iremos habilitando y creando los diversos componentes, llamando a las funciones destinadas para ése fin.
Esas funciones son listadas al comienzo del código:

```
extern GDT_DESC
extern IDT_DESC
extern idt_inicializar
extern mmu_inicializar
extern mmu_inicializar_dir_kernel
extern pintar
extern pintarTablero
extern deshabilitar_pic
extern resetear_pic
extern habilitar_pic
```

```
    ; Habilitar A20
    CALL habilitar_A20
    ; Cargar la GDT
    LGDT [GDT_DESC]
```

```
; Setear el bit PE del registro CR0 (esto pasa a modo protegido)
    MOV EAX,CR0
    OR EAX,1
    MOV CR0,EAX
    ; Saltar a modo protegido
    JMP (9*0x08):modoProtegido

    BITS 32
```

```
    modoProtegido:
        ;CODIGO

    ; Establecer selectores de segmentos
    XOR EAX, EAX
    MOV AX, 1011000b ;1011b == 11d (index de la GDT) | 0 (0 -> GDT / 1 -> LDT) | 00 (NIVEL DE PRIVILEGIO)
    MOV DS, AX
    MOV ES, AX
    MOV GS, AX
    MOV SS, AX
    MOV AX, 1101000b
    MOV FS, AX
    ; Establecer la base de la pila
    MOV ESP, 0x27000
    MOV EBP, ESP
    CALL pintar
    CALL idt_inicializar
LIDT [IDT_DESC]

    xor eax, eax
    xor edi, edi
    div edi
CALL pintarTablero
```

```
CALL mmu_inicializar_dir_kernel
MOV EAX, 0x27000
MOV CR3, EAX ;cargo en CR3 la direccion del page directory
MOV EAX, CR0
    OR EAX, 0x80000000 ;habilitamos paginacion
MOV CR0, EAX
    ; Imprimir mensaje de bienvenida

    ; Inicializar pantalla

    ; Inicializar el manejador de memoria

    ; Inicializar el directorio de paginas
    CALL mmu_inicializar

    CALL deshabilitar_pic
    CALL resetear_pic
    CALL habilitar_pic
    STI
```

8. Kernel.asm

8.1. Modificaciones:

En el kernel lo primero que hacemos es deshabilitar las interrupciones con **cli** y cambiamos el modo de video. Luego, iremos habilitando y creando los diversos componentes, llamando a las funciones destinadas para ése fin.
Esas funciones son listadas al comienzo del código:

```
extern GDT_DESC
extern IDT_DESC
extern idt_inicializar
extern mmu_inicializar
extern mmu_inicializar_dir_kernel
extern pintar
extern pintarTablero
extern deshabilitar_pic
extern resetear_pic
extern habilitar_pic
```

```
    ; Habilitar A20
    CALL habilitar_A20
    ; Cargar la GDT
    LGDT [GDT_DESC]
```

```
; Setear el bit PE del registro CR0 (esto pasa a modo protegido)
    MOV EAX,CR0
    OR EAX,1
    MOV CR0,EAX
    ; Saltar a modo protegido
    JMP (9*0x08):modoProtegido

    BITS 32
```

```
    modoProtegido:
        ;CODIGO

    ; Establecer selectores de segmentos
    XOR EAX, EAX
    MOV AX, 1011000b ;1011b == 11d (index de la GDT) | 0 (0 -> GDT / 1 -> LDT) | 00 (NIVEL DE PRIVILEGIO)
    MOV DS, AX
    MOV ES, AX
    MOV GS, AX
    MOV SS, AX
    MOV AX, 1101000b
    MOV FS, AX
    ; Establecer la base de la pila
    MOV ESP, 0x27000
    MOV EBP, ESP
    CALL pintar
    CALL idt_inicializar
LIDT [IDT_DESC]

    xor eax, eax
    xor edi, edi
    div edi
CALL pintarTablero
```

```
CALL mmu_inicializar_dir_kernel
MOV EAX, 0x27000
MOV CR3, EAX ;cargo en CR3 la direccion del page directory
MOV EAX, CR0
    OR EAX, 0x80000000 ;habilitamos paginacion
MOV CR0, EAX
    ; Imprimir mensaje de bienvenida

    ; Inicializar pantalla

    ; Inicializar el manejador de memoria

    ; Inicializar el directorio de paginas
    CALL mmu_inicializar

    CALL deshabilitar_pic
    CALL resetear_pic
    CALL habilitar_pic
    STI
```

9. Kernel.asm

9.1. Modificaciones:

En el kernel lo primero que hacemos es deshabilitar las interrupciones con **cli** y cambiamos el modo de video. Luego, iremos habilitando y creando los diversos componentes, llamando a las funciones destinadas para ése fin.
Esas funciones son listadas al comienzo del código:

```
extern GDT_DESC
extern IDT_DESC
extern idt_inicializar
extern mmu_inicializar
extern mmu_inicializar_dir_kernel
extern pintar
extern pintarTablero
extern deshabilitar_pic
extern resetear_pic
extern habilitar_pic
```

```
    ; Habilitar A20
    CALL habilitar_A20
    ; Cargar la GDT
    LGDT [GDT_DESC]
```

```
; Setear el bit PE del registro CR0 (esto pasa a modo protegido)
    MOV EAX,CR0
    OR EAX,1
    MOV CR0,EAX
    ; Saltar a modo protegido
    JMP (9*0x08):modoProtegido

    BITS 32
```

```
    modoProtegido:
        ;CODIGO

    ; Establecer selectores de segmentos
    XOR EAX, EAX
    MOV AX, 1011000b ;1011b == 11d (index de la GDT) | 0 (0 -> GDT / 1 -> LDT) | 00 (NIVEL DE PRIVILEGIO)
    MOV DS, AX
    MOV ES, AX
    MOV GS, AX
    MOV SS, AX
    MOV AX, 1101000b
    MOV FS, AX
    ; Establecer la base de la pila
    MOV ESP, 0x27000
    MOV EBP, ESP
    CALL pintar
    CALL idt_inicializar
LIDT [IDT_DESC]

    xor eax, eax
    xor edi, edi
    div edi
CALL pintarTablero
```

```
CALL mmu_inicializar_dir_kernel
MOV EAX, 0x27000
MOV CR3, EAX ;cargo en CR3 la direccion del page directory
MOV EAX, CR0
    OR EAX, 0x80000000 ;habilitamos paginacion
MOV CR0, EAX
    ; Imprimir mensaje de bienvenida

    ; Inicializar pantalla

    ; Inicializar el manejador de memoria

    ; Inicializar el directorio de paginas
    CALL mmu_inicializar

    CALL deshabilitar_pic
    CALL resetear_pic
    CALL habilitar_pic
    STI
```