UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
Facultad de Ingeniería, Arquitectura y Diseño

Ingeniero en Software y Tecnologías Emergentes

Manual de Talleres de la materia Organización de Computadoras

AUTOR(ES)
Jonatan Crespo Ragland

Taller No. 11 Interrupciones

Objetivo: Desarrollar el uso de interrupciones en ensamblador x86

Fundamentos teóricos del taller

- Lenguaje Ensamblador x86

Instrucciones para el desarrollo del taller

1. Desarrolla los siguientes puntos

Recursos

1. Apuntes de clase.
2. Fuentes bibliográficas.
3. Lápiz.
4. Equipo de cómputo.

Tiempo e instrucciones de entrega

Duración: 2 horas.

● Desarrollar lo siguiente en su cuaderno o computadora:

Qué es una interrupción en ensamblador:
Una interrupción es un evento que detiene temporalmente la ejecución normal del procesador para atender una tarea especial. La CPU guarda su estado, ejecuta una rutina especial llamada manejador de interrupción y luego regresa al punto donde se quedó. Sirven para manejar dispositivos, errores y servicios del sistema.

Qué son las interrupciones de hardware:
Son interrupciones generadas por dispositivos físicos, como el teclado, el temporizador o el disco duro. No las genera un programa sino el hardware. Cuando ocurre, el dispositivo envía una señal a la CPU (IRQ) para que atienda el evento.

Qué son las interrupciones de software:
Son interrupciones generadas por instrucciones dentro de un programa, usando la instrucción “int”. Se utilizan para pedir servicios al sistema operativo,

como imprimir, leer entrada o terminar un programa. Ejemplos comunes son int 21h en DOS e int 80h en Linux.

Diferencia entre interrupción enmascarable y no enmascarable: Una interrupción enmascarable sí puede bloquearse temporalmente con instrucciones como CLI. Las interrupciones no enmascarables (NMI) no pueden bloquearse y siempre deben atenderse porque indican fallos graves, como errores de memoria o fallas de hardware.

Tres interrupciones de hardware en x86:

1. IRQ0 → INT 08h: Interrupción del temporizador. Ocurre varias veces por segundo para controlar el reloj del sistema y la multitarea.

2. IRQ1 → INT 09h: Interrupción del teclado. Se activa cuando se presiona una tecla y envía el scan code al sistema.

3. IRQ14 → INT 0Eh: Interrupción del controlador de disco duro IDE. Indica que una operación de lectura o escritura ha terminado.

Tres interrupciones de software en x86:

- INT 21h: Interrupción del sistema DOS. Ofrece servicios como leer teclado, imprimir texto, manejar archivos y terminar programas.

- INT 10h: Interrupción de video BIOS. Se usa para imprimir caracteres en pantalla, mover el cursor o cambiar el modo de video.

- INT 80h: Interrupción de Linux para realizar llamadas al sistema. Permite ejecutar funciones como escribir en pantalla, abrir archivos, crear procesos, etc.

- **Modifica el siguiente programa en ensamblador para poder manejar la división entre 0 utilizando solamente saltos condicionales. One Compiler no permite un manejo de interrupciones en tiempo real, por lo que tendremos que simular su funcionamiento (también puedes encontrar el**

código en anexos). Este código deberá estar en su repositorio de código.
https://onecompiler.com/assembly/4452sffxe

```
section .data
 msg db "Resultado: ", 0
 len equ $ - msg

 err db "Error: division entre 0", 10, 0
 lenErr equ $ - err

 newline db 10, 0
 lenNL equ $ - newline

section .bss
 resultado resb 1

section .text
 global _start

_start:
 ; ============================
 ; Números hardcoded
 ; ============================
 mov al, '8' ; primer número (ASCII)
 sub al, '0' ; convertir a entero (8)

 mov bl, '0' ; segundo número (ASCII) → aquí prueba el error
 sub bl, '0' ; convertir a entero

 ; ============================  ;
Verificar división entre 0
 ; ============================
cmp bl, 0
 je division_cero ; si bl == 0 saltamos

 ; ============================  ;
División AL / BL
 ; ============================
xor ah, ah ; limpiar AH
 div bl ; resultado en AL

 ; Convertir resultado a ASCII
 add al, '0'
 mov [resultado], al
```

```
 ; ============================  ;
Imprimir "Resultado: "
 ; ============================
mov eax, 4
 mov ebx, 1
 mov ecx, msg
 mov edx, len
 int 0x80

 ; Imprimir el resultado
 mov eax, 4
 mov ebx, 1
 mov ecx, resultado
 mov edx, 1
 int 0x80

 jmp fin ; saltar para evitar imprimir error

division_cero:
 ; ============================  ;
Imprimir mensaje de error
 ; ============================
mov eax, 4
 mov ebx, 1
 mov ecx, err
 mov edx, lenErr
 int 0x80

fin:
 ; Imprimir salto de línea
 mov eax, 4
 mov ebx, 1
 mov ecx, newline
 mov edx, lenNL
 int 0x80

 ; Salir
 mov eax, 1
 xor ebx, ebx
 int 0x80
```

// Cambios: el mensaje de error y ahora se compara el valor primero antes de dividir.

Referencias

1. Ornare quam viverra orci sagittis eu volutpat. Aenean et tortor at risus. Feugiat in

ante metus dictum at tempor commodo.

2. Senectus et netus et malesuada fames ac. Dictum sit amet justo donec enim diam vulputate ut pharetra. Tristique senectus et netus et malesuada fames

Anexos

```
section .data

 msg db "Resultado: ", 0

 len equ $ - msg


 newline db 10, 0

 lenNL equ $ - newline


section .bss

 resultado resb 1


section .text

 global _start


_start:
```

```
 ; ============================  ;
Números hardcoded
 ; ============================
mov al, '8' ; primer número (ASCII)  sub
al, '0' ; convertir a entero (8)

 mov bl, '2' ; segundo número (ASCII)  sub
bl, '0' ; convertir a entero (2)
 ; ============================  ;
División AL / BL
 ; ============================
xor ah, ah ; limpiar AH para div  div bl ;
resultado en AL

 ; ============================  ;
Convertir resultado a ASCII  ;
============================
add al, '0'
 mov [resultado], al

 ; ============================  ;
Imprimir "Resultado: "
 ; ============================
mov eax, 4
```

```
 mov ebx, 1
 mov ecx, msg
 mov edx, len
 int 0x80

 ; ============================  ;
Imprimir el resultado
 ; ============================
mov eax, 4
 mov ebx, 1
 mov ecx, resultado
 mov edx, 1
 int 0x80

 ; ============================  ;
Imprimir salto de línea
 ; ============================
mov eax, 4
 mov ebx, 1
 mov ecx, newline
 mov edx, lenNL
 int 0x80

 ; ============================  ;
```

Salir

; ============================

mov eax, 1

xor ebx, ebx

int 0x80