UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA

Facultad de Ingeniería, Arquitectura y Diseño.

Ingeniería en Software y Tecnologías Emergentes

Organización de Computadoras

Taller 11

Alejandro Palomares Ceseña

1. Qué es una interrupción en ensamblador

Una **interrupción** es un mecanismo que detiene temporalmente la ejecución normal del procesador para atender un evento especial.
Cuando ocurre una interrupción, la CPU:

1. Guarda su estado actual.
2. Salta a una rutina especial llamada **ISR** (Interrupt Service Routine).
3. Atiende el evento.
4. Regresa a la ejecución normal.

2. Qué son las interrupciones de hardware

Son interrupciones generadas **físicamente por dispositivos externos**, por ejemplo:

- Teclado
- Mouse
- Temporizador
- Controlador de discos

El hardware envía una señal al procesador para pedir atención inmediata.
Pasan por el **PIC (Programmable Interrupt Controller)** en sistemas x86.

Ejemplo típico:
Cuando presionas una tecla, el teclado genera una interrupción y la CPU ejecuta la rutina encargada de leer la tecla.

3. Qué son las interrupciones de software

Son interrupciones provocadas por instrucciones dentro del programa, como:

int 0x80

El programador las usa para:

- Llamar al sistema operativo
- Hacer servicios del BIOS
- Manejar excepciones

Son un “puente” entre el programa y el sistema operativo.

4. Diferencia entre interrupción enmascarable y no enmascarable

Tipo	Descripción
Enmascarable (IRQ)	Se puede desactivar temporalmente. El sistema decide si ignorarlas o atenderlas.
No enmascarable (NMI)	No se pueden desactivar. Se usan para fallas graves.

NMI ejemplo:
Error de memoria o daño eléctrico → el CPU detiene todo y atiende la interrupción.

5. Tres interrupciones de hardware (x86)

1. IRQ0 – Temporizador del sistema

- Frecuencia: ~18.2 Hz
- Mantiene la hora del sistema y la multitarea.

2. IRQ1 – Teclado

- Se activa cuando se presiona o suelta una tecla.
- Procesa códigos de teclado (scancodes).

3. IRQ14 – Controlador de disco primario

- El disco duro indica que ha terminado de leer o escribir datos.

6. Tres interrupciones de software (x86)

1. int 0x80 – Llamadas al sistema Linux

Permite usar funciones como:

- escribir (sys_write)
- leer (sys_read)
- salir (sys_exit)

Ejemplo:

```
mov eax, 1
int 0x80
```

2. int 0x10 – Servicios de video del BIOS

Permite controlar:

- modo de video
- impresión de caracteres en pantalla
- color de texto

3. int 0x21 – Servicios de DOS

Se usa en MS-DOS:

- manejo de archivos
- teclado
- pantalla

Ejemplo: imprimir un carácter.

7. Programa modificado: Manejo de división entre 0 usando *solo saltos condicionales*

HelloWorld.asm 445sd7v2u

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

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

    newline db 10, 0
    lenNL equ $ - newline

section .bss
    resultado resb 1

section .text
    global _start

_start:
    ; ============================
    ; Números hardcoded
    ; ============================
    mov al, '8'       ; primer numero ASCII
    sub al, '0'       ; a entero

    mov bl, '0'       ; segundo numero ASCII (PRUEBA: divide entre 0)
    sub bl, '0'       ; a entero

    ; ============================
    ; Verificar division entre 0
    ; ============================
    cmp bl, 0
    je division_cero      ; si es 0 -> saltar a manejo

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

    add al, '0'
    mov [resultado], al
    jmp imprimir_resultado


division_cero:
    ; Resultado simulado para división entre 0
    mov eax, 4
    mov ebx, 1
    mov ecx, msgError
    mov edx, lenErr
    int 0x80
```

```
    jmp salir


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

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

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

salir:
    mov eax, 1
    mov ebx, 0
    int 0x80

    mov eax, 1
    xor ebx, ebx
    int 0x80

```