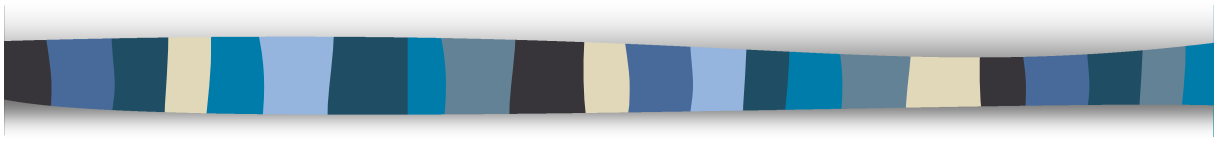
Interrupciones



Material Elaborado por el Profesor Ricardo González
A partir de Materiales de la Profesora
Angela Di Serio

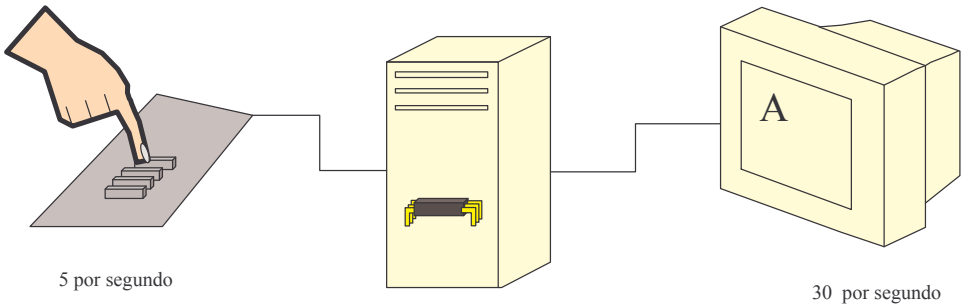
Eventos ocasionales y asíncronos

Hasta el momento hemos estado trabajando con programas que contemplan todas las situaciones que deben ocurrir, sin embargo, existen eventos que son poco comunes o que, simplemente no se puede predecir con certeza cuando van a ocurrir.

Debemos estarlos esperando todo el tiempo para ver si ocurren, esto puede ser poco recomendable, por lo que es preferible seguir haciendo otras cosas y atenderlos exclusivamente cuando ocurran, pero:

¿Cómo hacemos para atender este tipo de situaciones en los programas?

Atención de dispositivos de E/S



0,00000001 % del tiempo disponible se usa
99,99999999 % del tiempo ocioso

0,00000006 % del tiempo disponible se usa
99,99999994 % del tiempo ocioso

La opción de programación tradicional seria ir preguntando todo el tiempo si se ha pisado un tecla y de ser cierto el CPU detecta esta situación y envia a la pantalla el carácter correspondiente.
¿Por que en lugar de hacer esto, el CPU no se encarga de otras cosas, y sólo cuando alguien presione una tecla, se le avisa al CPU para que él atienda esta situación.?

Atención de dispositivos de E/S

1. El CPU le indica al dispositivo que requiere un servicio.
2. El CPU se pone a trabajar en otra tarea Tk
3. El dispositivo interrumpe al CPU cuando ha completado el servicio
4. El CPU toma la información proporcionada por el dispositivo y
5. Continua una vez obtenido el servicio deseado en el punto en el que dejo a la tarea Tk.

Atención de dispositivos de E/S

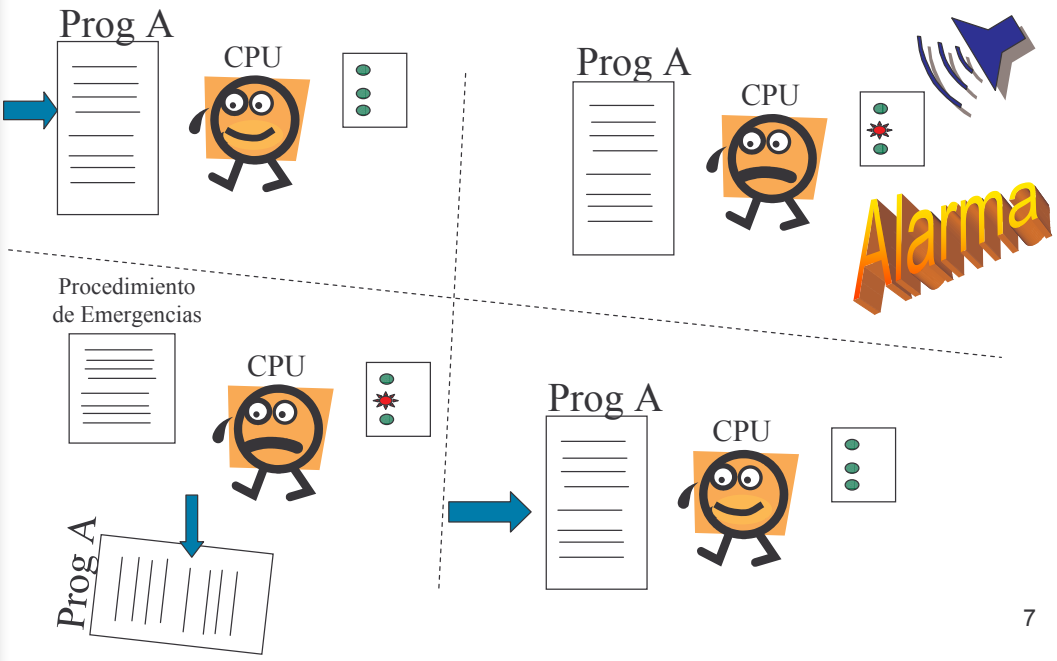
- Desde el punto de vista del programa del usuario, una interrupción es precisamente eso, una interrupción en la secuencia normal de ejecución del programa. El programa del usuario no tiene que incluir ningún código para posibilitar las interrupciones.
- Se añade el ciclo de interrupción al ciclo de instrucción.
- Con el uso de interrupciones, el procesador puede dedicarse a ejecutar otras instrucciones mientras una operación de E/S se está procesando.

Interrupciones

Una interrupción es la respuesta del sistema ante la ocurrencia de un evento excepcional o asíncrono. Esta respuesta por lo general implica las siguientes actividades:

- Automáticamente se salvaguarda el estado del CPU para permitir un reinicio posterior de la operaciones
- Se genera automáticamente una ruptura de control del programa actual, que entrega el control del computador a una rutina manejadora de interrupciones, la que a su vez se encarga de tratar el tipo particular de interrupción que ha ocurrido.

Interrupciones



Aspectos a considerar en las interrupciones

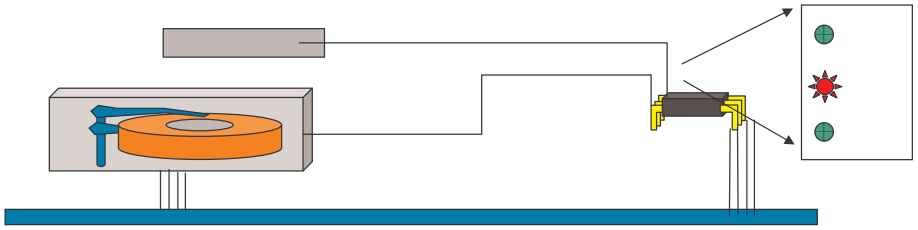
- Señal de petición de interrupción
- Procesamiento de una interrupción
- Rutina de servicio de la interrupción
- Vector de Interrupciones
- Habilitación o deshabilitación de interrupciones
- Prioridad de la interrupción
- Interrupciones enmascarables y no enmascarables

Señal de petición de la Interrupción

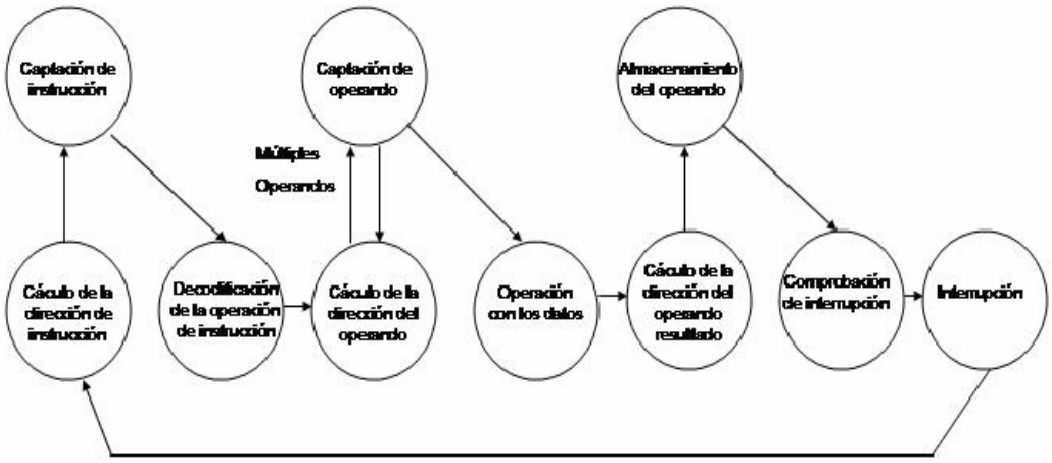
La interrupción debe procesarse sin intervención del software que se está ejecutando en el momento en el que ocurre.

Para que esto pueda ocurrir, el hardware debe proveer un mecanismo básico, a través del cual, se dé curso a la interrupción.

Para esto es necesario que el CPU tenga una línea de entrada que provenga del dispositivo y que una señal que llegue por dicha línea obligue al CPU a atenderla.

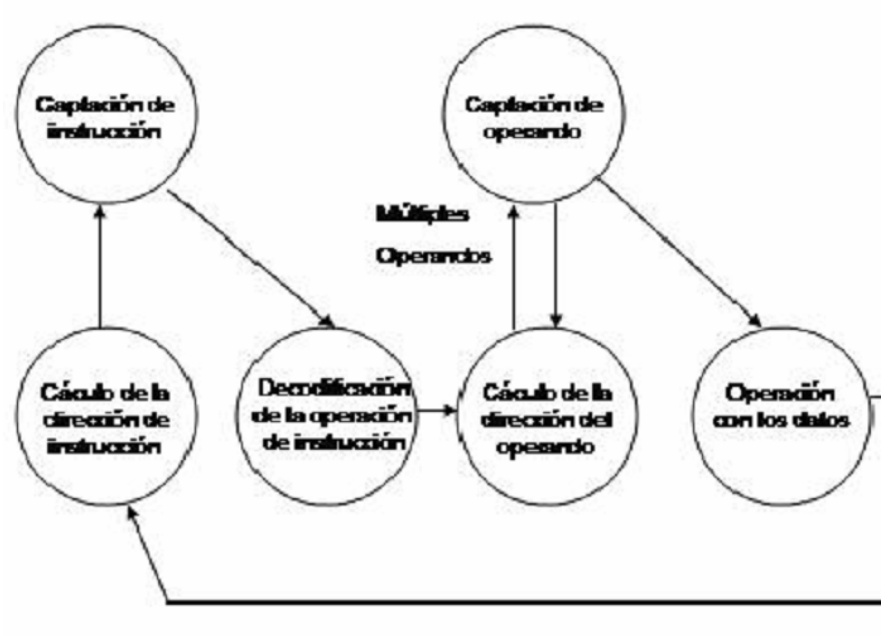


Ciclo de interrupción en el ciclo de ejecución de una instrucción

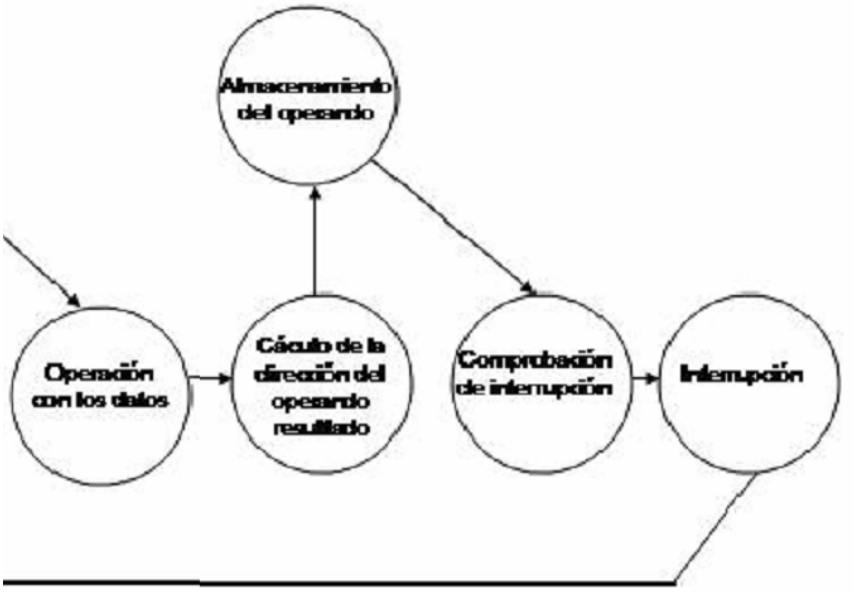


Esta técnica supone cierta penalización u *overhead* pues deben ejecutarse instrucciones extras que permiten determinar el tipo de interrupción que ocurrió y decidir la acción apropiada. Pero esto es mejor que esperar a que se lleve a cabo una operación de E/S.

Ciclo de interrupción en el ciclo de ejecucion de una instrucción



Ciclo de interrupción en el ciclo de ejecucion de una instrucción



Procesamiento de una interrupción

Cuando un dispositivo termina una operación, se produce la siguiente secuencia de eventos en el hardware:

1. El dispositivo envía una señal de interrupción al procesador
2. El procesador termina la ejecución de la instrucción en curso antes de responder a la interrupción como se muestra en la figura anterior.
3. El procesador comprueba si hay interrupciones, determina que hay una, y envía una señal de reconocimiento al dispositivo que originó la interrupción. La señal de reconocimiento hace que el dispositivo desactive su señal de interrupción
4. En este instante, el procesador necesita prepararse para transferir el control a la rutina de servicio de interrupción. Para empezar, debe guardar la información necesaria para continuar el programa en curso, en el punto en que fue interrumpido. La información mínima es el estado del procesador (PSW) y la dirección de la siguiente instrucción a ejecutar, que está contenida en el contador de programa. Estos registros se pueden introducir en la pila de control del sistema.
5. Se pasa el control a la rutina manejadora de la interrupción.

Procesamiento de una interrupción

5. Se pasa el control a la rutina manejadora de la interrupción

En este punto el procesador carga el contador del programa con la posición de inicio de la rutina de servicio de interrupciones.

- Según sea la arquitectura del computador y el diseño del sistema de operación, puede haber un solo programa manejador de interrupciones, o uno por cada tipo de interrupción, o uno por cada dispositivo y cada tipo de interrupción.
- Si hay más de una rutina manejadora, el procesador debe determinar el tipo de interrupción ocurrida para llamar al programa asociado.

Rutina de Servicio de Interrupciones

La ejecución de la rutina manejadora de interrupciones da lugar a las siguientes operaciones:

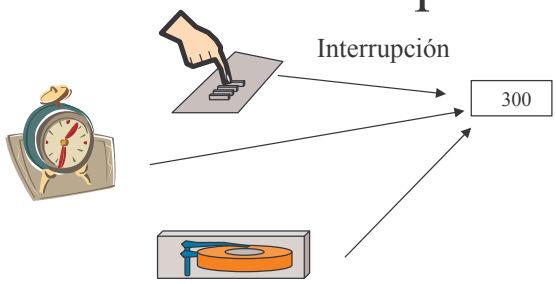
6) Hasta este momento, se han almacenado en la pila del sistema el contador de programa y la PSW del programa interrumpido. Sin embargo, hay otra información que se considera estado del programa en ejecución. Se deben guardar los contenidos de los registros del procesador, puesto que estos registros pueden ser utilizados por la rutina de manejo de la interrupción, alterando los valores que tenían en el programa interrumpido.
7) La rutina de interrupción puede continuar ahora procesando la interrupción.
8) Cuando el procesamiento de la interrupción finaliza, los valores de los registros almacenados se recuperan de la pila y se vuelven a almacenar en los registros.
9) El paso final es recuperar los valores de la PSW y del contador del programa desde la pila. Como resultado de esto, la siguiente instrucción que se ejecute pertenecerá al programa previamente interrumpido.

Rutina de Servicio de Interrupciones

Dado que el manejador de interrupciones puede ser ejecutado desde cualquier punto, no puede haber una preparación explícita, como si ocurre con las llamadas a las subrutinas.

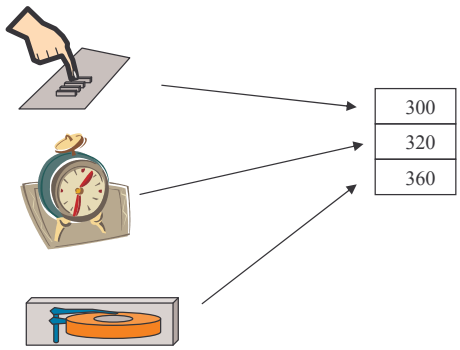
El manejador de interrupciones debe ser escrito cuidadosamente para asegurar que los registros usados por el programa interrumpido no se vean afectados. Por lo tanto, el manejador debe hacer todo el trabajo de salvado de registros, pues puede ser invocado en cualquier punto y antes de finalizar debe recuperar los contenidos de los registros usados.

Vector de Interrupciones



Manejador de Interrupciones

300	Si teclado ent x
304	Si reloj ent y
308	Si I/O ent z
30A	



Manejador de Interrupciones

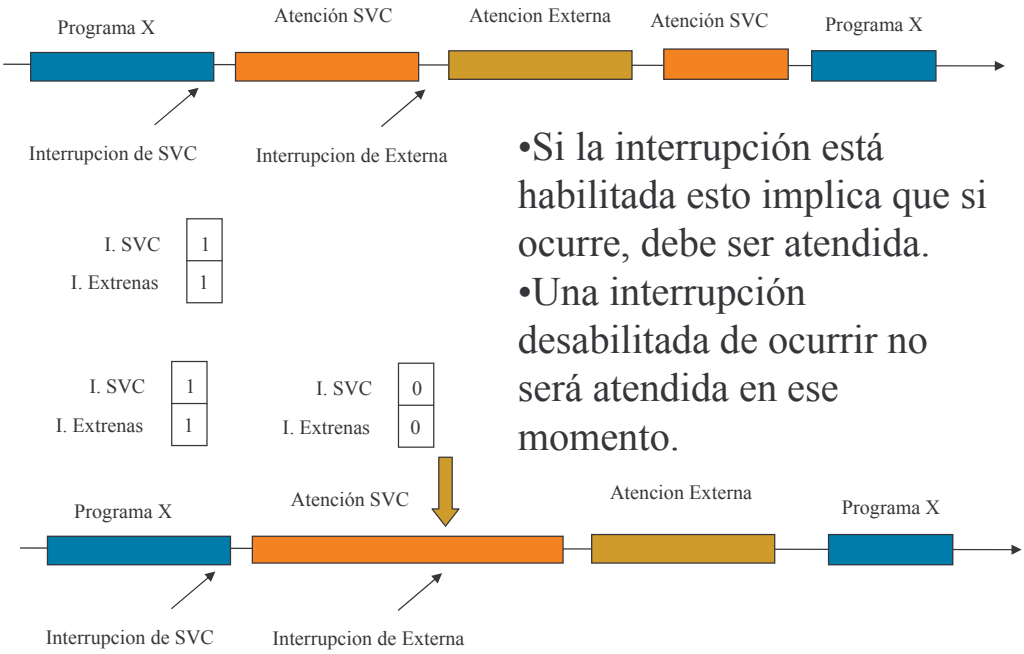
300	Int. de teclado x
320	Int. de reloj ent y
360	Int. de I/O ent z

Vector de Interrupciones

Uno de los pasos descritos anteriormente implica la búsqueda de la dirección de la rutina de servicio de la interrupción. Si bien las direcciones de las distintas rutinas podrían ser fijas y estar “cableadas” en hardware, existe una solución sencilla que permite mayor flexibilidad para definir las direcciones en la que se encuentran dichas rutinas. En esta solución lo que se fija en el hardware son las direcciones en las que el usuario podrá escribir las verdaderas direcciones de las rutinas.

Por ejemplo en una arquitectura sencilla las localidades de memoria 100, 104 y 108 podrían estar reservadas para que contengan las direcciones de las rutinas de servicio de interrupción del teclado, del monitor y la impresora. Cuando ocurre una interrupción del monitor, el hardware buscará en la dirección 104, la dirección a la que debe saltar.

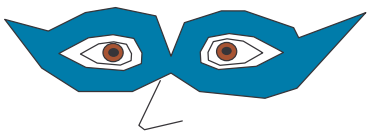
Habilitación o deshabilitación de Interrupciones



- Si la interrupción está habilitada esto implica que si ocurre, debe ser atendida.
- Una interrupción desabilitada de ocurrir no será atendida en ese momento.

Mascara de Interrupciones

0 - Deshabilitad o Enmascaradas
1 - Habilitadas o Des enmascaradas



0	1	1	0
Programa	Externas	E/S	SVC

Overflow, Div. por Cero, Instruc. Inválida, Timer, DD1, DD2, Syscall

Overflow < Instruc. Inválida < Timer < DD1

Habilitación o deshabilitación de Interrupciones

- Un problema que puede ocurrir con la atención de las interrupciones es un anidamiento de interrupciones.
- La manera más fácil de enfrentar este problema es disponer de instrucciones que deshabiliten las interrupciones y antes de retornar de la rutina de servicio, se deben habilitar las interrupciones nuevamente.
- A veces es necesario deshabilitar interrupciones cuando una rutina de servicio está actualizando estructuras de datos críticas que pueden ser accedidas desde otras rutinas de procesamiento de interrupciones.
- Otra forma para resolver el problema de la habilitación/deshabilitación de interrupciones es mediante el empleo de prioridades

Prioridad de Interrupción

- Un mecanismo más general para el control de las interrupciones anidadas consiste en asignar niveles de importancia a las distintas interrupciones y evitar que ocurra una interrupción de menor importancia durante la ejecución de una rutina de servicio de interrupción.
- Además de evitar las interrupciones anidadas, este mecanismo permite asignar prioridades a los dispositivos de acuerdo al grado de urgencia que requiera su atención.

Interrupciones no enmascarables

Algunas interrupciones podrían estar definidas como de alta prioridad, de manera que no sea posible deshabilitarlas ni evitarlas por medio del mecanismo de prioridades. Un ejemplo sería la interrupción de una fuente de poder para indicar que sólo quedan unos minutos de energía eléctrica. Este tipo de interrupción debe proceder a como de lugar, salvando datos para evitar que quede información inconsistente en el disco.

Tipos de Excepciones o Interrupciones

El mecanismo de interrupciones es de uso más general. Hasta ahora lo hemos visto aplicado a E/S. Pero hay una variedad de razones por las cuales es deseable interrumpir la ejecución de un programa.

A parte de los dispositivos de E/S demandando un servicio, el Sistema de Operación puede interrumpir la ejecución de un proceso. Para esto, el HW provee generalmente un *timer* o temporizador que se activará cada cierto tiempo e interrumpe el programa retornando el control al Sistema de Operación. Otro uso de los mecanismos de interrupción consiste en ocuparse de condiciones extraordinarias que pueden ocurrir dentro de la ejecución normal de un proceso.

Por ejemplo cuando una instrucción aritmética produce un *overflow* o desbordamiento, es importante que esto sea reconocido para que el programa pueda hacer algo o que finalice con un mensaje apropiado de error. Es ineficiente que el programa explícitamente verifique si ocurrió un desbordamiento cada vez que ejecuta una instrucción aritmética.

Tipos de Excepciones o Interrupciones

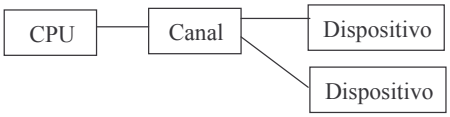
Esta situación puede ser manejada generando una excepción. En este caso se conoce como TRAP porque es el resultado directo de la ejecución de un programa y no depende de un evento externo. Es un evento síncrono que ocurre cada vez que se ejecute el programa usando los mismos datos, en estos casos el evento ocurrirá en el mismo sitio.

Otra razón para invocar el manejador de interrupciones o excepciones se presenta si el programa intenta hacer algo no permitido o no definido. Por ejemplo, acceder direcciones de memoria fuera del rango permitido, códigos de operación que no existen, etc.

Tipos de Excepciones o Interrupciones

Clasificación de interrupciones para los sistemas IBM 360-370-390

- Interrupciones de E/S (I/O)
  - Invalid I/O command
  - I/O channel end
  - I/O device end
- Interrupciones de Programa
  - Invalid CPU Instruction
  - Fixed point aritmetic Overflow
  - Storage protection violation
- Interrupciones de llamada al Supervisor (SVC)
- Interrupciones Externas
  - Internal Timer going off
  - Operator Interrupt button
  - CPU-to-CPU communication interrupt
- Interrupción por chequeo de la máquina (posible hardware failure was detected)



¿Cómo se comunica un dispositivo de E/S ?

Un módulo de E/S es el elemento del computador responsable del control de uno o más dispositivos externos y del intercambio de datos entre esos dispositivos y la memoria principal o los registros del CPU. Así, el módulo de E/S debe tener una interfaz interna al computador con el CPU y la memoria principal y una interfaz externa al computador con el dispositivo.

La complejidad de los módulos de E/S y el número de dispositivos externos que controlan varían considerablemente. En la figura 2 se muestra un diagrama de bloques de un módulo de E/S. El módulo se conecta al computador a través de un conjunto de líneas (buses). Los datos, que se transfieren a y desde el módulo, se almacenan temporalmente en uno o más registros de datos. Además puede haber uno o más registros de estado que proporcionan información del estado presente.

Un registro de estado puede realizar labores de gestión del registro para recibir información de control del CPU. La lógica que hay en el módulo interactúa con el CPU, a través de una serie de líneas de control. Estas líneas son las que utiliza el CPU para proporcionar las órdenes al módulo de E/S

¿Cómo se comunica un dispositivo de E/S ?

Módulo de Entrada/Salida

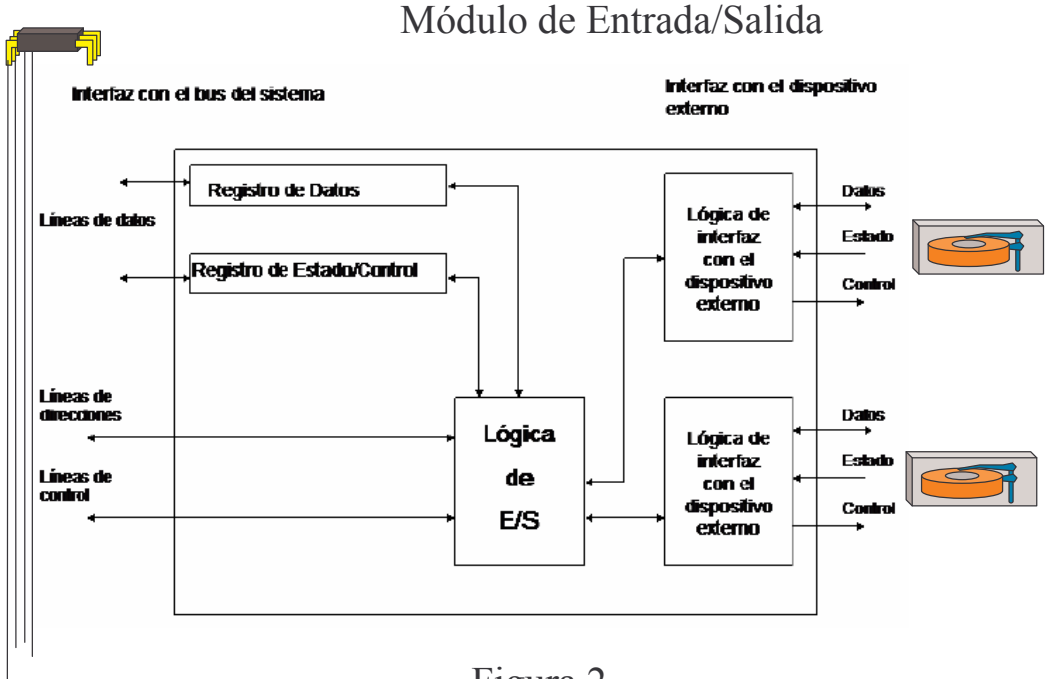


Figura 2

Cómo se comunica un dispositivo de E/S ?

Para indicarle un comando a un dispositivo de E/S, el procesador debe ser capaz de direccionar el dispositivo y suplirle una o más instrucciones de comando. Cuando el CPU, la memoria y los dispositivos de E/S comparten un bus común, son posibles dos modos de direccionamiento :

- E/S asignada a memoria (Memory Mapped I/O)
- E/S aislada

Con la E/S asignadas a memoria, existe un único espacio de direcciones para las posiciones de memoria y los dispositivos de E/S. El CPU considera a los registros de estado y de datos de los módulos de E/S como posiciones de memoria y utiliza las mismas instrucciones para acceder tanto a memoria como a los dispositivos de E/S.

Manejo de dispositivos

El registro de Estado/Control asociado a un dispositivo generalmente contiene información que nos indica el estado de dicho dispositivo. En el registro de Estado/Control podemos encontrar los siguientes campos:

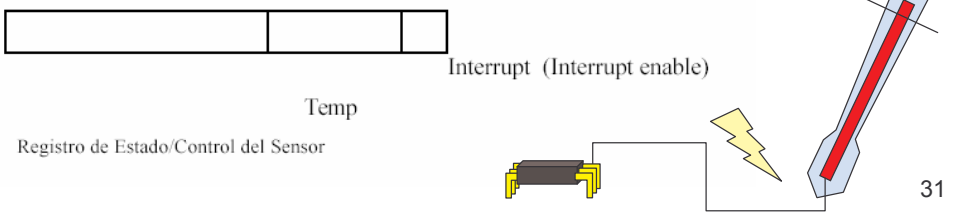
- READY: un bit que nos indica si el dispositivo está listo para recibir algún comando o que finalizó con el servicio requerido.
- Tipo de Servicio: También puede tener asociado una serie de bits que sirven para indicarle el tipo de servicio que se le está solicitando (Read, Write, etc).
- Interrup Enable: Bit que indica si el dispositivo está habilitado para generar interrupciones para el procesador
- Parámetros: bits para incluir parámetros necesarios para el servicio solicitado.

Como mencionamos anteriormente, estos registros pueden estar en posiciones reservadas de memoria (E/S asignada a memoria) o pueden estar en registros especiales y son necesarias instrucciones especificas para acceder a dicha información.

Ejemplo 1 del uso de Interrupciones

Un sensor de temperatura que registra la temperatura e interrumpe al procesador si la temperatura detectada excede un cierto valor prefijado.

Supongamos que este dispositivo se maneja asignado a memoria, es decir que los registros de datos y de Estado/Control que tiene asociados se encuentran en alguna dirección de memoria reservada (por ejemplo 0xFFFF1000). Queremos escribir un programa simple que permita habilitar la interrupción de este sensor cuando la temperatura exceda los 80°C



Ejemplo 1 del uso de Interrupciones

```
define Sensor 0xFFFF1000                    # Dispositivo asignado a memoria
VectorInterrupt[Sensor] = SensorManejador # Instalación de la rutina manejadora que
                                          #   se hará cargo de la interrupción que ocasione el Sensor
Sensor.Temp = 80                            # Temperatura a la cual interrumpirá el sensor
Sensor.interrupt = ON                       # habilitación de interrupciones
.....
# Aquí va cualquier código que deba ejecutar el procesador
SensorManejador()
    {     se incluye cualquier acción que se quiera efectuar cuando el sensor detecte que la
          temperatura excedió los 80°C prefijados
          Si no queremos tener interrupciones anidadas mientras se ejecuta la rutina,
          entonces se deben deshabilitar momentáneamente las interrupciones por parte de
          este dispositivo o de todos los dispositivos que maneja el sistema.
    Sensor.interrupt = OFF
    ..... Acciones de la rutina
    Sensor.interrupt = ON
    }
```

Ejemplo 1 del uso de Interrupciones

Noten que la rutina SensorManejador no es invocada en ningún momento.

Cuando ocurra la interrupción ocasionada por el dispositivo Sensor, el procesador se dará cuenta de que existe una interrupción de algún dispositivo y tendrá que ver cuál es el dispositivo que está interrumpiendo.

Una vez conocido cuál de los dispositivos interrumpió, se ejecuta la rutina manejadora asociada con dicha interrupción (SensorManejador).

Ya finalizada la rutina manejadora de interrupciones, se retorna a la siguiente instrucción del programa en donde ocurrió la interrupción.

Ejemplo 2 del uso de Interrupciones

Supongamos que disponemos de un temporizador (TIMER) que interrumpe cada segundo. Se desea que implemente un reloj que lleve la hora del día.

VectorInterrup[TIMER] = TimerManejador

Reloj = 0 horas, 0 minutos, 0 segundos

.....

```
TimerManejador()
{
    segundos = segundos + 1
    if (segundos = = 60 )
    {       segundos = 0
            minutos = minutos + 1
            if ( minutos = = 60 )
            {           minutos = 0
                        horas = horas + 1
                        if ( horas = = 24 )
                        horas = 0
            }
    }
}
```

Ejemplo 3 del uso de Interrupciones

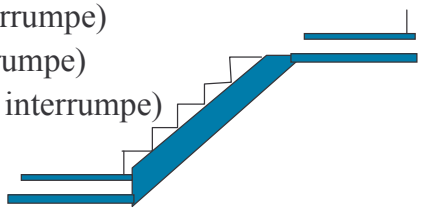
Problema de la Escalera Mecánica.

Supongamos que una escalera mecánica posee una plancha en la entrada de la escalera que, al ser pisada por un usuario, activa el motor que pone a funcionar la escalera.

Además dispone de otra plancha en la salida de la escalera. Cuando esta plancha es pisada, se detendrá el funcionamiento de la escalera mecánica.

Dispositivos presentes en el problema:

- Plancha de Entrada (dispositivo que interrumpe)
- Plancha de Salida (dispositivo que interrumpe)
- Motor de la escalera (dispositivo que no interrumpe)



Registro de Estado/Control de las Planchas

Enable: un bit que indica si la plancha puede interrumpir al procesador

Registro de Estado/Control del motor

Encendido : un bit que indica si el motor está encendido o apagado

Ejemplo 3 del uso de Interrupciones

Supongamos que los distintos dispositivos que intervienen en este problema están asignados a diversas direcciones de memoria

Plancha Entrada 0x FFFF1000

Plancha Salida 0x FFFF1004

Motor 0x FFFF1008

VectorInterrupt[Plancha Entrada] = PlanchaEntradaManejador

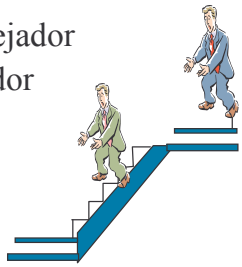
VectorInterrupt[Plancha Salida] = PlanchaSalidaManejador

Motor.Encendido = OFF

PlanchaEntrada.Enable=TRUE

PlanchaSalida.Enable = TRUE

integer contador = 0



```
PlanchaEntradaManejador( )
        { contador ++
        Motor.Encendido = ON
        }
```

```
PlanchaSalidaManejador( )
        { contador - -
        if (contador <= 0)
                { Motor.Encendido = OFF
                contador = 0
                }
        }
```

Ejemplo 3 del uso de Interrupciones

Esta versión no es del todo correcta pues puede ocurrir lo que se conoce como Condición de Carrera debido a que estamos usando una variable que es compartida por ambos manejadores y si ocurren interrupciones anidadas, su valor final puede no ser correcto

¿Qué pasa si entra una madre con su hijo y ambos pisan la plancha de entrada, pero el niño sale de la escalera en los brazos de su madre ? El motor no se apaga.

Una posible solución es que tengamos sólo una plancha de entrada y un temporizador que va a permitir que el motor se apague después de un tiempo razonable para que el usuario de la escalera mecánica llegue a la plancha de salida.

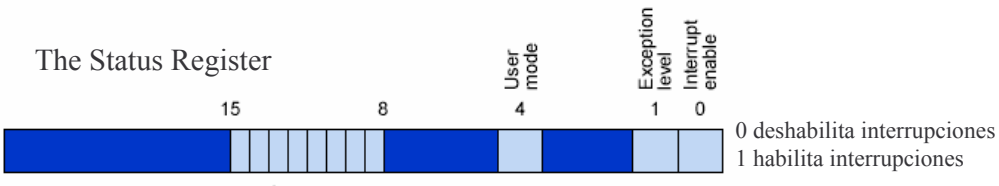
Interrupciones en SPIM

mfc0
mtc0

Registros del Coprocesador 0 para el manejo de las excepciones e interrupciones

Register name	Register number	Usage
BadVAddr	8	memory address at which an offending memory reference occurred
Count	9	timer
Compare	11	value compared against timer that causes interrupt when they match
Status	12	interrupt mask and enable bits
Cause	13	exception type and pending interrupt bits
EPC	14	address of instruction that caused exception
Config	16	configuration of machine

The Status Register



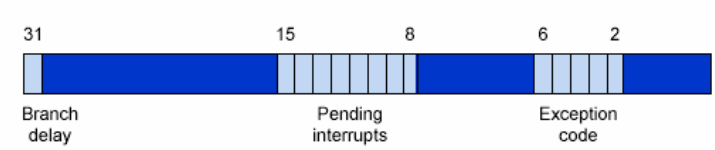
1 habilita la interrupción
0 la deshabilita

1 user
0 kernel

0
1cuando ocurre una excepción

Interrupciones SPIM

eret



The Cause Register

Exception Code

Number	Name	Cause of exception
0	Int	interrupt (hardware)
4	AdEL	address error exception (load or instruction fetch)
5	AdES	address error exception (store)
6	IBE	bus error on instruction fetch
7	DBE	bus error on data load or store
8	Sys	syscall exception
9	Bp	breakpoint exception
10	RI	reserved instruction exception
11	CpU	coprocessor unimplemented
12	Ov	arithmetic overflow exception
13	Tr	trap
15	FPE	floating point

E/S (memory mapped I/O)

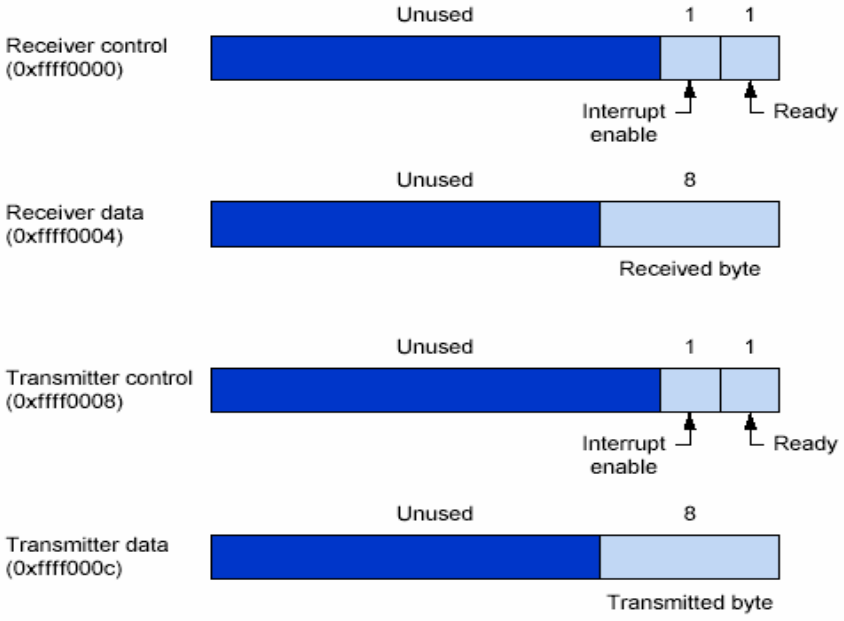


FIGURE A.8.1 The terminal is controlled by four device registers, each of which appears as a memory location at the given address. Only a few bits of these registers are actually used. The others always read as 0s and are ignored on writes.