Programando interrupciones

Laboratorio de Sistemas Digitales con Microprocesadores

Para que tenga éxito el proceso de la interrupción

- Para continuar con éxito el programa que se estaba ejecutando después de una interrupción, la siguiente información debe ser guardada en el stack:
  - El apuntador a instrucción del procesador (IP), el cuál apunta a la siguiente instrucción que va a ser ejecutada.
    - El IP hace referencia a un desplazamiento relativo al valor del registro segmento de código (CS), y ambos valores (CS:IP) son almacenados en el stack, ocupando 4 bytes.
  - El registro de banderas del procesador, el cual debe ser guardado para que una vez que el código de la interrupción haya hecho cambios en las banderas, el registro de banderas sea restaurado para continuar siendo utilizado por el código original del programa.
    - Este registro ocupa dos bytes en el stack.
  - Cualquier otro registro del procesador cuyo contenido cambiará debido al código de la interrupción,
    - Emplea un espacio adicional de dos bytes en el stack.

¿Cómo se guarda la información?

- Tanto CS:IP como el registro de banderas se almacenan de manera automática en el stack cuando ocurre una interrupción.
- Otros registros del procesador deben ser salvados de manera explícita por la rutina de interrupción (comúnmente llamada “manejador de interrupciones” o “rutina de servicio o atención a la interrupción (ISR)).
- Los registros (en general) no son salvados automáticamente debido a que muchos manejadores de interrupciones alteran únicamente unos cuantos registros y perderían tiempo guardando la información de aquellos registros que no emplean.

Al terminar la interrupción…

- Cuando la rutina de atención a la interrupción finaliza,
  - Restaura aquellos registros previamente salvados y
  - Ejecuta una instrucción IRET (Interrupt Return, Retorno de interrupción)
    - IRET recupera el registro de banderas y el CS:IP almacenados en el stack
  - Finalmente regresa al lugar indicado en el procesador.
    - Con el CS:IP restaurados, el programa que fue interrumpido inicia de nuevo su ejecución a partir de la instrucción que le indique el IP.

Interrupciones hardware

- Las interrupciones hardware podrían ser iniciadas por eventos tan diversos como un pulso del timer chip de la computadora, una señal desde un módem, o por presionar un botón del mouse.
  - Estas interrupciones son manejadas por un chip denominado controlador programable de interrupciones (PIC, Programmable Interrupt Controller).
- La interrupción de teclado es un ejemplo de interrupción hardware.
  - La interrupción se inicia cada vez que llega un código de tecla desde el teclado.
  - La interrupción analiza el código y (en muchos casos) lo transforma en código ASCII y lo sitúa en el buffer del teclado.
  - Entonces la interrupción del teclado finaliza y cualquier software que se esté ejecutando es libre de leer en el buffer la tecla pulsada.
  - Si la interrupción del teclado no procesa el código entrante al momento en que llega, el código podría perderse cuando otra tecla sea presionada.

Interrupciones software

- En realidad, es un poco inapropiado llamarlas “interrupciones”.
- Mientras que una interrupción hardware, tal como la interrupción del teclado, puede ser activada en cualquier momento, una interrupción software que pasa códigos de carácter desde el buffer del teclado a un programa trabaja únicamente al momento que un programa pregunta por ello – esto realmente no interrumpe a nadie.

Tabla de vectores de interrupción

- Las interrupciones hardware y software están relacionadas en que el mismo mecanismo –una tabla de vectores- es utilizada para iniciarlas.
- Un vector es una dirección de cuatro bytes de un controlador de interrupciones, dando su segmento y desplazamiento en memoria.
- Los controladores de interrupciones podrían estar ubicados en cualquier lugar de la memoria convencional.
  - Algunos están ubicados en la parte alta de la memoria de ROM BIOS, otras se encuentran dentro del COMMAND.COM, y todavía otras pueden encontrarse dentro de los device drivers u otro software residente en memoria, o aún dentro de programas de aplicación.
  - Los 1024 bytes inferiores de memoria contienen vectores de interrupción, por lo que hay espacio para 256 vectores de interrupción en total.
  - Los vectores más bajos en la tabla son usados por interrupciones hardware. Estos están numerados 8 más alto que el número de interrupción actual. Entonces, el vector para la interrupción 0 está en 0000:0020, la interrupción 1 inicia en la 0000:0024, la interrupción 2 está en la 0000:0028, y así. La figura muestra la trayectoria que un programa sigue en la ejecución de una interrupción 21h.

Excepciones

- Las posiciones más bajas de la tabla de vectores de interrupción están dedicadas a una clase especial de interrupciones: excepciones.
  - Una excepción es esencialmente una interrupción iniciada por el mismo CPU cuando ocurre un error que no puede ser manejado, como cuando un programa intenta dividir un número por cero, o cuando este encuentra instrucciones de máquina que no reconoce.
  - Cada tipo de error tiene su propio vector de interrupción que puede apuntar a una rutina de recuperación.
  - Los programadores normalmente no necesitan escribir manejadores de excepciones, ya que los compiladores evitan la aparición de algunos tipos de errores y proporcionan sus propios manejadores de excepciones para otros.

Interrupciones BIOS, DOS y OTRAS

- Las interrupciones están disponibles para cualquier programa.
  - Estas incluyen las interrupciones BIOS, las cuales están localizadas en la parte alta de la memoria ROM, y por tanto están construidas dentro de la máquina.
  - También están las interrupciones DOS que se encuentran dentro del COMMAND.COM.
  - Interrupciones que monitorean el mouse o controlan la memoria expandida son proporcionadas por los device drivers que acompañan a estos dispositivos.

Uso de almacenamiento

- Muchas rutinas de interrupción emplean variables y buffers de memoria asignados para su uso exclusivo.
  - Si una rutina de servicio de interrupción software está en ejecución y por alguna razón es interrumpida–quizá por la activación de un programa residente en memoria (un TSR)- la misma interrupción podría ser llamada nuevamente.
  - Cuando esto pasa, las variables usadas por la interrupción serán llenadas con nuevos valores.
  - Cuando el control es regresado a la primera instancia de la interrupción, está intentará terminar el trabajo que estaba haciendo. Con sus variable cambiadas, podrían haber errores, y podría suspender la operación de la máquina.
  - Es posible diseñar interrupciones que escriban sus datos en la pila, o asignar buffer de memoria dinámicamente, de manera que una llamada a una interrupción no interfiera con otra.
    - Tales interrupciones son denominadas re-entrantes.
    - Sin embargo, las interrupciones de DOS no son re-entrantes.

Uso de almacenamiento

- Con sus variable cambiadas, podrían haber errores, y podría suspender la operación de la máquina. Es posible diseñar interrupciones que escriban sus datos en la pila, o asignar buffer de memoria dinámicamente, de manera que una llamada a una interrupción no interfiera con otra. Tales interrupciones son denominadas re-entrantes. Sin embargo, las interrupciones de DOS no son re-entrantes.

Interrupción		Uso
Hex	Decimal	
00h	0	Generada por la CPU cuando se intenta hacer una división por cero
01h	1	Utilizada par ir paso a paso por los programas (como DEBUG)
02h	2	Interrupción no enmascarable
03h	3	Utilizada para establecer puntos de ruptura en programas (igual que con DEBUG)
04h	4	Generada cuando operaciones aritméticas dan operaciones de desbordamientos
05h	5	Invoca la rutina de servicio de imprimir pantalla de la ROM BIOS
06h	6	Reservada para DOS
07h	7	Reservada para DOS
08h	8	Generada por el tic-tac del reloj del hardware
09h	9	Generada por acción del teclado
0Ah a 0Dh	10 a 13	Reservadas para BIOS
0Eh	14	Señala atención al diskette (por ejemplo, para señalar operación completada)
0Fh	15	Utilizada para controlar la impresora
10h	16	Invoca servicios de vídeo de la ROM BIOS
11h	17	Invoca el servicio de lista de equipamiento de ROM BIOS
12h	18	Invoca servicio de tamaño de memoria de la ROM BIOS
13h	19	Invoca servicios de disco de la ROM BIOS
14h	20	Invoca servicios de comunicaciones de la ROM BIOS
15h	21	Invoca servicios del sistema de la ROM BIOS
16h	22	Invoca los servicios estándar del teclado de la ROM BIOS
17h	23	Invoca los servicios de la impresora de la ROM BIOS
18h	24	Activa el lenguaje BASIC de la ROM
19h	25	Invoca la rutina cargadora de la secuencia de arranque de la ROM BIOS (invocarla equivale a hacer un RESET)
1Ah	26	Invoca los servicios de hora y fecha de la ROM BIOS
1Bh	27	Interrupción de la ROM BIOS para Ctrl-Break
1Ch	28	Interrupción generada con cada pulso de reloj
1Dh	29	Apunta a la tabla de parámetros de control del vídeo
1Eh	30	Apunta a la tabla de parámetros de la unidad de disco
1Fh	31	Apunta a los caracteres gráficos del CGA
20h	32	Invoca al servicio de terminación de programa del DOS
21h	33	Invoca a todos los servicios de llamada a función DOS
22h	34	Dirección de la rutina de terminación del programa del DOS
23h	35	Dirección de la rutina de break del teclado del DOS
24h	36	Dirección de la rutina de errores críticos del DOS
25h	37	Invoca al servicio de lectura absoluta del DOS
26h	38	Invoca al servicio de escritura absoluta del DOS
27h	39	Termina un programa, quedando residente
28h	40	Dos Idle
29h	41	Interno DOS. PutChar Rápido
2Ah a 2Dh	42 a 46	Reservado para DOS
2Eh	48	Ejecutar comando
2Fh	47	Interrupción múltiple del DOS
30h a	48 a 50	Reservado para DOS
32h		
33h	51	Funciones del driver del ratón
34h a 3Eh	52 a 62	Reservado para DOS
3Fh	63	Gestor Overlay
40h	64	Reasignación Disquete BIOS
41h	65	Apunta a la tabla de parámetros del disco duro
42h	66	Gestor Vídeo Reasignado
43h	67	Apunta a los caracteres gráficos de vídeo (EGA, PS/2)
44h	68	API red Novell
45h	69	Reservado
46h	70	Parámetros del disco duro
47h a 49h	71 a 73	Reservadas para BIOS
4Ah	74	Alarma usuario
4Bh a 5Fh	75	Reservadas para BIOS
60h a 66h	A 102	Reservadas a programas
67h	103	Invoca al gestor de memoria expandida LIM
68h a 69h	104 a 105	Reservadas para BIOS
70h	106	Reloj tiempo real
71h a 74h	107 a 110	Reservadas para BIOS
75h	111	Redirigido a interrupción NMI
76h a 79h	112 a 114	Reservadas
7Ah	115	Software Novell (API)
7Bh a 7Fh	116 a 120	No usadas
80h a F0h	121 a	Reservado para BASIC
F1h a FFh	A 255	Reservadas a programas

```
title  Control-Break Handler                (CTRLBK.ASM)

.model small
.stack 100h
.data
msg label byte
 db 'Ctrl-Break demonstration. (CTRLBK.EXE).',0dh,0ah
 db 'This program disables Ctrl-Break. Press any'
 db  0dh,0ah
 db 'keys on the keyboard, and press ESC to return to
DOS.'
 db  0dh,0ah,0

.code
extrn Writestring:proc

main  proc
    mov   ax,@data
    mov   ds,ax
    mov   dx,offset msg  ; display greeting message
    call  Writestring

    install_handler:
    push  ds           ; save DS
    mov   ax,@code     ; initialize DS
    mov   ds,ax
    mov   ah,25h       ; set interrupt vector
    mov   al,23h       ; for interrupt 23h
    mov   dx,offset break_handler
    int   21h
    pop   ds           ; restore DS

L1: mov   ah,1         ; wait for a key, echo it
    int   21h
    cmp   al,1Bh       ; ESC pressed?
    jnz   L1           ; no: continue
    mov   ax,4C00h     ; yes: exit program
    int   21h
main  endp
```

```
; The following routine executes when Ctrl-Break
; is pressed.

break_handler proc
    push  ax
    push  dx
    mov   ah,2       ; sound a beep
    mov   dl,7
    int   21h
    pop   dx
    pop   ax
    iret
break_handler endp

end main
```