UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Departamento de Ingeniería Electrónica

Técnicas Digitales III

Guía de Trabajos Prácticos

Primer cuatrimestre

Ciclo lectivo 2019 Plan 95A

TABLA DE CONTENIDO

La presente guía puede ser resuelta utilizando el lenguaje que considere más adecuado. La cátedra recomienda el uso de ensamblador (NASM) y C (C11).

1. INTERACCIÓN CON LAS HERRAMIENTAS

Instalar la máquina virtual ***Bochs*** siguiendo las instrucciones indicadas en http://wiki.electron.frba.utn.edu.ar/doku.php?id=td3:bochs.
Configurar la máquina virtual para simular una PC con 512MB de memoria RAM y 64kB de ROM y un procesador x86 genérico.

Objetivos conceptuales

I. *Familiarizarse con los comandos de **Bochs**.*
II. *Entender el funcionamiento de la imagen de la máquina virtual y su generación*
III. *Familiarizarse con la estructura de los archivos Makefile y de configuración de Bochs*

2. INICIALIZACIÓN BÁSICA UTILIZANDO SOLO ENSAMBLADOR CON ACCESO A 1MB

Escribir un programa que se ejecute en una ROM de 64kB y permita copiarse a sí mismo en cualquier zona de memoria. A tal fin se deberá implementar la función

void *td3_memcopy(void *destino, const void *origen, unsigned int num_bytes);

Para validar el correcto funcionamiento del programa, el mismo deberá copiarse en las direcciones indicadas a continuación y mediante Bochs verificar que la memoria se haya escrito correctamente.

i. 0x00000
ii. 0xF0000

Objetivos conceptuales

i. Familiarizarse con el lenguaje ASM y las herramientas asociadas (NASM).
ii. Familiarizarse con las herramientas de depuración provistas por el Bochs.
iii. Comprender el mapa de memoria del procesador.
iv. Identificar todos los registros y datos que utiliza el procesador para acceder a cada instrucción de código y dato. [1][2]

Referencias

[1] Volumen 1, Capítulo 3, sección 3, Intel® 64 and IA-32 Architectures Software Developer Manuals,
[2] Volumen 1, Capítulo 3, sección 4, Intel® 64 and IA-32 Architectures Software Developer Manuals,

3. INICIALIZACIÓN BÁSICA UTILIZANDO SOLO ENSAMBLADOR CON ACCESO A 4GB

Activar el mecanismo conocido como **A20 GATE** para acceder al mapa completo de memoria del procesador en modo real.
Adicionalmente agregar el código necesario a fin de que el programa pueda

a. Copiarse a sí mismo en la dirección 0x00000000 y ejecutarse desde dicha ubicación
b. Copiarse a 0x00300000 y finalizar estableciendo al procesador en estado ***halted*** en forma permanente
c. Establecer la pila en 0x1FFFB000

Objetivos conceptuales

i. Familiarizarse con el lenguaje ASM y las herramientas asociadas (NASM).
ii. Familiarizarse con las herramientas de depuración provistas por el Bochs.
iii. Comprender el mapa de memoria de la PC y las restricciones históricas.
iv. Identificar todos los registros y datos que utiliza el procesador para acceder a cada instrucción de código, dato y pila. [1][2]

4. INICIALIZACIÓN BÁSICA UTILIZANDO EL LINKER

Modificar el código del ejercicio anterior para satisfacer los siguientes requerimientos:

b. El programa debe situarse al inicio de la ROM (0xFFFF0000)
c. Copiarse y ejecutarse en las siguientes direcciones:
   i. 0x00000000
   ii. 0x00300000
   iii. Dirección a elección. En esta última debe finalizar la ejecución.

El mapa de memoria propuesto

Sección	Dirección inicial
Binario copiado 1	00000000h
Binario copiado 2	00300000h
Binario copiado 3	A elección
Pila	1FFFB000h
Secuencia inicialización ROM	FFFF0000h
Vector de reset	FFFFFFF0h

Objetivos conceptuales

I. Familiarizarse con el lenguaje ASM y las herramientas asociadas (NASM).
II. Familiarizarse con el "linker", su lenguaje de "script" y las herramientas asociadas (ld).
III. Identificar todos los registros y datos que utiliza el procesador para acceder a cada instrucción de código, dato y pila. [1][2]

Condiciones generales

A partir de este punto de la guía se recomienda fuertemente plantear el ejercicio con el siguiente esquema de ficheros.

Makefile o ***make.sh*** : comandos necesarios para construir el binario

linker.lds : *script* para el *linker*

bochs.cfg : configuración utilizada para el Bochs en cada ejercicio

init.s : solo el código necesario para inicializar al procesador en modo protegido y en ejercicios posteriores la paginación.

main.s : funcionalidad solicitada en cada ejercicio

functions.s : funciones auxiliares y/o frecuentemente implementadas en ensamblador

functions.c : funciones auxiliares y/o frecuentemente implementadas en C

sys_tables.s : tablas de sistema.

5. MODO PROTEGIDO 32BITS

En base al ejercicio anterior adecuarlo para que el mismo se ejecute en modo protegido 32bits.

a) Crear una estructura GDT mínima con modelo de segmentación FLAT [3][4][5]
b) En la zona denominada Núcleo solo debe copiarse código.
c) Definir una pila dentro del segmento de datos e inicializar el par de registros **SS:ESP** [3][4] adecuadamente. Realice la definición de forma dinámica de modo que pueda modificarse su tamaño y ubicación de manera simple.

El mapa de memoria para las diferentes secciones debe ser el siguiente

Sección	Dirección inicial
Rutinas	00000000h
Núcleo	00300000h
Pila	1FFFB000h
Secuencia inicialización ROM	FFFF0000h
Vector de reset	FFFFFFF0h

Objetivos conceptuales

I. *Comprender los requerimientos necesarios para que un programa pueda ejecutarse en modo protegido.*
II. *Identificar todos los registros y datos que utiliza el procesador para acceder a cada instrucción de código, dato y pila.*
III. *Analizar el esquema de direccionamiento entre modo protegido y real, identificando diferencias y similitudes.*
IV. *Comprender el significado y la implicancia de cada campo de las tablas de descriptores y los mecanismos de protección que activan.*

Referencias

[3] Volumen 3, Capítulo 2, sección 2, Intel® 64 and IA-32 Architectures Software Developer Manuals
[4] Volumen 3, Capítulo 3, sección 2, Intel® 64 and IA-32 Architectures Software Developer Manuals,

[5] Volumen 3, Capítulo 3, sección 4, Intel® 64 and IA-32 Architectures Software Developer Manuals,

6. TECLADO POR ENCUESTA (*POLLING*)

Modificar el ejercicio anterior implementando las siguientes funcionalidades

a) Almacenar en una tabla de 64kB las teclas presionadas que corresponden a dígitos hexadecimales. A tal fin se debe realizar una rutina que encueste el buffer de teclado (dirección de E/S 0x60 datos y 0x64 estado/comando) en forma periódica. Al llegar al final de la tabla se sobrescriben los valores iniciales.
b) Al presionar "S" el programa finaliza estableciendo al procesador en estado ***halted*** en forma permanente.
c) Establezca todos los datos del programa en *Datos*.

El mapa de memoria debe ser el siguiente

Sección	Dirección inicial
Rutina de teclado	00000000h
Núcleo	00300000h
Tabla de dígitos	00310000h
Datos	003E0000h
Pila	1FFFE000h
Secuencia inicialización ROM	FFFF0000h
Vector de reset	FFFFFFF0h

Objetivos conceptuales

i. Identificar el esquema de direccionamiento de los periféricos (E/S, Memoria).

7. CONFIGURACIÓN DEL SISTEMA DE INTERRUPCIONES

Tomando el ejercicio anterior, agregar una IDT [6][7] capaz de manejar todas las excepciones del procesador e interrupciones mapeadas por ambos PIC, es decir de la 0x00 hasta el tipo 0x2F y cumpla con los siguientes requerimientos.

a) Configurar el PIC maestro y esclavo de manera que utilicen el rango de tipos de interrupción **0x20-0x27** y **0x28-0x2F**, respectivamente.
b) Inicializar el registro de máscaras [8], de modo que estén deshabilitadas, todas las interrupciones de hardware en ambos PIC's.
c) Implementar en todas las excepciones una rutina que permita identificar el número de excepción generada y finalice deteniendo la ejecución de instrucciones mediante la instrucción "**hlt**". Se propone como método de identificación almacenar en dx el número de excepción.
d) Generar de **manera apropiada** [9] (no emulándolas por interrupciones de software) para comprobar su funcionamiento las excepciones #DE, #UD, #DF y #GP. Se recomienda asociar la generación de cada una de las excepciones indicadas previamente a la pulsación de diferentes teclas. A tal fin se propone la siguiente correspondencia: #DE=**Y**, #UD=**U**, #DF=**I**, #GP=**O**.
e) La IDT se debe ubicar en *Tablas de sistema*

El mapa de memoria debe ser el siguiente:

Sección	Dirección inicial
Rutina de teclado e ISR	00000000h
Tablas de sistema	00100000h
Núcleo	00300000h
Tabla de dígitos	00310000h
Datos	003E0000h
Pila	1FFFB000h
Secuencia inicialización ROM	FFFF0000h
Vector de reset	FFFFFFF0h

Objetivos conceptuales

I. *Comprender el esquema de numeración de los vectores de interrupción y su asociación con la decodificación realizada por los PIC.*
II. *Entender la diferencia entre IRQ y tipo de interrupción.*
III. *Identificar los tipos de excepciones y las condiciones que las generan.*
IV. *Comprender el significado y la implicancia de cada campo de las tablas de descriptores y los mecanismos de protección que activan.*
V. *Analizar la gestión de la pila realizada por cada excepción.*
VI. *Identificar todos los registros y datos que utiliza el procesador para acceder al controlador de una excepción.*
VII. *Identificar las diferencias principales entre una excepción y una interrupción*

Referencias

[6] Volumen 3, Capítulo 6, sección 10, Intel® 64 and IA-32 Architectures Software Developer Manuals,
[7] Volumen 3, Capítulo 6, sección 11, Intel® 64 and IA-32 Architectures Software Developer Manuals,
[8] Volumen 3, Capítulo 2, sección 3, Intel® 64 and IA-32 Architectures Software Developer Manuals,
[9] Volumen 3, Capítulo 6, sección 15, Intel® 64 and IA-32 Architectures Software Developer Manuals.

8. INTERRUPCIONES DE HARDWARE

Utilizando lo realizado anteriormente, implementar las siguientes modificaciones:

a) La rutina de adquisición de teclas debe realizarse en el controlador de teclado (**IRQ1** [10]). Se debe tener en cuenta que por cada presión de una tecla se producen dos interrupciones, una por el ***make code*** y otra por el ***break code.***
b) Los dígitos correspondientes al alfabeto hexadecimal conformarán un número de 64bits, es decir si se presionan las teclas 1234ABCD, se debe almacenar en la tabla de dígitos como una entrada que contiene al número 000000001234ABCDh. Cada nuevo número se insertará en la tabla cuando se presione ENTER. Por razones de simplicidad el buffer circular de teclado dispondrá de una longitud de 9 bytes. En la tabla se ingresarán los últimos 16 dígitos hexadecimales presionados al pulsar ENTER (123JHAB4567CDEFLMN.ENTER equivale a 000123AB4567CDEFh). Si al presionar

ENTER se han ingresado menos de 8 Bytes, se completarán con ceros en las posiciones MSB (1E.ENTER equivale 000000000000001Eh).

c) Escribir el controlador del temporizador (**IRQ0** [10]) de modo que interrumpa cada 100ms. Verifique el correcto funcionamiento almacenando en alguna dirección de *Datos* el número de veces que se produce la interrupción. Tenga en cuenta que la implementación del *timer tick* en ***Bochs*** no garantiza ejecución del tipo tiempo real, es decir observará una falta de correspondencia temporal entre la unidad de tiempo calculada y la que ***Bochs*** ejecuta en la práctica

El mapa de memoria es el siguiente:

Sección	Dirección inicial
ISR	00000000h
Tablas de sistema	00100000h
Núcleo	00300000h
Tabla de dígitos	00310000h
Datos	003E0000h
Pila	1FFFB000h
Secuencia inicialización ROM	FFFF0000h
Vector de reset	FFFFFFF0h

Objetivos conceptuales

I. *Identificar todos los registros y datos que utiliza el procesador para acceder al controlador de una interrupción.*
II. *Comprender la implicancia del término "enmascarable"*
III. *Analizar la gestión de la pila realizada por una interrupción y compararla con la de una excepción.*

Referencias

[10] Volumen 3, Capítulo 6, sección 12, Intel® 64 and IA-32 Architectures Software Developer Manuals.

9. RUTINA TEMPORIZADA Y CONTROLADOR DE VIDEO

Modificar el programa desarrollado hasta el momento considerando las siguientes consignas:

a) Implementar una rutina que sume todos los números almacenados en la tabla de dígitos, presente el resultado parcial en pantalla y el cómputo final sea almacenado en alguna variable situada en *Datos*. La rutina debe cumplir los siguientes requerimientos:
   a. Ejecutarse cada 1000ms.
   b. **No implementarse dentro de la IRQ0.**
   c. Situarse en la zona de *Tarea 1.*
   d. Mientras no se ejecute establecer al procesador en estado ***halted.***

b) Adecuar el código y el *linker script* para satisfacer el siguiente mapa de memoria

Sección	Dirección inicial
ISR	00000000h
Tablas de sistema	00100000h
Núcleo	00300000h
Tabla de dígitos	00310000h
Tarea 1	00320000h
Datos	003E0000h
Pila	1FFFB000h
Secuencia inicialización ROM	FFFF0000h
Vector de reset	FFFFFFF0h

Objetivos conceptuales

I. *Analizar la implementación del direccionamiento realizada por el compilador y el linker.*
II. *Asociar los esquemas de direccionamiento del procesador con los utilizados por el programador y las herramientas (compilador y linker).*

10. PAGINACIÓN BÁSICA

Tomando como base al ejercicio anterior implementar un sistema de paginación [11] en modo *identity mapping* y adecuarlo a los siguientes lineamientos:

a) Estructurar el programa de forma tal que disponga de las siguientes secciones (la denominación se realiza acorde al estándar ELF) con sus respectivas propiedades:
   I. **Sección de código (TEXT):** no debe contener ningún tipo de dato/variable.
   II. **Sección de datos inicializados (DATA):** debe subdividirse en una subsección de solo lectura y otra de lectura/escritura.
   III. **Sección de datos no inicializados (BSS):**
b) Implementar un controlador básico de #PF que indique el motivo de la excepción.
c) El tamaño del binario compactado no debe superar 64kB.
d) El mapa de memoria luego de la expansión del binario debe cumplir con el siguiente esquema:

Sección	Dirección física
ISR	00000000h
Tablas de sistema	00100000h
Tablas de paginación	00110000h
Núcleo	00400000h
Tabla de dígitos	00410000h
TEXT Tarea 1	00420000h
BSS Tarea 1	00421000h
DATA Tarea 1	00422000h
Datos	004E0000h
Pila	1FFFB000h
Pila Tarea 1	1FFFE000h
Secuencia inicialización ROM	FFFF0000h
Vector de reset	FFFFFFF0h

Utilizar alguna herramienta interpretación de ficheros binario para analizar los datos de posicionamiento en memoria.

Objetivos conceptuales

i. Identificar los esquemas de direccionamiento del procesador.
ii. Analizar la implementación del direccionamiento realizada por el compilador y el linker.
iii. Asociar los esquemas de direccionamiento del procesador con los utilizados por el programador y las herramientas (compilador y linker).
iv. Relacionar las características de las diferentes secciones con los tipos de memoria y el mapa de direcciones de la PC.
v. Entender el funcionamiento del "linker script".

Referencias

[11] Volumen 3, Capítulo 4, sección 1, Intel® 64 and IA-32 Architectures Software Developer Manuals.

11. PAGINACIÓN AVANZADA

Modificar el esquema de paginación del ejercicio anterior para satisfacer el siguiente mapa de memoria.

Sección	Dirección física inicial	Dirección lineal inicial
ISR	00000000h	00000000h
VIDEO	000B8000h	00010000h
Tablas de sistema	00100000h	00100000h
Tablas de paginación	00110000h	00110000h
Núcleo	00400000h	00400000h
Tabla de dígitos	00410000h	00410000h
TEXT Tarea 1	00421000h	00510000h
BSS Tarea 1	00422000h	00511000h
DATA Tarea 1	00423000h	00512000h
Datos	004E0000h	004E0000h
Pila	1FFFB000h	1FFFB000h
Pila Tarea 1	1FFFE000h	00413000h
Secuencia inicialización ROM	FFFFF000h	FFFF0000h
Vector de reset	FFFFFFF0h	FFFFFFF0h

Objetivos conceptuales

i. Comprender en profundidad el mecanismo de paginación.
ii. Dominar la herramienta "linker script".

12. PAGINACIÓN REAL

En base al código anterior modificar las siguientes funcionalidades:

a) La rutina de suma (Tarea 1) debe intentar leer en la dirección que se obtenga como resultado de la suma. En caso de que dicho número supere la RAM del sistema (512MB) se omitirá la lectura.
b) El controlador de **#PF** al detectar que el error se debe a una página no presente, deberá asignar a la dirección que produjo el error una nueva página a partir de la dirección física 0x08000000h.

Tenga en cuenta que las estructuras de paginación deberán completarse en forma dinámica.

Objetivos conceptuales

I. Comprender en profundidad el controlador la excepción de Page Fault.

Referencias

Se recomienda leer nuevamente:
Volumen 3, Capítulo 6, sección 15, Intel® 64 and IA-32 Architectures Software Developer Manuals, la descripción de la excepción de Page Fault (14).

13. CONMUTACIÓN DE TAREAS

Incorpore al programa desarrollado hasta el momento una capacidad mínima de administración de tareas [11]. Para ello se requiere, agregar las siguientes prestaciones:

a) Implementar 3 tareas a saber
   i. **Suma** (2 tareas). Estas tareas de ejecutarán cada 100 y 200 ms. respectivamente. En todos los casos utilizarán los mismos datos como sumandos (Tabla de dígitos), presentado el resultado en pantalla y al finalizar debe establecer al procesador en estado *halted*. Deshabilitar (**no borrar**) el código de generación de PF.
   ii. **Idle** (1 tarea). Su única función es establecer al procesador en estado *halted y se debe ejecutar cuando ninguna otra tarea se encuentre en ejecución*.
b) Modificar el valor del temporizador 0 del PIT, para que genere una interrupción cada 10 mseg aproximadamente.
c) Modificar el controlador de la interrupción 32 (**IRQ0, timer tick**), para que actué como conmutador de tareas (**scheduler**). Diseñe dicho mecanismo [12] para que despache las tareas en forma secuencial. El mecanismo de conmutación de contextos deberá implementarlo finalmente de forma manual (transitoriamente puede analizar el mecanismo automático provisto por el procesador) [13][14].
d) Modificar el mapa de memoria a alguno de los siguientes esquemas (indicar el esquema utilizado en el código)

OPCIÓN A

Sección	Dirección física inicial	Dirección lineal inicial
ISR	00000000h	00000000h
VIDEO	000B8000h	00010000h
Tablas de sistema	00100000h	00100000h
Tablas de paginación	00110000h	00110000h
Núcleo	00500000h	00400000h
Tabla de dígitos	00510000h	00510000h
TEXT Tarea 0	00501000h	00500000h
BSS Tarea 0	00502000h	00601000h
DATA Tarea 0	00503000h	00602000h
Tabla de dígitos	00510000h	00510000h
TEXT Tarea 1	00521000h	00610000h
BSS Tarea 1	00522000h	00611000h
DATA Tarea 1	00523000h	00612000h
TEXT Tarea 2	00531000h	00620000h
BSS Tarea 2	00532000h	00621000h
DATA Tarea 2	00533000h	00622000h
Datos	005E0000h	005E0000h
Pila Núcleo	1FFFB000h	1FFFB000h
Pila Tarea 0	1FFFC000h	00603000h
Pila Tarea 1	1FFFE000h	00613000h
Pila Tarea 2	1FFFD000h	00623000h
Secuencia inicialización ROM	FFFFF000h	FFFF0000h
Vector de reset	FFFFFFF0h	FFFFFFF0h

OPCIÓN B

Sección	Dirección física inicial	Dirección lineal inicial
ISR	00000000h	00000000h
VIDEO	000B8000h	00010000h
Tablas de sistema	00100000h	00100000h
Tablas de paginación	00110000h	00110000h
Núcleo	00500000h	00500000h
Tabla de dígitos	00510000h	00510000h
TEXT Tarea 0	00501000h	00610000h
BSS Tarea 0	00502000h	00611000h
DATA Tarea 0	00503000h	00612000h
Tabla de dígitos	00510000h	00610000h
TEXT Tarea 1	00521000h	00610000h
BSS Tarea 1	00522000h	00611000h
DATA Tarea 1	00523000h	00612000h
TEXT Tarea 2	00531000h	00610000h
BSS Tarea 2	00532000h	00611000h
DATA Tarea 2	00533000h	00612000h
Datos	005E0000h	005E0000h
Pila Núcleo Tarea 0	1FFF8000h	00614000h
Pila Núcleo Tarea 2	1FFF9000h	00614000h
Pila Núcleo Tarea 1	1FFFA000h	00614000h
Pila Núcleo	1FFFB000h	1FFFB000h
Pila Usuario Tarea 0	1FFFC000h	00613000h
Pila Usuario Tarea 2	1FFFD000h	00613000h
Pila Usuario Tarea 1	1FFFE000h	00613000h
Secuencia inicialización ROM	FFFFF000h	FFFF0000h
Vector de reset	FFFFFFF0h	FFFFFFF0h

Objetivos conceptuales

i. *Identificar todos los registros y datos utilizados para realizar la conmutación de tarea.*
ii. *Comprender que el espacio de direcciones observado por cada tarea es completamente independiente de la dirección física y la traducción solo es conocida por el SO.*

Referencias

[12] Volumen 3, Capítulo 7, sección 1, Intel® 64 and IA-32 Architectures Software Developer Manuals,
[13] Volumen 3, Capítulo 7, sección 3, Intel® 64 and IA-32 Architectures Software Developer Manuals,
[14] Volumen 3, Capítulo 7, sección 2, Intel® 64 and IA-32 Architectures Software Developer Manuals,
[15] Volumen 3, Capítulo 7, sección 7, Intel® 64 and IA-32 Architectures Software Developer Manuals.

14. SIMD

Modificar la Tarea 2 para que realice la suma aritmética por desborde en tamaño word y la Tarea 1 para que lo realice en tamaño quadruple word
Implementar el soporte necesario para el resguardo de los registros MMX/SSE mediante la excepción 7 (***#NM***) [15][16], así como también realizar las modificaciones correspondientes en la función de cambio de contexto.

Objetivos conceptuales

i. *Identificar todos los registros y datos que utiliza el procesador para verificar si se ha utilizado una instrucción SIMD.*
ii. *Comprender los diferentes tipos de aritmética utilizados por el procesador.*

Referencias

[16] Volumen 3, Capítulo 6, sección 15, Intel® 64 and IA-32 Architectures Software Developer Manuals,
[17] Volumen 3, Capítulo 12, Intel® 64 and IA-32 Architectures Software Developer Manuals,

15. NIVELES DE PRIVILEGIO

En base a lo elaborado anteriormente implementar un sistema que permita manejar niveles de privilegio. A tal fin modificar/agregar los siguientes ítems:

a) La GDT debe contemplar los descriptores de nivel 3 (**PL=11** usuario) [17], tanto para código como para datos.
b) Adecuar las diferentes entradas de la tabla de paginación según corresponda a usuario o supervisor, verificando además que los permisos de lectura y escritura sean consistentes con la sección asociada.
c) Las tareas 1 y 2 deben ejecutarse en nivel 3. Analizar si es necesario disponer de una pila por cada tarea y realizar las modificaciones pertinentes acorde a su respuesta.
d) Diseñar un mecanismo apropiado de acceso para las siguientes funciones de sistema (*system calls*). Utilice el vector **80h** para los servicios, o bien un **CALL FAR**.
    i. ***void td3_halt(void);***
    ii. ***unsigned int td3_read(void *buffer, unsigned int num_bytes);***
    iii. ***unsigned int td3_print(void *buffer, unsigned int num_bytes);***
e) Modificar el mapa de memoria a alguno de los siguientes esquemas (indicar el esquema utilizado en el código)

OPCIÓN A

Sección	Dirección física inicial	Dirección lineal inicial
ISR	00000000h	00000000h
VIDEO	000B8000h	00010000h
Tablas de sistema	00100000h	00100000h
Tablas de paginación	00110000h	00110000h
Núcleo	00500000h	00500000h
Tabla de dígitos	00510000h	00510000h
TEXT Tarea 0	00501000h	00600000h
BSS Tarea 0	00502000h	00601000h
DATA Tarea 0	00503000h	00602000h
Tabla de dígitos	00510000h	00510000h
TEXT Tarea 1	00521000h	00610000h
BSS Tarea 1	00522000h	00611000h
DATA Tarea 1	00523000h	00612000h
TEXT Tarea 2	00531000h	00620000h
BSS Tarea 2	00532000h	00621000h
DATA Tarea 2	00533000h	00622000h
Datos	005E0000h	005E0000h
Pila Núcleo	1FFFB000h	1FFFB000h
Pila Tarea 0	1FFFC000h	00603000h
Pila Tarea 1	1FFFE000h	00613000h
Pila Tarea 2	1FFFD000h	00623000h
Secuencia inicialización ROM	FFFFF000h	FFFF0000h
vector de reset	FFFFFFF0h	FFFFFFF0h

Objetivos conceptuales

i. *Identificar todos los registros y datos que utiliza el procesador para verificar si es posible efectuar una conmutación de privilegio.*
ii. *Analizar en qué momento actúa cada unidad de direccionamiento para validar el acceso a una región de memoria determinada*
iii. *Identificar todos los registros y datos que utiliza el procesador para verificar si es posible acceder a un dato desde un código de cierto nivel de privilegio.*
iv. *Analizar la gestión de la pila realizada para una interrupción al conmutar de nivel de privilegio.*
v. *Reconocer los campos necesarios para identificar el privilegio de una sección en las tablas de paginación.*
vi. *Analizar la gestión de la pila realizada para una interrupción.*
vii. *Comprender qué información se almacena en el espacio de contexto de cada.*

OPCIÓN B

Sección	Dirección física inicial	Dirección lineal inicial
ISR	00000000h	00000000h
VIDEO	000B8000h	00010000h
Tablas de sistema	00100000h	00100000h
Tablas de paginación	00110000h	00110000h
Núcleo	00500000h	00500000h
Tabla de dígitos	00510000h	00510000h
TEXT Tarea 0	00501000h	00610000h
BSS Tarea 0	00502000h	00611000h
DATA Tarea 0	00503000h	00622000h
Tabla de dígitos	00510000h	00510000h
TEXT Tarea 1	00521000h	00610000h
BSS Tarea 1	00522000h	00611000h
DATA Tarea 1	00523000h	00612000h
TEXT Tarea 2	00531000h	00610000h
BSS Tarea 2	00532000h	00611000h
DATA Tarea 2	00533000h	00612000h
Datos	005E0000h	005E0000h
Pila Núcleo Tarea 1	1FFFA000h	00614000h
Pila Núcleo Tarea 0	1FFF8000h	00614000h
Pila Núcleo Tarea 2	1FFF9000h	00614000h
Pila Núcleo	1FFFB000h	1FFFB000h
Pila Usuario Tarea 0	1FFFC000h	00613000h
Pila Usuario Tarea 2	1FFFD000h	00613000h
Pila Usuario Tarea 1	1FFFE000h	00613000h
Secuencia inicialización ROM	FFFFF000h	FFFF0000h
vector de reset	FFFFFFF0h	FFFFFFF0h

Referencias

[18] Volumen 3, Capítulo 5, Intel® 64 and IA-32 Architectures Software Developer Manuals.

16. PRIMER RECUPERATORIO

Modificar la Tarea 1 de forma tal que interprete el número ingresado como un vector y aplique sobre la lista la operación de producto escalar. A tal fin los primeros 32 bits corresponderán a la parte real y los restantes a la imaginaria (formato cartesiano 2D).

17. SEGUNDO RECUPERATORIO

Agregar dos tareas adicionales que calculen el máximo valor de la tabla de dígitos, la tarea 3 lo hará en C y la tarea 4 mediante SIMD. En ambos casos de se deberá presentar en pantalla el tiempo empleado.

Sección	Dirección física inicial	Dirección lineal inicial
ISR	00000000h	00000000h
VIDEO	000B8000h	00010000h
Tablas de sistema	00100000h	00100000h
Tablas de paginación	00110000h	00110000h
Núcleo	00500000h	00500000h
Tabla de dígitos	00510000h	00510000h
TEXT Tarea 0	00501000h	00600000h
BSS Tarea 0	00502000h	00601000h
DATA Tarea 0	00503000h	00602000h
Tabla de dígitos	00510000h	00510000h
TEXT Tarea 1	00521000h	00610000h
BSS Tarea 1	00522000h	00611000h
DATA Tarea 1	00523000h	00612000h
TEXT Tarea 2	00531000h	00620000h
BSS Tarea 2	00532000h	00621000h
DATA Tarea 2	00533000h	00622000h
TEXT Tarea 3	00541000h	00630000h
BSS Tarea 3	00542000h	00631000h
DATA Tarea 3	00543000h	00632000h
TEXT Tarea 4	00551000h	00640000h
BSS Tarea 4	00552000h	00641000h
DATA Tarea 4	00553000h	00642000h
Datos	005E0000h	005E0000h
Pila Tarea 3	1FFF9000h	00633000h
Pila Tarea 4	1FFFA000h	00643000h
Pila Núcleo	1FFFB000h	1FFFB000h
Pila Tarea 0	1FFFC000h	00603000h
Pila Tarea 1	1FFFE000h	00613000h
Pila Tarea 2	1FFFD000h	00623000h
Secuencia inicialización ROM	FFFFF000h	FFFF0000h
vector de reset	FFFFFFF0h	FFFFFFF0h

Sección	Dirección física inicial	Dirección lineal inicial
ISR	00000000h	00000000h
VIDEO	000B8000h	00010000h
Tablas de sistema	00100000h	00100000h
Tablas de paginación	00110000h	00110000h
Núcleo	00500000h	00500000h
Tabla de dígitos	00510000h	00510000h
TEXT Tarea 0	00501000h	00610000h
BSS Tarea 0	00502000h	00611000h
DATA Tarea 0	00503000h	00622000h
Tabla de dígitos	00510000h	00510000h
TEXT Tarea 1	00521000h	00610000h
BSS Tarea 1	00522000h	00611000h
DATA Tarea 1	00523000h	00612000h
TEXT Tarea 2	00531000h	00610000h
BSS Tarea 2	00532000h	00611000h
DATA Tarea 2	00533000h	00612000h
TEXT Tarea 3	00541000h	00610000h
BSS Tarea 3	00542000h	00611000h
DATA Tarea 3	00543000h	00612000h
TEXT Tarea 4	00551000h	00610000h
BSS Tarea 4	00552000h	00611000h
DATA Tarea 4	00553000h	00612000h
Datos	005E0000h	005E0000h
Pila Núcleo Tarea 3	1FF01000h	00614000h
Pila Núcleo Tarea 4	1FF02000h	00614000h
Pila Usuario Tarea 3	1FF0E000h	00613000h
Pila Usuario Tarea 4	1FF0F000h	00613000h
Pila Núcleo Tarea 1	1FFFA000h	00614000h
Pila Núcleo Tarea 0	1FFF8000h	00614000h
Pila Núcleo Tarea 2	1FFF9000h	00614000h
Pila Núcleo	1FFFB000h	1FFFB000h
Pila Usuario Tarea 0	1FFFC000h	00613000h
Pila Usuario Tarea 2	1FFFD000h	00613000h
Pila Usuario Tarea 1	1FFFE000h	00613000h
Secuencia inicialización ROM	FFFFF000h	FFFF0000h
vector de reset	FFFFFFF0h	FFFFFFF0h