INTRODUCCIÓN

Las interrupciones y excepciones son temas de gran importancia al estudiar los microprocesadores de las computadoras, pues se trata de fenómenos continuos y no aislados en toda computadora. El simple hecho de presionar una tecla o hacer clic, produce una interrupción. De igual forma el reloj de la computadora se mantiene constantemente generando un tipo de interrupciones que permite el buen funcionamiento de un sistema operativo.

En este documento se da a conocer el funcionamiento de las interrupciones en los microprocesadores, comenzando con los conceptos elementales de lo que son y para qué sirven las interrupciones, hasta llegar a detalles técnicos y descripciones detalladas de cada una de las interrupciones que se dan en un CPU.

También se da a conocer la diferencia entre una interrupción y una excepción, permitiendo al lector reconocer las características y particularidades de estos fenómenos, su utilidad y funcionamiento.

OBJETIVO GENERAL

“Comprender la importancia y funcionamiento de las interrupciones en las computadoras”

OBJETIVOS ESPECÍFICOS

- Conocer los diferentes tipos de interrupciones.
- Diferenciar entre una interrupción y una excepción.
- Entender el motivo por el cual se dan las interrupciones.
- Comprender la utilidad de las interrupciones.

INTERRUPCIONES

El microprocesador está en constante interacción con los periféricos del computador. Estos últimos requieren que en determinados momentos se ejecute otro programa, comúnmente conocido como rutina de servicio de interrupción, para procesar los datos que ellos generan. Ahora bien, ¿cómo puede el microprocesador estar al tanto que en determinado momento el periférico requiere de los servicios del microprocesador? Una de las respuestas a dicha pregunta es implementando una rutina que supervise constantemente el estado del periférico. Cuando esta rutina detecta que se cumplen ciertas condiciones, se ejecuta la subrutina de servicio. Este método o procedimiento trae como desventaja emplear innecesariamente ciclos de máquina del procesador en determinar el estado del periférico, ya que la necesidad de ejecutar la subrutina de servicio es aleatoria. Una manera más refinada de solucionar este problema es implementar un sistema en el microprocesador que permita que el periférico le notifique la necesidad de sus servicios. Este es el concepto de una interrupción, de modo que una interrupción es una solicitud al microprocesador para que suspenda el programa en ejecución, y se ejecute la rutina de servicio de interrupción.

A nivel físico, una interrupción se solicita activando una señal que llega a la unidad de control del microprocesador. El agente generador o solicitante de la interrupción activa la mencionada señal cuando necesita que se le atienda, es decir, que se ejecute un programa que le atienda. Ante la solicitud de una interrupción, siempre y cuando esté habilitado ese tipo de interrupción, la unidad de control realiza un ciclo de aceptación de interrupción. Este ciclo se lleva a cabo en cuanto termina la ejecución de la instrucción máquina que se esté ejecutando y consiste en las siguientes operaciones:

1. **Terminar la ejecución en curso:** el programa o proceso actual debe ser interrumpido temporalmente.
2. **Salvar algunos registros del procesador, como son el de estado y el contador de programa, de modo que la CPU, al terminar el proceso que**

dio lugar a la interrupción, pueda seguir ejecutando el programa que fue interrumpido a partir de la última instrucción: Los registros del procesador se emplean para controlar instrucciones en ejecución, manejar direccionamiento de memoria y proporcionar capacidad aritmética. Los registros son espacios físicos dentro del microprocesador con capacidad de 4 bits hasta 64 bits dependiendo del procesador que se emplee. Uno de los registros que se salvan es el de estado, pues este registro deja constancia de algunas condiciones que se dieron en la última operación realizada y que habrán de ser tenidas en cuenta para operaciones posteriores. Por ejemplo, en el caso de hacer una resta, tiene que quedar constancia si el resultado fue cero, positivo o negativo. Otro de los importantes registros del procesador que se guardan es el registro contador del programa, que lo que hace es indicar la posición en la que está el procesador en su secuencia de instrucciones, de modo que contiene la dirección de la instrucción que es ejecutada, o la dirección de la próxima instrucción a ser ejecutada. Por esta razón se le llama puntero de instrucciones. El contador de programa es incrementado automáticamente en cada ciclo de instrucción de tal manera que las instrucciones son leídas en secuencia desde la memoria. Ciertas instrucciones, tales como las bifurcaciones y las llamadas y retornos de subrutinas, interrumpen la secuencia al colocar un nuevo valor en el contador de programa. Es importante que antes de ejecutar una interrupción, el procesador guarde ciertos registros que permitirán regresar luego de que la interrupción se realice, se continúen ejecutando los procesos que se tenían de manera eficiente y con toda normalidad.

3. **La CPU salta a la dirección donde está almacenada la rutina de interrupción y ejecuta esa rutina que tiene como objetivo atender al dispositivo que generó la interrupción.**

4. **Una vez que la rutina de la interrupción termina el procesador vuelve a tomar el control:** el procesador retoma los procesos que quedaron

pendientes y que se estaban ejecutando antes que se generara la interrupción.

MOTIVO DE LAS INTERRUPCIONES

Las interrupciones se pueden generar por diversas causas, que se pueden clasificar de la siguiente forma:

1. **Excepciones de programa:** hay determinadas causas que hacen que un programa presente un problema en su ejecución, por lo que deberá generarse una interrupción, de forma que el sistema operativo trate dicha causa. Ejemplos son el desbordamiento en las operaciones aritméticas, la división por cero, el intento de ejecutar una instrucción con código de operación incorrecto o de direccional una posición de memoria prohibida.

2. **Interrupciones de reloj:** el oscilador que gobierna las fases de ejecución de las instrucciones máquina se denomina reloj. Cuando se dice que un microprocesador es de 1.60GHz, lo que se está especificando es que el oscilador que gobierna el ritmo de su funcionamiento interno produce una onda cuadrada con una frecuencia de 1.60GHz. La señal producida por este oscilador o por cualquier otro se divide mediante un divisor de frecuencia para generar una interrupción cada cierto intervalo de tiempo. Estas interrupciones, que se están produciendo constantemente, se denominan interrupciones de reloj o ticks, dando lugar al reloj como generador de interrupciones periódicas. El objetivo de estas interrupciones es hacer que el sistema operativo entre a ejecutar operaciones de forma sistemática cada cierto intervalo de tiempo. De esta manera, el sistema operativo puede evitar que un programa monopolice el uso de la computadora y puede hacer que entren a ejecutarse programas en determinados instantes de tiempo. Estas interrupciones se producen cada varios milisegundos, por ejemplo cada 20 milisegundos.

3. **Interrupciones de E/S:** una de las funciones principales del kernel o núcleo de cualquier sistema operativo es mantener una comunicación tal con el microprocesador, que permita controlar correctamente los dispositivos de E/S. Los controladores de estos dispositivos necesitan interrumpir para indicar las operaciones que realizan.

4. **Excepciones del hardware:** La detección de un error de paridad en la memoria o un corte de corriente se avisan mediante la correspondiente interrupción. Las memorias RAM se dividen en estáticas y dinámicas. Una computadora usa tanto memoria de nueve bits (ocho bits y un bit de paridad, en 9 chips de memoria RAM dinámica) como memoria de ocho bits sin paridad. En el primer caso los ocho primeros son para datos y el noveno es para el chequeo de paridad, que se refiere al uso de bits de paridad para verificar si los datos han sido transmitidos correctamente. El bit de paridad es añadido a cada siete bits que se transmite. El bit de paridad para cada byte (siete bits de datos más un bit de paridad) se pone para que todos los bytes tengan un número impar o par de grupos de bits. Por ejemplo, si dos dispositivos se están comunicando con paridad par (la forma más común de chequeo de paridad), cuando el dispositivo transmisor envía datos, se cuenta el número de grupos de bits en cada grupo de siete bits. Si el número de bits es par, se pone el bit de paridad a 0; si el número de bits es impar, se pone el bit de paridad a 1. De esta forma, cada byte tiene un número par de grupos de bits. En el caso de recepción, el dispositivo chequea cada byte para asegurarse que tiene un número par de grupos de bits. Si el receptor encuentra un número impar de grupos de bits, sabe que se ha producido un error durante la transmisión. El transmisor y el receptor deben estar de acuerdo en chequear la paridad y el tipo de paridad a usar, par o impar. Si ambos no tienen configurada la misma paridad, la comunicación se hace imposible. Este chequeo de paridad sirve para los dispositivos de almacenamiento de memoria, por ejemplo, para cada vez que se arranca la máquina. Así, una interrupción se genera cuando se detecta un error de paridad.

5. **Instrucciones de TRAP:** Estas instrucciones permiten que un programa genere una interrupción. Estas instrucciones se emplean fundamentalmente para solicitar los servicios del sistema operativo. Precisamente la activación misma del sistema operativo solamente se realiza mediante el mecanismo de las interrupciones. Cuando es un proceso en ejecución el que desea un servicio del sistema operativo ha de utilizar una instrucción TRAP, que genera la interrupción pertinente. En los demás casos será una interrupción, interna o externa, la que reclame la atención del sistema operativo. Cuando se programa en un lenguaje de alto nivel, como C, la solicitud de un servicio del sistema operativo se hace mediante una llamada a una función (por ejemplo: n = fork(), que es para la creación de un nuevo proceso). No hay que confundir esta llamada con el servicio del sistema operativo. La función fork del lenguaje C no realiza el servicio fork, simplemente se limita a solicitar este servicio del sistema operativo. En general estas funciones que solicitan los servicios del sistema operativo se componen de:

    a) Una parte inicial que prepara los parámetros del servicio de acuerdo con la forma en que los espera el sistema operativo.

    b) La instrucción TRAP que realiza el paso al sistema operativo.

    c) Una parte final que recupera los parámetros de contestación del sistema operativo, para devolverlos al programa que le llamó.

Todo este conjunto de funciones se encuentran en una biblioteca del sistema y se incluyen en el código en el momento de su carga en memoria.

Para completar la imagen de que se está llamando a una función, el sistema operativo devuelve un valor, como una función real. Al programador le parece, por tanto, que invoca al sistema operativo como a una función. Sin embargo, esto no es así, puesto que lo que hace es invocar una función que realiza la solicitud al sistema operativo. El siguiente código muestra una hipotética implementación de la llamada al sistema *fork*.

```
Int fork(){
    int r;
    LOAD R8, FORK_SYSTEM_CALL
    TRAP
    LOAD r, R9
    return(r);    }
```

El código anterior carga en uno de los registros de la computadora (el registro R8 por ejemplo) el número que identifica la llamada al sistema (en este caso FORK_SYSTEM_CALL). En el caso de que la llamada incluyera parámetros, éstos se pasarían en otros registros o en la pila. A continuación, la función de la biblioteca ejecuta la instrucción TRAP, con lo que se transfiere el control al sistema operativo, que accede al contenido del registro R8 para identificar la llamada a ejecutar y realizar el trabajo. Cuando el control se transfiere de nuevo al proceso que invocó la llamada fork, se accede al registro R9 para obtener el valor devuelto por la llamada y éste se retorna, finalizando así la ejecución de la función de biblioteca. Por ejemplo, si el programa quiere escribir datos en un archivo, el código del programa de usuario hace un CALL a la rutina en código máquina write, con código similar al mostrado anteriormente para la llamada fork. Esta rutina prepara los parámetros de la operación de escritura y ejecuta la instrucción TRAP. El sistema operativo trata la interrupción, identifica que se trata de una solicitud de servicio y que el servicio solicitado es la escritura en un archivo. Seguidamente ejecuta la rutina que lanza la pertinente operación de E/S. Finalmente, se ejecuta el planificador y el activador para dar paso a la ejecución de otro proceso.

Cuando el periférico termina la operación, su controlador genera una solicitud de interrupción que es tratada por el sistema operativo. Como resultado de la misma, el proceso anterior pasará a estar listo para su ejecución. En su momento, se seleccionará este proceso para que se ejecute. En este momento la rutina en

código máquina write recibe los parámetros que ha devuelto el sistema operativo y ejecuta un RET para volver al programa de usuario que la llamó.

Como complemento al mecanismo de aceptación de interrupción, las computadoras incluyen una instrucción máquina para retorno de interrupción, llamada RETI. El efecto de esta instrucción es restituir los registros de estado y PC, desde el lugar en que fueron salvados al aceptarse la interrupción (por ejemplo: desde la pila).

Las computadoras incluyen varias señales de solicitud de interrupción, cada una de las cuales tienen una determinada prioridad. En caso de activarse al tiempo varias de estas señales, se tratará la de mayor prioridad, quedando las demás a la espera de ser atendidas. Además, la computadora incluye un mecanismo de inhibición selectiva que permite detener todas o determinadas señales de interrupción. Las señales inhibidas no son atendidas hasta que pasen a estar desinhibidas. La información de inhibición de las interrupciones suele incluirse en la parte del registro de estado que solamente es modificable en nivel de núcleo, por lo que su modificación queda restringida al sistema operativo.

EXCEPCIONES

Se dan cuando la CPU intenta ejecutar una instrucción incorrectamente construida, como divisiones por cero, etc. Las excepciones, al igual que las interrupciones, deben estar identificadas.

CLASES DE EXCEPCIONES

Las instrucciones de un programa pueden estar mal construidas por diversas razones:

- El código de operación puede ser incorrecto.

- Se intenta realizar alguna operación no definida, como dividir por cero.
- La instrucción puede no estar permitida en el modo de ejecución actual.
- La dirección de algún operando puede ser incorrecta o se intenta violar alguno de sus permisos de uso.

DIFERENCIA ENTRE INTERRUPCIONES Y EXCEPCIONES

Cuando la CPU intenta ejecutar una instrucción incorrectamente construida, la unidad de control lanza una excepción para permitir al sistema operativo ejecutar el tratamiento adecuado. Al contrario que en una interrupción, la instrucción en curso es abortada.

El sistema operativo ocupa una posición intermedia entre los programas de aplicación y el hardware. No se limita a utilizar el hardware a petición de las aplicaciones ya que hay situaciones en las que es el hardware el que necesita que se ejecute código del SO. En tales situaciones el hardware debe poder llamar al sistema, pudiendo deberse estas llamadas a dos condiciones:

1. Algún dispositivo de E/S necesita atención.
2. Se ha producido una situación de error al intentar ejecutar una instrucción del programa (normalmente la aplicación).

En ambos casos, la acción realizada no está ordenada por el programa de aplicación, es decir, no figura en el programa. Según los dos casos anteriores, se tienen las interrupciones y las excepciones:

1. Interrupción: señal que envía un dispositivo de E/S a la CPU para indicar que la operación de la que estaba ocupado, ya ha terminado.
2. Excepción: una situación de error detectada por la CPU mientras ejecutaba una instrucción, que requiere tratamiento por parte del sistema operativo.

Peter Norton, distinguido informático nacido en 1943, dijo lo siguiente respecto a las interrupciones: “(con interrupciones) el procesador no desperdicia su tiempo buscando trabajo – cuando hay algo qué hacer, el trabajo va en busca del procesador.”

Se puede afirmar entonces que tanto las interrupciones como las excepciones alteran el flujo del programa. La diferencia entre ambas es que las interrupciones son usadas para manejar eventos externos (puertos seriales, teclado) y las excepciones son usadas para manejar errores en las instrucciones (división por cero, opcode indefinido).

Las interrupciones son manejadas por el procesador después de que finaliza la instrucción actual. Si se encuentra una señal en su pin de interrupción, buscará la dirección del manejador de interrupciones en la tabla de interrupciones y pasará el control de la rutina. Después de retornar de el manejador de rutina de interrupciones, retornará la ejecución del programa en la instrucción posterior a la instrucción interrumpida.

TRATAMIENTO DE LAS INTERRUPCIONES

Una interrupción se trata en todo caso, después de terminar la ejecución de la instrucción en curso. El tratamiento depende de cuál sea el dispositivo de E/S que ha causado la interrupción, ante la cual debe poder identificar el dispositivo que la ha causado.

IMPORTANCIA DE LAS INTERRUPCIONES

El mecanismo de tratamiento de las interrupciones permite al sistema operativo utilizar la CPU en servicio de una aplicación, mientras otra permanece a la espera de que concluya una operación en un dispositivo de E/S.

El hardware se encarga de avisar al sistema operativo cuando el dispositivo de E/S ha terminado y el sistema operativo puede intervenir entonces, si es conveniente, para hacer que el programa que estaba esperando por el dispositivo, se continúe ejecutando.

En ciertos intervalos de tiempo puede convenir no aceptar señales de interrupción. Por ello las interrupciones pueden inhibirse por programa (aunque esto no deben poder hacerlo las mismas).

IMPORTANCIA DE LAS EXCEPCIONES

El mecanismo de tratamiento de las excepciones es esencial para impedir, junto a los modos de ejecución de la CPU y los mecanismos de protección de la memoria, que las aplicaciones realicen operaciones que no les están permitidas. En cualquier caso, el tratamiento específico de una excepción lo realiza el sistema operativo.

Como en el caso de las interrupciones, el hardware se limita a dejar el control al sistema operativo, y éste es el que trata la situación como convenga.

Es bastante frecuente que el tratamiento de una excepción no retorne al programa que se estaba ejecutando cuando se produjo la excepción, sino que el sistema operativo aborte la ejecución de ese programa. Este factor depende de la pericia del programador para controlar la excepción adecuadamente.

TIPOS DE INTERRUPCIONES

Hasta este momento se ha venido hablando de los que son las interrupciones, su función y cómo trabajan. Sin embargo es también necesario clasificar las interrupciones. Se ha dicho que una interrupción se genera cuando se requiere que la CPU deje de ejecutar el proceso en curso y ejecute una función específica de quien produce la interrupción. Cuando se ejecuta esta función específica se dice que la CPU está atendiendo la interrupción. Se puede realizar una clasificación de las interrupciones, atendiendo a la fuente que las produce:

- **Interrupción de software:** se produce cuando un usuario solicita una llamada del sistema (a través de un programa).
- **Interrupciones de hardware:** son causadas cuando un dispositivo de hardware requiere la atención de la CPU para que se ejecute su manejador.

- **Excepciones:** son interrupciones causadas por la propia CPU, cuando ocurre algo no deseado, por ejemplo una división por cero.

En cuanto a las interrupciones de hardware, son producidas por varias fuentes, por ejemplo por el teclado, pues cada vez que se presiona una tecla se genera una interrupción. Una interrupción de tipo hardware es una señal producida por un dispositivo físico del ordenador. Esta señal informa a la CPU que el dispositivo requiere de su atención. La CPU parará el proceso que está ejecutando para atender la interrupción. Cuando la interrupción termina, la CPU reanuda en donde fue interrumpida, pudiendo ejecutar el proceso parado originalmente o bien otro proceso.

TABLA DESCRIPTORA DE INTERRUPCIONES PARA PROCESADORES i386

Nº de Interrupción	Descripción
0	División por cero
1	Ejecución paso a paso
2	Interrupción no enmascarable
3	Punto de ruptura
4	Desbordamiento
5	Volcar pantalla por impresora
6	Código de operación incorrecto
7	Dispositivo no disponible
8	Falta doble
9	Desbordamiento de Segmento de coprocesador
10	TTS inválida
11	Segmento no presente
12	Excepción de pila
13	Protección general
14	Falta de página
15	Reservada
16	Error de punto flotante

17	Chequeo de alineación
18	Chequeo de máquina
19-31	Reservado por el microprocesador
32-255	Disponible para interrupciones de software y hardware

0. **Interrupción 0, división por cero:** Se da cuando el divisor en una división es cero o cuando el cociente en una división es mayor que el valor máximo que permite el destino.

1. **Interrupción 1, ejecución paso a paso**: Ocurre después de ejecutar una instrucción si la "bandera" TF (Trap Flag) vale 1. Esto permite la ejecución de un programa paso a paso, lo que es muy útil para la depuración de programas.

2. **Interrupción 2, no enmascarable:** Una interrupción no enmascarable causa que la CPU deje lo que está haciendo, cambie el puntero de instrucción para que apunte a una dirección particular y continúe ejecutando el código de esa dirección. Se diferencia de los otros tipos de interrupción en que los programadores no pueden hacer que la CPU las ignore, aunque algunos ordenadores pueden por medios externos bloquear esa señal, dando un efecto similar al resto de las interrupciones. Al no poderse desactivar son empleadas por dispositivos para los que el tiempo de respuesta es crítico, como por ejemplo el coprocesador matemático Intel 8087 en el IBM PC, el indicador de batería baja, o un error de paridad que ocurra en la memoria. En algunos ordenadores Clónicos (ordenador que se monta a partir de diferentes marcas) las interrupciones no enmascarables se usaban para manejar las diferencias entre su hardware y el original de IBM. Así, si se intentaba acceder a uno de estos dispositivos se lanzaba una interrupción no enmascarable y la BIOS ejecutaba el código para el hardware presente en la máquina. También se podían lanzar interrupciones no enmascarables por el usuario, permitiendo interrumpir el programa actual

para permitir la depuración. En este caso al lanzarse una interrupción no enmascarable se suspendía la ejecución del programa actual y el control se transfería a un depurador para que el programador pudiera inspeccionar el estado de la memoria, los registros, etc. Estas instrucciones no enmascarables eran lanzadas de diferentes maneras, como pulsando un botón, por medio de una combinación de teclas o por medio de un programa. En juegos, se producía una instrucción no enmascarable y se interrumpía el juego, de esta manera se podían conseguir vidas extras por ejemplo modificando el área de memoria donde se guardaban las vidas restantes.

3. **Interrupción 3, punto de ruptura:** es una detención intencional o lugar de pausa en un programa, que tiene propósitos de depuración paso a paso. De manera más general, un punto de ruptura es un medio para adquirir conocimiento acerca de un programa durante su ejecución. Durante la interrupción, el programador inspecciona el examen de ambiente (registros, memoria, archivos, etc.) para detectar si el programa funciona como se esperaba. En la práctica, un punto de ruptura consiste de una o más condiciones que determinan cuándo la ejecución de un programa debería ser interrumpida. La forma más común de punto de ruptura es una donde la ejecución del programa es interrumpida antes de que una instrucción especificada por el programador sea ejecutada. A esto a menudo se le llama punto de ruptura de la instrucción. Otro tipo de condiciones pueden también ser usadas, tales como lectura, escritura, o modificación de una específica localización en un área de memoria. A esto a menudo se le llama.punto de ruptura de datos o punto de reloj. Otros tipos de condiciones incluyen la ejecución de interrupciones en un tiempo particular, o al presionar una tecla, y así sucesivamente. Muchos procesadores incluyen soporte de hardware para puntos de rupture (típicamente instrucciones y datos de puntos de ruptura). Dicho hardware podría incluir limitaciones, por ejemplo no permitir puntos de rupture en instrucciones localizadas en huecos de retardo (contienen instrucciones que son ejecutadas sin tener en cuenta los efectos

de la instrucción presedente). Este tipo de limitación es impuesta por la microarquitectura del procesador, por tanto varía de un procesador a otro. Sin soporte de hardware, los depuradores tienen que implementar puntos de ruptura en el software, que, particularmente para los puntos de ruptura de datos, puede impactar la actuación de la aplicación que se depura.

4. **Interrupción 4, desbordamiento:** ocurre cuando una cantidad excede la capacidad que se tiene. Por ejemplo cuando un número es excesivamente grande (más de lo que una determinada computadora puede procesar), ocurre un desbordamiento.

5. **Interrupción 5, volcar pantalla por impresora:** el servicio de impresión de pantalla está relacionado con la impresora y es solicitado mediante la interrupción 5. Este servicio se diseñó para ser manejado mediante interrupciones, y puede ser invocado desde cualquier programa que lo necesite. Esta interrupción envía el contenido del buffer de pantalla al puerto de la impresora.

6. **Interrupción 6, código de operación incorrecto:** la parte de la instrucción de lenguaje máquina de la computadora que designa la operación que se debe realizar, es incorrecta (opcode o código de operación incorrecto).

7. **Interrupción 7, dispositivo no disponible:** si ejecutamos una instrucción cuando un dispositivo no está disponible para el trabajo que se está pidiendo, como cuando ejecutamos una instrucción de punto flotante cuando la unidad de punto flotante no está disponible.

8. **Interrupción 8, doble falta:** En la arquitectura x86, una excepción doble falta ocurre si el procesador encuentra un problema mientras trata de servir una interrupción o excepción pendiente. Un ejemplo de una situación donde una doble falta ocurriría es cuando una interrupción es activada pero el segmento en el que el manejador de interrupciones reside es inválido. Si el procesador encuentra un problema cuando llama al manejador de doble falta, un triple falta es generado y el procesador se apaga. Como el doble

falta puede solamente suceder debido a errores del kernel, rara vez son causados por aplicaciones externas al kernel en un sistema operativo moderno en modo protegido (con microprocesador que posea protección de memoria y soporte de hardware para memoria virtual así como de commutación de tareas), a menos que el programa de alguna manera ganara acceso al kernel, como lo hacen algunos virus.

9. **Interrupción 9, desbordamiento de segmento de coprocesador**: esta excepción ocurre en modo protegido bajo las siguientes condiciones:

   - Un operando de una instrucción de coprocesador envuelve alrededor un límite de direccionamiento (0FFFFH para pequeños segmentos, 0FFFFFFFH para grandes segmentos, cero para segmentos extendidos hacia abajo). Un operando podría envonver alrededor un límite de direccionamiento cuando el límite segmento está cerca de un límite de direccionamiento y el operando está cerca del segmento de direccionamiento válido más grande. Debido a la envoltura alrededor, el inicio y el fin direccionado de tal operando estará cerca del final del segmento.

   - Ambos el primer y el último bite del operando (considerando envolvimiento alrededor) están en direcciones localizadas en el segmento y en páginas presentes y accesibles.

   - Los operandos se refieren a direcciones inaccesibles. Hay dos maneras en que tal operando podría expandirse a direcciones inaccesibles:

   1. El límite de segmento no es igual al límite del direccionamiento (por ejemplo, si el límite de direccionamiento es FFFFH y el límite de segmento es FFFDH). Por tanto, el operando se expandirá a direcciones que no están dentro del segmento (por ejemplo, un operando de 8 bytes que comienza un desplazamiento válido de FFFC expandirá direcciones

FFFC-FFFF y 0000-0003; sin embargo, direcciones FFFE y FFFF no son válidas, pues éstas exceden el límite);

2. El operando comienza y termina en páginas presentes y accesibles pero bytes intermedios del operando caen ya sea en una página no presente o en una página para la que el procedimiento actual no tiene derechos de acceso.

La dirección de las instrucciones numéricas que fallan y los operandos de datos podrían ser perdidos; un FSTENV no retorna direcciones confiables. Como con el 80286/80287, la excepción de desbordamiento de segmento debería ser manejada ejecutando una instrucción FNINIT (por ejemplo, un FINIT sin un WAIT que le preceda). La dirección de retorno en la pila no necesariamente apunta a la dirección fallida ni tampoco a la siguiente instrucción. La instrucción numérica fallida no es reiniciable.

El caso 2 puede ser evitado ya sea alineando todos los segmentos en los límites de una página o no iniciándolos dentro de 108 bytes del inicio o del final de la página. (El tamaño máximo de un operando de coprocesador es de 108 bytes). En caso 1 puede ser evitado asegurándose que el espacio entre el último desplazamiento básico y el primer desplazamiento de un segmento es ya sea no menor que 108 bytes o cero (por ejemplo, el segmento está lleno en tamaño). Si ninguna limitación de diseño de sistema de software es aceptable, el manejador de excepciones debería ejecutar FNINIT y debería probablemente terminar la tarea.

10. **Interrupción 10, TTS inválida:** indica que el interruptor de tareas fue activado y encontró información inválida en el objetivo de tarea. Condiciones inválidas son por ejemplo: segmento de código no ejecutable, segmento de datos que no se puede leer, etc.

11. **Interrupción 11, segmento no presente:** indica que la bandera de segmento o descriptor de puerta está limpio. El sistema operativo usa la excepción de segmento no presente para implementar memoria virtual al

nivel de segmento. Si el manejador de excepciones carga el segmento y retorna exitosamente, el programa interrumpido reactiva la ejecución. El sistema operativo podría usar el segmento no presente en la bandera de descriptor de puerta para su propio uso.

12. **Interrupción 12, excepción de pila:** esta excepción será generada cuando la instrucción ENTER es ejecutada y si no hay suficiente espacio de pila o mientras se trata de cargar segmentos de registro de pila y si se detecta que no hay segmento de pila presenta. Es posible recuperarse de este error extendiendo la pila, cuando el límite es menos o cargando el segmento que hace falta en memoria.

13. **Interrupción 13, protección general:** indica que el procesador detectó una violación de protección como escritura a un segmento de código o segmento de datos de sólo escritura y muchas otras razones que existen para esta excepción.

14. **Interrupción 14, falta de página:** esta falta ocurre principalmente durante el mecanismo de traducción de página para traducir desde una línea de dirección a una dirección física.

15. **Interrupción 15, reservada:** reservada por el microprocesador.

16. **Interrupción 16, error de punto flotante:** esta interrupción ocurre cuando ocurren errores aritméticos o cuando se detectan errores en cálculos computacionales matemáticos. Se identifican cinco tipos de excepciones de punto flotante:

    - **Inválido:** operación con operandos matemáticos inválidos –por ejemplo, 0.0/0.0, sqrt(-1.0), y log(-37.8)

    - **División por cero:** El divisor es cero y el dividendo es un número finito diferente de cero –por ejemplo, 9.9/0.0

- **Desbordamiento:** operación que produce un resultado que excede el rango del exponente –por ejemplo, MAXDOUBLE+0.0000000000001e308. Se puede dar también que se produzca un resultado que es demasiado pequeño para ser representado como un número normal –por ejemplo, MINDOUBLE * MINDOUBLE.

- **Inexactitud:** operación que produce un resultado que no puede ser representado con precisión infinita –por ejemplo, 2.0/3.0, log(1.1) y 0.1 en la entrada.

17. **Interrupción 17, chequeo de alineación:** indica que el procesador detectó un operando de memoria desalineado cuando el chequeo está habilitado. Este chequeo es llevado solamente en el segmento de pila de datos, no en el código o segmento del sistema. Un ejemplo de violación de chequeo de alineación es guardar un tipo double en una dirección que no admite divisibilidad por 4.

18. **Interrupción 18, chequeo de máquina:** es un error de hardware de la computadora que ocurre cuando el CPU de una computadora detecta un problema de hardware irreparable. En versiones de Windows anteriores a Windows XP, este error solía ser desplegado usando la “pantalla azul de la muerte” conteniendo el mensaje de error (los parámetros dentro del paréntesis podrían variar):

    STOP: 0x0000009C (0x00000004, 0x00000000, 0xb2000000, 0x00020151) "MACHINE_CHECK_EXCEPTION"

    En Linux, esta excepción se escribe en el registro del kernel y/o la pantalla de la consola (usualmente solamente en la consola cuando el error no es recuperable y la máquina se traba como resultado):

    CPU 0: Machine Check Exception: 0000000000000004

    Bank 2: f200200000000863

Kernel panic: CPU context corrupt

Este error se debe usualmente al fallo o sobrecarga de componentes de hardware donde el error no puede ser más específicamente identificado con un mensaje de error diferente. Diagnosticar el mensaje de error puede ser difícil, aunque los procesadores Intel Pentium generan códigos más específicos que pueden ser decodificados contactando al fabricante.

Esta excepción requiere que se reinicie el sistema para continuar la ejecución normal y a menudo indica un problema a largo plazo o de naturaleza general.