UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Departamento de Ingeniería Electrónica

Técnicas Digitales III

Guía de Trabajos Prácticos

Primer cuatrimestre

Ciclo lectivo 2018 **Plan 95A**

TABLA DE CONTENIDO

La presente guía puede ser resuelta utilizando el lenguaje que considere más adecuado. La catedra 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 por consola.*
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 si 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 la memoria se halla 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 de la PC y las restricciones de cada modo.
iv. Identificar todos los registros y datos que utiliza el procesador para acceder a cada instrucción de código y dato. [1][2]

Referencias

[1] Volúmen 1, Capítulo 3, sección 3, Intel® 64 and IA-32 Architectures Software Developer Manuals,
[2] Volúmen 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 si 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 de cada modo.
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 ENLAZADOR

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

a. El programa debe situarse al inicio de la ROM (0xFFFF0000)
b. 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	FFFFF000h
Vector de reset	FFFFFFF0h

Objetivos conceptuales

I. Familiarizarse con el lenguaje ASM y las herramientas asociadas (NASM).
II. Familiarizarse con el enlazador 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]

5. MODO PROTEGIDO 32BITS

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

a) Crear una estructura GDT mínima con modelo de segmentación FLAT [3][4][5] y
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.
d) Plantee el ejercicio con el siguiente esquema de archivos:
Makefile o make.sh : comandos necesarios para construir el binario
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
sys_tables.s : tablas de sistema.

El mapa de memoria propuesto

Sección	Dirección inicial
Rutina de teclado	00000000h
Núcleo	00300000h
Pila	1FFFB000h
Secuencia inicialización ROM	FFFFF000h
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] Volúmen 3, Capítulo 2, sección 2, Intel® 64 and IA-32 Architectures Software Developer Manuals
[4] Volúmen 3, Capítulo 3, sección 2, Intel® 64 and IA-32 Architectures Software Developer Manuals,
[5] Volúmen 3, Capítulo 3, sección 4, Intel® 64 and IA-32 Architectures Software Developer Manuals,

6. TECLADO POR ENCUESTA

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) 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 propuesto

Sección	Dirección inicial
Rutina de teclado	00000000h
Núcleo	00300000h
Tabla de dígitos	00310000h
Datos	003E0000h
Pila	1FFFE000h
Secuencia inicialización ROM	FFFFF000h
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) las siguientes excepciones para comprobar su funcionamiento (#DE, #UD, #DF, #GP, #PF). Se recomienda asociar la generación de cada una de las excepciones indicadas previamente a las siguientes teclas (#DE=Y, #UD=U, #DF=I, #GP=O, #PF=P).
e) La IDT se debe ubicar en *Tablas de sistema*

El mapa de memoria propuesto

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	FFFFF000h
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] Volúmen 3, Capítulo 6, sección 10, Intel® 64 and IA-32 Architectures Software Developer Manuals,
[7] Volúmen 3, Capítulo 6, sección 11, Intel® 64 and IA-32 Architectures Software Developer Manuals,
[8] Volúmen 3, Capítulo 2, sección 3, Intel® 64 and IA-32 Architectures Software Developer Manuals,
[9] Volúmen 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 1234ABCDh. 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 8 dígitos hexadecimales presionados al pulsar ENTER (123JGHAB.ENTER equivale a 000123ABh). Si al presionar ENTER se han ingresado menos de 8Bytes, se completaran con ceros en las posiciones MSB (1E.ENTER equivale 0000001Eh).
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.

El mapa de memoria propuesto

Sección	Dirección inicial
ISR	00000000h
Tablas de sistema	00100000h
Núcleo	00300000h
Tabla de dígitos	00310000h
Datos	003E0000h
Pila	1FFFB000h
Secuencia inicialización ROM	FFFFF000h
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] Volúmen 3, Capítulo 6, sección 12, Intel® 64 and IA-32 Architectures Software Developer Manuals.

9. TAREA TEMPORIZADA

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 y su resultado sea almacenados en alguna variable situada en *Datos*. La rutina debe cumplir los siguientes requerimientos:
   a. Ejecutarse cada 500ms.
   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	FFFFF000h
Vector de reset	FFFFFFF0h

Objetivos conceptuales

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

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) El tamaño del binario compactado no debe superar 64kB.
c) El mapa de memoria luego de la expansión del binario debe cumplir con el siguiente esquema:

Sección	Dirección inicial
ISR	00000000h
Tablas de sistema	00100000h
Tablas de paginación	00110000h
Núcleo	00300000h
Tabla de dígitos	00310000h
TEXT Tarea 1	00320000h
BSS Tarea 1	00321000h
DATA Tarea 1	00322000h
Datos	003E0000h
Pila	1FFFB000h
Pila Tarea 1	1FFFE000h
Secuencia inicialización ROM	FFFFF000h
Vector de reset	FFFFFFF0h

Utilizar la herramienta de vuelco de 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 enlazador.*
iii. *Asociar los esquemas de direccionamiento del procesador con los utilizados por el programador y las herramientas (compilador y enlazador).*
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] Volúmen 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
Tablas de sistema	00100000h	00100000h
Tablas de paginación	00110000h	00110000h
Núcleo	00300000h	00300000h
Tabla de dígitos	00310000h	00310000h
TEXT Tarea 1	00321000h	00410000h
BSS Tarea 1	00322000h	00411000h
DATA Tarea 1	00323000h	00412000h
Datos	003E0000h	003E0000h
Pila	1FFFB000h	1FFFB000h
Pila Tarea 1	1FFFE000h	00413000h
Secuencia inicialización ROM	FFFFF000h	FFFFF000h
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 adición. 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 10000000h.

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:
Volúmen 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 200ms respectivamente. En todos los casos utilizarán los mismos datos como sumandos (Tabla de dígitos), pero el resultado debe ser almacenado en su propia BSS 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 mapa de memoria acorde a:

Sección	Dirección física inicial	Dirección lineal inicial
ISR	00000000h	00000000h
Tablas de sistema	00100000h	00100000h
Tablas de paginación	00110000h	00110000h
Núcleo	00300000h	00300000h
TEXT Tarea 0	00301000h	00400000h
BSS Tarea 0	00302000h	00401000h
DATA Tarea 0	00303000h	00402000h
Tabla de dígitos	00310000h	00310000h
TEXT Tarea 1	00321000h	00410000h
BSS Tarea 1	00322000h	00411000h
DATA Tarea 1	00323000h	00412000h
TEXT Tarea 2	00331000h	00420000h
BSS Tarea 2	00332000h	00421000h
DATA Tarea 2	00333000h	00422000h
Datos	003E0000h	003E0000h
Pila	1FFFB000h	1FFFB000h
Pila Tarea 0	1FFFC000h	00403000h
Pila Tarea 2	1FFFD000h	00423000h
Pila Tarea 1	1FFFE000h	00413000h
Secuencia inicialización ROM	FFFF0000h	FFFFF000h
Vector de reset	FFFFFFF0h	FFFFFFF0h

c) Modificar el valor del temporizador 0 del PIT, para que genere una interrupción cada 10 mseg aproximadamente.

d) 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].

Objetivos conceptuales

i. *Identificar todos los registros y datos que utiliza el procesador 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 sistema operativo.*

Referencias

[12] Volúmen 3, Capítulo 7, sección 1, Intel® 64 and IA-32 Architectures Software Developer Manuals,
[13] Volúmen 3, Capítulo 7, sección 3, Intel® 64 and IA-32 Architectures Software Developer Manuals,
[14] Volúmen 3, Capítulo 7, sección 2, Intel® 64 and IA-32 Architectures Software Developer Manuals,
[15] Volúmen 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 byte y la Tarea 1 para que lo realice en tamaño doublé 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] Volúmen 3, Capítulo 6, sección 15, Intel® 64 and IA-32 Architectures Software Developer Manuals,
[17] Volúmen 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 debe 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 **FAR CALL**.
    i. ***void td3_halt(void);***
    ii. ***unsigned int td3_read(void *buffer, unsigned int num_bytes);***

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 que información se almacena en el espacio de contexto de cada.*

Referencias

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

16. PRIMER RECUPERATORIO

Agregar dos tareas adicionales de suma, una de las cuales la realizará en formato saturado signado y la otra en saturado no signado. En ambos casos el tamaño es de a byte. Las tareas se deberán agregar en utilizando las siguientes entradas en el mapa de memoria

TEXT Tarea 3	00341000h	00430000h
BSS Tarea 3	00342000h	00431000h
DATA Tarea 3	00343000h	00432000h
TEXT Tarea 4	00351000h	00440000h
BSS Tarea 4	00352000h	00441000h
DATA Tarea 4	00353000h	00442000h
Pila Tarea 3	1FFFA000h	00433000h
Pila Tarea 4	1FFFB000h	00443000h

17. SEGUNDO 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 32bits corresponderán a la parte real y los restantes a la imaginaria (formato cartesiano 2D).