

Práctica 6 - Interrupciones

Tipos de
interrupciones

→ Excepciones: que van a ser generadas por el procesador cuando se cumpla una condición.

→ Interrupciones → Internas: su origen se da en una llamada a la instrucción INT por parte de un proceso

→ Externas: su origen se da en un dispositivo externo, reloj y teclado.

Tipos de
excepciones

1. Fault: Excepción que podría corregirse para que el programa continúa su ejecución. El procesador guarda en la pila la dirección de la instrucción que produjo la falla. Algunos faults suman un código de error a la pila

2. Traps:

Se produce al terminar la ejecución de una instrucción trap. CPU guarda en pila la siguiente instrucción a la trap.

3. Aborts: Excepción que no siempre puede determinar que le cause ni recupera la ejecución de la tarea. Reporta errores de hardware o inconsistencias en tablas del sistema

Implementación interrupciones

- 1- Escribimos la rutina de atención de cada excepción/interrupción
 - * ↳ Usar IRET en lugar de RET
- 2- Definimos IDT con los descriptores correspondientes.
- 3- Cargamos descriptor de IDT en IDTR.

* Antes de atender una excepción/interrupción el procesador pushes en la pila

Por eso es importante usar IRET, haz que sacas mas cosas de la pila

- 1- Registro EFLAGS
- 2- Registro CS
- 3- Registro EIP
- 4- En algunas excepciones tambien error code

• Si el error code es opcional, cómo sabe el procesador si tiene que quitarlo de la pila o no?

↳ No lo sabe → cuando escribimos el código de la excepción antes de hacer el IRET haz que sacarlo de la pila.

↳ Pila alineada a 4 bytes

↳ Ese es el estado de la pila cuando no hay cambio de privilegio

excepciones e

interrupciones

anteriores

- Cuando hay cambio de privilegio (en nuestro caso interrupciones internas) habrá otra pila que tendrá el siguiente estado:

- (➤ 1- SS (stack segment reg)
2- ESP
3- EFLAGS
4- CS
5- EIP
6- Error code (a veces)

- La IDT es una tabla muy parecida a la GDT en donde cada entrada es un descriptor de comprobada (ojo descriptor)

↳ IDT encogida en IDTR

↳ 0-15 : IDT LIMIT

16-47: IDT Base Address

→ La IDT contiene
descriptores para

1- Comprobadas
de terza

2- Comprobadas

de interrupción

Similares a las de interrupción
pero manejan distinto
el estado del bit de
interrupción en EFLAGS

Atributos importantes compuesta de interrupción

- 1 - Offset : la dirección en memoria donde comienza la rutina de atención de interrupción.
- 2 - Segment : indica que selector debe utilizarse al ejecutar el código de la rutina
- 3 - P, DPL : indican si la rutina está en memoria o no y el nivel de privilegio.
- 4 - Bits 8 : 4-7 → tipo específico de la compuesta, usamos 1110 en este taller

bit D indica si es de 32 o 16 bits

• A la hora de atender una interrupción lo que sucede es que se va a buscar en la IDT (cargada en el DTB) el descriptor correspondiente.

→ Una vez con el descriptor necesitamos buscar en la GDT nuestro descriptor de segmento para conseguir la base address

→ Base address + offset : rutina de atención de la interrupción

Interrupciones externas

- Las interrupciones externas son administradas por un controlador de interrupciones (PIC)
 - ↳ Las interrupciones por hardware del procesador se pisan con los índices establecidos por el PIC.
- Se deben remapear los controladores a un espacio designado a dispositivos de entrada/salida.
- Para acceder a los PIC se usan las instrucciones "IN" y "OUT"

PIC 1 20 h y 21 h
PIC 2 A0h y A1h

↓
Estas instrucciones

utilizan puertos → se identifican con número de puerto y tamaño (8/16,32 bits)

El procesador lee y escribe de un puerto usando IN y OUT

Para configurar los PIC hay que enviar palabras en un orden particular

Activar int externas

call pic - reset ; comprobar PIC

call pic - enable ; habilitar PIC

sti ; habilitar interrupciones

→ setear el flag de interrupciones

Interrupción de Reloj

→ Vamos a contar con un componente externo al procesador que va a generar una interrupción a intervalos regulares

→ PIT

programmable
interval
timer

→ necesario a la hora de implementar el manejo de tareas (scheduling)

Keyboard Controller

• Leemos teclas a través del puerto 0x60 (in al, 0x60)

• Obtenemos un scan code

→ El teclado genera

2 interrupciones

scancode = 0xxxxxx make: tecla presionada

scancode = 1xxxxxx break: se suelta la tecla

ej: presionar a → scancode 0x1E soltar a 0x4E (0x1E + 0x80)
presionar b → scancode 0x30

NOTA

Atender interrupciones externas

- ISR XX :

pushad

...
call pic-finish] notificemos que atendimos
la interrupción

...
popad

reti

Syscalls

- Es una forma que tiene el sistema operativo de exponer funcionalidad a las aplicaciones

↳ Ej: ① Manejo de procesos (fork)
② Sistemas de archivo
③ Dispositivos E/S

- Se pueden implementar de varias formas

↳ En este caso los implementamos con interrupciones de software, o sea asociando una o varias syscalls a una interrupción particular.

↳ Una app puede acceder a la syscall 76 con

int 0x4C

se atienden con rutinas de atención igual que antes
global _isr_XX

_isr_XX:

pushad

popad
iret

no lleva a pic-finish ya
que es interrupción interna

Resumen

• Pasos para utilizar interrupciones

- ① Definir una IDT con sus descriptores asociados para excepciones, int externas (reloj y teclado) y syscalls
- ② Escribir rutinas de atención: IRET, pic-finish, in OUT
- ③ Cargar IDTR y habilitar interrupciones consideraciones