**Memoria caché:** es un tipo de memoria cercana al procesador, pequeña y rápida que guarda datos para que las solicitudes futuras de los mismos se puedan atender con mayor rapidez; los datos almacenados en una caché pueden ser el resultado de un cálculo anterior o el duplicado de datos almacenados en otro lugar, generalmente, da velocidad de acceso más rápido en comparación con buscar datos en la memoria principal o en un dispositivo de entrada/salida. A la hora de buscar un dato para su utilización en algún cálculo u otra operación, primero se busca en la caché antes de recurrir a la memoria principal ya que acceder a la caché resulta más cercano y ágil. En definitiva, la memoria caché es útil para tener cerca y a un rápido alcance datos que se han utilizado recientemente, o recurrentemente, y que se estima o se cree que se van a utilizar nuevamente en el futuro. Esta estimación de que ciertos datos se van a utilizar o reutilizar, está basada en dos principios de criterio muy importantes:

**· Criterio de localidad espacial:** considerar como posibles datos a acceder en el futuro a datos cercanos/contiguos a los que fueron utilizados recientemente.

**· Criterio de localidad temporal:** considerar como posibles datos a acceder en el futuro a datos recientemente utilizados/accedidos.

· Latencia: tiempo que demora una señal en llegar de un punto a otro

· Ancho de banda: cantidad de información que puede transmitirse por unidad de tiempo.

· Tiempo de respuesta: tiempo total requerido desde que un extremo de la comunicación solicita un dato/señal, hasta que lo recibe en su totalidad. Está relacionado con la latencia, sumando a los tiempos de procesamiento.

· Cache de correspondencia directa: al usar caché de correspondencia directa, a cada bloque de memoria principal se le asigna un lugar específico en la memoria caché. Es muy estricta en el ordenamiento de los bloques. Este tipo de organización es simple y fácil de gestionar, pero es la que tiene la tasa de fallos más alta. Hoy en día no se utiliza con frecuencia.

· Cache asociativa: utilizando caché asociativa, a cada bloque de memoria principal se le asigna aleatoriamente cualquier lugar de la memoria caché, lo cual hace que este tipo de organización sea flexible, pero al mismo tiempo existe una posibilidad de que ocurran colisiones. La memoria caché completamente asociativa es la que tiene la tasa de fallos más baja. No obstante, no es la organización más utilizada ya que es la más cara y compleja de gestionar debido a que examina en paralelo todas las etiquetas para dar con la dirección que corresponde a un determinado bloque de memoria principal.

· Cache asociativa por vías/conjuntos: Resulta ser una solución de compromiso entre la organización de mapeo directo y la completamente asociativa. Un bloque de la memoria principal se ubica de manera determinada en una de las vías de la memoria caché, pero dentro de esa vía puede disponerse en cualquier conjunto. Es una combinación entre la organización de correspondencia/mapeo directo y la completamente asociativa. Para ubicar un determinado bloque se busca primero de manera directa, y luego de manera asociativa. Este tipo de organización es muy utilizada, mejora la prestación de la cache directa y aprovecha la flexibilidad de la caché asociativa. Mientras más vías tiene este tipo de organización, más se va a asemejar a la de correspondencia directa, y mientras más conjuntos tenga, más se va a asimilar a la organización completamente asociativa.

1. ¿Por qué se utiliza un sistema de memoria con una jerarquía?

El objetivo de un sistema de memoria en una computadora es que tenga una gran capacidad y un tiempo de acceso reducido por el costo más bajo posible, entonces utilizar una jerarquía permite tener una solución de compromiso entre los distintos criterios, como lo son la velocidad, coste por bit, capacidad, consumo y fiabilidad, logrando que la memoria de la computadora se estructure en varios niveles de distintos tiempos de accesos y capacidades con el objetivo de conseguir unas mejores prestaciones. Cada nivel de la jerarquía se caracteriza por la distancia a la que se encuentra del procesador, los niveles más cercanos tienen una velocidad más elevada, el coste por bit es mayor y son de menor capacidad, la información fluye de un nivel inferior a uno superior según va siendo necesaria aprovechando los principios de localidad de referencias. El objetivo final de la jerarquía de memorias es conseguir que, cuando el procesador acceda a un dato, este se encuentre en el nivel más rápido de la jerarquía. Utilizando esta jerarquía de memoria y ordenando cada tipo de memoria desde las más rápidas y pequeñas (cercanas al procesador) hasta las que presentan mayor capacidad, pero son más lentas y están más distantes del procesador, podemos recordarlas: registros dentro del procesador, memoria caché, memoria principal, memoria secundaria. La idea de esta jerarquía radica en aprovechar las características ventajosas de cada tipo de memoria (alta velocidad de acceso y/o transferencia, gran capacidad) e intentar administrar inteligentemente las características que las limitan (altos costos, baja velocidad de transferencia, dificultad de acceso).

2. ¿Cuál es la ventaja de introducir una memoria caché?

La ventaja de introducir una memoria cache es que reduce el tiempo de acceso en comparación al de la memoria principal. Si un dato está en la memoria caché, es posible proporcionarlo al procesador sin acceder a la memoria principal, lo que significa ahorrar ese tiempo que se invertiría en encontrarlo y transferirlo. Si el dato deseado no se encuentra en la caché, primero se transfiere el mismo desde la memoria principal a la caché y luego se lo proporciona al procesador. En la memoria caché los datos se almacenan aprovechando los principios de localidad de temporal y espacial, los cuales se detallan a continuación:

\*Localidad temporal: alta probabilidad de volver a hacer referencia a un mismo dato en un corto espacio de tiempo, es decir, volver a acceder a un dato que ya he accedido recientemente.

\*Localidad espacial: alta probabilidad que la siguiente referencia sea a un dato almacenado en una posición cercana a la anterior, es decir, acceder a un dato que está cerca de otro que ya he accedido recientemente.

Estos principios nos permiten reducir el número de accesos que debe hacer el procesador a la memoria principal y mejorar notablemente la performance del sistema.

3. ¿A qué se refiere con metadatos almacenados en la memoria cache? ¿Por qué son necesarios? Dé ejemplos.

Los metadatos son necesarios para el manejo y organización de la caché, son datos asociados a cada línea de la caché que proporcionan información sobre el bloque de datos almacenado en cada línea. Los metadatos son el bit de validez, el bit sucio (dirty bit) y la etiqueta, también podemos mencionar los contadores.

Bit de validez: Indica si el dato que está en una línea de la cache es válido o es basura, si el bit no está activo significa que el dato está vacío o no es válido.

Dirty bit: se utiliza para indicar si un dato en la caché ha sido modificado, lo cual es útil para mantener la coherencia de los datos entre memoria caché y memoria principal, si el bit dirty está en 1, significa que el dato ha sido modificado y en consecuencia es necesario utilizar un cierto mecanismo de actualización.

Etiqueta: está compuesta por varios bits, el contenido de la etiqueta me permite realizar la correspondencia entre la línea de la memoria caché y el bloque de memoria principal que almacena.

Contadores: sirven para tener noción del uso de una línea de la cache y en consecuencia definir su reemplazo o no, hay distintas políticas de reemplazo, LRU (menos recientemente utilizada) se reemplaza la línea que más tiempo tiene sin ser referenciada, LFU (menos frecuentemente utilizada) se reemplaza la línea que haya sido menos referenciada, FIFO, etc.

5. ¿De qué depende la cantidad de bits utilizados para el índice? ¿Y para la etiqueta?

Dada una dirección de memoria principal de 32 bits, los cuales se definen para qué se utilizan; la cantidad de bits utilizados para el índice depende de la cantidad de líneas que tiene nuestra caché, por ejemplo, si tenemos una cache de correspondencia directa de 1024 líneas, necesito 10 bits de índice para direccionar cada fila de la cache 2^10 = 1024

La etiqueta depende de lo siguiente:

· De los bits que necesito para direccionar un byte dentro de una palabra que en MIPS son dos porque las palabras están alineadas en múltiplos de 4 bytes.

· Y el tamaño del bloque de la cache, siguiendo el ejemplo anterior, tengo un tamaño de bloque de 4 palabras lo cual necesito 2 bits para direccionar una palabra dentro del bloque 2^2 = 4.

Entonces, si a la dirección proporcionada por la memoria principal se le restan los bits del índice, los bits para direccionar un byte dentro de una palabra y los bits para direccionar una palabra dentro del bloque, entonces quedan 18 bits para la etiqueta que me permite identificar qué bloque de la memoria principal se encuentra en una determinada línea de la cache.

6. ¿Cuáles son los métodos de actualización utilizados en memorias cache?

Los métodos de actualización se utilizan para que no haya inconsistencia de datos, ellos son:

· Escritura directa: Simultáneamente se escribe en memoria caché y memoria principal. No se utiliza porque empeora las prestaciones accediendo muchas veces a memoria principal, lo que incrementa demasiado los tiempos.

· Post escritura: Se actualiza la memoria principal cuando se reemplaza la línea de la memoria caché. Se utiliza el dirty bit como mecanismo para indicar inconsistencia entre memoria caché y memoria principal. Luego se debe actualizar el dato para recuperar la consistencia. Este método es mejor porque reduce el número de accesos a memoria principal.

7. Cuáles son los métodos de sustitución utilizados en memorias caché?

¿Qué línea se reemplaza cuando hace falta espacio para otra?

Existen varios métodos de reemplazo, ellos son:

· Aleatoria (random): La línea a reemplazar se elige en forma aleatoria.

· Contadores:

· Menos recientemente utilizada (LRU o pseudo LRU) Siguiendo el principio de localidad se reemplaza la línea que más tiempo pasó sin ser referenciada.

· Menos frecuentemente usada (LFU): Se reemplaza la línea que haya sido menos referenciada.

· FIFO (no se usa normalmente): Se sigue un orden de reemplazo atendiendo al orden en el que se han ido utilizando las líneas, como una cola.

8. ¿De qué parámetros depende la tasa de aciertos en memorias caché?

La tasa de aciertos depende de muchos factores, y empieza a tomar relevancia cuando el procesador requiere de un dato, lo busca en la cache y en el caso de que lo encuentre almacenado allí, lo utiliza naturalmente y de esta manera ha ocurrido un acierto de caché. La cantidad de aciertos ocurridos en una gran escala de búsquedas en la caché va a depender de cómo está diseñada la misma: su organización, políticas de reemplazo y mecanismos de actualización. También depende si se hizo un buen uso de los principios de localidad de referencias que son fundamentales para mejorar la tasa de aciertos y la performance de la memoria. Adicionalmente, es importante que el programador escriba un código adecuado que aproveche lo mejor posible estos principios mencionados, para hacer un mejor y eficiente uso de la memoria.

Tasa de aciertos = Numero de aciertos / número de accesos a memoria.

9. ¿De qué parámetros depende la penalización por fallos en memorias cache?

La penalización por fallo viene determinada por el tiempo necesario para llevar el bloque que contiene el dato que provocó el fallo desde memoria principal hasta la caché. El tamaño de bloque va a inferir en la penalización, ya que al tener que llevar un bloque muy grande va a tener un tiempo de acceso costoso, entonces se incrementará la penalización por fallo y empeorarán las prestaciones de la caché. Otro aspecto importante a considerar es si se hizo un buen uso de los principios de localidad de referencias, que permiten reducir los fallos. Va a depender de cómo está diseñada la caché, su organización, políticas de reemplazo y mecanismos de actualización para que ocurran menos fallos. No debemos olvidarnos también cómo fue escrito el código a la hora de ser programado, si fue revisado y optimizado para mejorar la eficiencia en el uso de la memoria, o no.

4) Explique de qué depende la tasa de fallos y la penalización.

La tasa de fallos empieza a tomar relevancia cuando el procesador requiere de un dato, lo busca en la caché y no se encuentra almacenado allí. Entonces diremos que ha ocurrido un fallo en la caché. Seguido de eso, el procesador continúa buscando el dato en la memoria principal hasta encontrarlo, llevando el bloque de datos que contiene el que buscamos de la memoria principal hacia la caché. Se almacena en esta última y se lo proporciona al procesador. Entonces, la tasa de fallos va a depender de la penalización por fallos que viene determinada por el tiempo necesario para llevar el bloque desde memoria principal hasta la caché. El tamaño del bloque va a incidir en la penalización, si se da el caso en que es necesario llevar un bloque muy grande de la memoria principal a la caché, esto va a tener un tiempo de acceso costoso y se incrementará la penalización por fallo. Otro aspecto importante a considerar es si se hizo un buen uso de los principios de localidad de referencias, que permiten reducir los fallos. Va a depender de cómo está diseñada la caché, su organización, políticas de reemplazo y mecanismos de actualización para que ocurran menos fallos. No debemos olvidarnos también cómo fue escrito el código a la hora de ser programado, si fue revisado y optimizado para mejorar la eficiencia en el uso de la memoria, o no.

**Memoria Virtual:** es una capa de abstracción entre la dirección de memoria que ve el procesador (dirección virtual) y la dirección física real. La memoria virtual permite aislar procesos, permitiendo que cada uno de ellos lea un mapa de memoria distinto, lo que da la posibilidad de que estos distintos procesos coexistan simultáneamente sin interferirse. Esto significa mayor seguridad en el sistema.

La memoria virtual permite ahorrar complejidad de programación, requerida si la misma no está implementada. A su vez, la utilización de memoria virtual permite tener una memoria aparente mayor a la memoria principal. Así mismo, la implementación de memoria virtual cumple un papel importante al considerar la seguridad en la ejecución de procesos, ya que previene que estos interfieran entre sí.

Aun así, tenemos que tener en cuenta que, en principio un sistema sin memoria virtual es más rápido que un sistema con memoria virtual.

Acceder a la memoria secundaria es mucho más lento que acceder a la memoria principal, y hace caer muy considerablemente el rendimiento.

**Memoria Virtual Paginada:** Permite dividir la memoria en páginas, lo que nos posibilita trabajar con un mapa de direcciones de memoria exclusivo para cada proceso.

La tabla de página es parte de la lógica de traslación, requiere espacio en memoria y contiene información.

A partir de la dirección virtual, se busca la dirección física.

En la tabla de páginas hay por lo menos 2 campos: un bit de validez, y el número de marco, el cual es el número de página de la memoria física. El número de marco son los bits que se necesitan para encontrar la dirección física.

Metadatos: Son datos asociados a cada línea de la caché que proporcionan información sobre el bloque de datos almacenado en cada línea. Los metadatos son el bit de validez, el bit sucio y la etiqueta.

Inconvenientes de trabajar con tabla de páginas: Los problemas de trabajar con tabla de páginas pueden aparecer -por ejemplo- debido a una cantidad excesiva de tablas de páginas, y peor aún si estas tablas son de un tamaño importante, lo que significa un uso exagerado de la capacidad de la memoria principal ya que las tablas de páginas están almacenadas en la misma. Algunas soluciones para esto pueden ser adoptar una tabla de páginas multinivel, que implica mayor complejidad en el sistema, utilizar tablas hash, implementar el uso de registros límite y/o paginar las tablas.

Otra complejidad de la tabla de páginas es la necesidad de acceder dos veces a la memoria principal al buscar un dato. El primer acceso es para utilizar la tabla de páginas en memoria principal, y el segundo acceso es para llegar al dato buscado inicialmente.

Soluciones: Implementación de la memoria T.L.B. la cual es una memoria caché especial, asociativa, muy pequeña y muy rápida. La TLB almacena las traducciones de las direcciones de memoria virtuales recientes en sus respectivas direcciones de memoria físicas para poder encontrarlas y acceder a los datos ubicados allí. En caso de encontrar las direcciones virtuales en la TLB, se estaría ahorrando una cantidad considerable de tiempo al no tener que recurrir a la tabla de páginas ni al sistema operativo a través de una excepción, para ubicar esa dirección física buscada.

10. Cuáles son las ventajas de utilizar Memoria Virtual?

Una ventaja que proporciona la utilización de memoria virtual es el alojamiento temporal de información entre la memoria principal y el disco. Permite compartir la memoria de manera útil, asignándole a cada programa una porción de memoria virtual, aislando procesos haciendo que no interfieran entre sí y de esta manera incrementando la seguridad. Al utilizar memoria virtual, se aprecia de manera aparente un espacio de direcciones mayor al real, es decir, el espacio de direcciones que utilizan los programas durante su ejecución puede ser mayor al de las direcciones físicas de la memoria principal.

11. ¿De qué depende la cantidad de bits utilizados para desplazamiento y número de página? ¿Cómo influye en el rendimiento?

Los d bits utilizados para desplazamiento me permiten saber el tamaño de cada página (2^d) y los bits p del número de página me permiten saber la cantidad de páginas por proceso (2^p) y luego se traducen utilizando un mecanismo de traducción para saber en qué número de marco de MP está ubicada una determinada página de MV. El tener muchos bits de numero de página va a aumentar la cantidad de entradas de la tabla de páginas y aumentar su tamaño lo cual puede afectar a la memoria útil ocupando mucho espacio. Hay que encontrar un equilibrio.

12. ¿Cómo se diferencian las políticas de actualización y sustitución respecto de memoria cache?

Estas políticas son implementadas por el SO:

· Política de emplazamiento: Todas las páginas son de igual tamaño y coinciden en tamaño con los marcos de página. Una página se puede ubicar en cualquier marco libre.

· Política de reemplazo de página: Cuando no hay ningún marco de página libre se debe reemplazar algunas de las páginas de MP. Los principales algoritmos son FIFO, LFU o LRU.

· Políticas de actualización de la MS: Es necesario mantener la coherencia entre MP y MS, cuando se modifica una página en MP es necesario actualizar esa misma página en MS. Se emplea la política de post-escritura, una página modificada se actualiza en MS solo cuando se reemplaza.

13. ¿Cuáles son las problemáticas de almacenar la tabla de Paginas en memoria Principal? ¿Cómo se solucionan?

La primera problemática es que la tabla de páginas puede ser de gran tamaño, si tengo cientos de programas corriendo cada uno con su tabla de página van a ocupar mucho espacio de mi memoria útil, entonces existen varios métodos para reducir la cantidad de almacenamiento que se requiere para la tabla de páginas, por ejemplo:

· Varios niveles de tablas (tabla de páginas multinivel): este método permite que el espacio de direcciones pueda estar dividido sin tener que reservar hueco para la tabla de páginas entera. Es muy útil para espacios de direccionamiento muy grandes. La principal desventaja de este método que es requiere un proceso más complejo para la traducción de direcciones.

· Uso de registros límite: que restrinja el tamaño de la tabla de páginas para un proceso dado. Si el número de página virtual es mayor que lo que indica el registro límite, se añaden entradas a la tabla de páginas.

· Tablas hash: consiste en aplicar una función de hashing a la dirección virtual, de forma tal que el tamaño de la estructura de datos de la tabla de páginas requiere solamente ser igual al número de páginas que residen en la memoria principal.

· Paginar las tablas de páginas: Simplemente se permite que las tablas de páginas residan en el espacio de direccionamiento virtual.

La segunda problemática se trata de que se duplica el tiempo de acceso porque se requieren dos accesos a MP por cada referencia, es decir, uno para obtener la dirección física (traducción) y otro para el acceso a datos, entonces se implementa una cache "especial" llamada TLB dedicada a memoria virtual la cual es muy rápida y pequeña, almacena las traducciones más recientes aprovechando los principios de referencias lo cual me evita ir hasta la tabla de páginas y mejoro la performance de mi sistema.

14. Cuáles son los problemas de rendimiento que introduce Memoria Virtual? ¿Cómo se solucionan?

Los problemas de rendimiento que introduce memoria virtual es que se duplica el tiempo de acceso porque se requiere dos accesos a MP por cada referencia, es decir, uno para obtener la dirección física (traducción) y otro para el acceso a los datos, entonces se soluciona implementando la TLB.

También cuando ocurre un fallo de página, es decir, no se encuentra la página en MP lo cual se genera una excepción y el SO toma el control, busca la página en la memoria secundaria y lo provee a la memoria principal aplicando políticas de emplazamiento, sustitución o actualización. Esto se soluciona aplicando buenas políticas de sustitución y ajustando el tamaño de la página.

- ¿Cómo está formada la tabla de página? ¿Cuáles son sus campos? ¿De qué depende su tamaño?

La tabla de página sirve para traducir una dirección virtual a una física, está formada por el número de página física y bits de control, su tamaño depende de la cantidad de entradas que viene determinado por los p bits de numero de página de la dirección virtual (2^p) \* el tamaño de las entradas que corresponden a los bits d de desplazamiento (2^d).

- Explique qué es la TLB y cómo impacta en el rendimiento del sistema

La TLB es una caché muy pequeña y muy rápida que suele ser totalmente asociativa, es una caché especial utilizada cuando se cuenta con memoria virtual en el sistema. La TLB guarda las traducciones de direcciones virtuales más recientes aprovechando el principio de localidad, logrando mejorar el rendimiento consideradamente porque ya no se requieren dos accesos a MP por cada referencia (uno para obtener la dirección física [la traducción en sí] y otro para el acceso a los datos), lo cual incrementa el tiempo de acceso. Al tener incorporada la TLB disminuimos los tiempos de búsqueda y acceso ya que, de esta manera, evitamos recurrir a la tabla de páginas, que implica una penalización considerable.

- ¿Cuál es el papel del sistema operativo en la memoria virtual?

El sistema operativo es responsable de disponer un mapa de memoria único para cada proceso, aislándolos (mejorando la seguridad) y administrando la memoria. Por ejemplo, cuando ocurre un fallo de página, es decir, se busca una página y no está presente en la memoria principal entonces ocurre una excepción, ahí el sistema operativo toma el control y se encarga de buscar la página en la memoria secundaria y decide aplicando ciertas políticas de emplazamiento, sustitución o actualización en qué lugar de la memoria principal se almacena la página solicitada. También es responsable de reservar la memoria física y de actualizar las tablas de páginas, de forma tal que los espacios de direcciones virtuales de los distintos procesos no se pisen.

**Entrada/Salida:**

• ¿Cuál es el papel del sistema operativo en el manejo de E/S?

El S.O. gestiona la entrada y salida, actuando como interfaz y capa de abstracción entre el hardware y el usuario. El S.O. debe garantizar la seguridad y un funcionamiento controlado de las operaciones de E/S. Los usuarios hacen uso del dispositivo E/S sin tener que encargarse de todo lo que realiza el S.O., lo que implica una mayor comodidad y despreocupación para el usuario, sumado a una importante prevención de errores/equivocaciones por descuido o desconocimiento del usuario y que potencialmente puedan desencadenar consecuencias severas en el sistema. A su vez, el S.O. también gestiona las interrupciones generadas por los dispositivos E/S, junto con las que son ocasionadas por los demás programas de la computadora. El sistema operativo es el que proporciona un acceso equitativo a los recursos compartidos de E/S, priorizando maximizar la eficiencia y la productividad. El sistema operativo también puede actuar sobre dispositivos de E/S, y manejarlos.

Un aspecto importante a tener en cuenta, es que los dispositivos de E/S deben estar en condiciones de poder notificar al S.O. cuando han completado una operación o ha ocurrido un error, para que el S.O. pueda actuar en consecuencia adecuadamente. Un caso típico de esto se da, por ejemplo, cuando un disco ya ha realizado una búsqueda y consiguió (o no) los resultados.

Mecanismos de control de operaciones de Entrada/Salida:

* Entrada y salida Programada:

Es un mecanismo simple y rudimentario que no requiere hardware adicional, pero que resulta muy poco eficiente para el procesador. Se desperdician recursos. El procesador comanda toda la operación, se puede llegar a quedar esperando en un bucle millones de veces sin hacer más que eso (esto se traduce en una ineficiencia absoluta). El procesador programa (instruye) al controlador de E/S. El sistema operativo se encarga de posibilitar el trabajo que hace el procesador. Resulta una buena opción para implementar en un sistema muy sencillo donde se ejecuten pocos procesos individuales, como, por ejemplo, un lavarropas.

• ¿En qué condiciones es más conveniente utilizar E/S programada?

Se utiliza en dispositivos de E/S de bajo ancho de banda y frecuencia regular y baja, donde estamos más interesados en la reducción del coste del controlador del dispositivo y de la interfaz. La E/S programada también se utiliza en sistemas embebidos simples de hardware y software limitados con un único proceso, donde el procesador no tiene otras tareas adicionales y puede encargarse de comandar y gestionar la transferencia de datos sin que esto implique un uso ineficiente del mismo. En definitiva, cuando hablamos de E/S programada, es el procesador el que hace todo el trabajo en las transferencias y no es un problema tenerlo pendiente permanentemente del controlador de E/S, muchas veces esperándolo cuando no hay transferencias de datos activas, porque no hay otras tareas de las cuales se tenga que encargar (por eso se elige este tipo de disposición en sistemas simples, poco sofisticados y de bajo ancho de banda y transmisión de datos).

* Manejo de interrupciones:

Excepción (interrupción): suceso no planificado que interrumpe la ejecución del programa; se utiliza para detectar -por ejemplo- desbordamiento.

Interrupción: excepción que proviene del exterior del procesador. (Algunas arquitecturas utilizan el término interrupción para todas las excepciones).

Las interrupciones son excepciones particulares ocasionadas por el hardware. La implementación del manejo de interrupciones tiene sentido cuando el sistema presenta más de un proceso a ejecutar. Las interrupciones no impiden que la instrucción en curso termine de ejecutarse. Cuando aparece una interrupción, el procesador termina de ejecutar la instrucción en curso, y luego se encarga de prestarle atención a la interrupción que llegó proveniente del controlador de E/S. Al implementar manejo de interrupciones, el procesador no comprueba todo el tiempo el registro de estado del controlador de E/S, lo que mejora la eficiencia del sistema quitándole trabajo al procesador, no se desperdicia innecesariamente su uso en todo momento.

El controlador de E/S debe tener cierta inteligencia para poder trabajar con interrupciones. Es este controlador el que le avisa al procesador cuando lo necesita para una transferencia, mientras tanto el procesador puede seguir trabajando en otras tareas importantes.

• ¿En qué caso es más conveniente utilizar interrupciones?

Es más conveniente utilizar interrupciones en dispositivos de E/S de bajo ancho de banda, poca transferencia y frecuencia irregular, para gestionar dispositivos que requieren de una atención inmediata del sistema, en donde estamos más interesados en tener una controladora inteligente que en proporcionar una transferencia de alto ancho de banda. Por ejemplo, las interrupciones son útiles para vincular el mouse o teclado con la PC. También, las interrupciones son una buena opción cuando nuestro sistema presenta un procesador que debe encargarse de diversas tareas y se desea que su uso no sea malgastado ineficientemente esperando las solicitudes del controlador de E/S ante una operación de ese tipo.

• ¿Qué complejidades trae apareado implementar excepciones en las diferentes arquitecturas?

La implementación del manejo de interrupciones implica una complejidad adicional de hardware, lo que se traduce en un costo superior al tener que implementar componentes adicionales, como por ejemplo el de una línea física conductora que vincule al controlador de E/S con el procesador para que el primero pueda transmitirle al último las interrupciones a atender. Otra modificación adicional importante que es necesaria para implementar el manejo de interrupciones es la modificación de la unidad de control del procesador, agregando nuevas señales que manejan los registros EPC y de causa, para poder atender prioritariamente las interrupciones que van surgiendo. El anidamiento de interrupciones también es una complejidad adicional asociada al manejo de interrupciones.

• ¿En qué casos en necesario inhibir las interrupciones?

En los casos donde el procesador está ejecutando instrucciones importantes o críticas y es crucial que no sea interrumpido.

• Explique qué soporte de S.O. se requiere para el manejo de interrupciones.

El soporte del S.O. debe permitir que las interrupciones generadas por los dispositivos de E/S se gestionen de manera adecuada, eficiente, segura y estable. Es el S.O. el que decide cuál será la interrupción a atender, empleando políticas de prioridad para evaluar el orden en que conviene gestionarlas. A su vez, el S.O. guarda el estado del procesador antes de manejar la interrupción, pasa a modo supervisor y ejecuta una rutina relacionada a la interrupción que se realizó, en este punto se pueden ejecutar las operaciones de E/S y realizarse las transferencias de datos necesarias. Al finalizar la rutina, vuelve al estado del procesador previa a la interrupción y continua el avance del programa que se estaba ejecutando antes de manejar la interrupción. El S.O. actúa como una capa de abstracción entre el hardware y el programa de usuario que realizó la operación de E/S permitiendo que se trabaje adecuadamente y de forma segura con los eventos externos al procesador.

* DMA (Acceso directo a memoria):

Su función es transferir información entre la memoria principal y dispositivos de entrada/salida de manera directa e independiente del procesador. Para utilizar DMA, el sistema ya tiene que estar preparado para utilizar interrupciones. Conviene utilizar DMA cuando se necesita transmitir una cantidad de información considerable (mucha información) utilizando un ancho de banda importante.

El DMA se implementa incorporando un controlador especializado que transfiere datos entre un dispositivo de E/S y la memoria, independientemente del procesador.

La implementación de DMA permite que distintos componentes de una computadora accedan a la memoria del sistema para leer o escribir de manera independiente al procesador.​ Muchos sistemas de hardware utilizan DMA, incluyendo controladores de unidades de disco, tarjetas gráficas y tarjetas de sonido. DMA es una característica esencial en todos los ordenadores modernos, ya que permite a dispositivos de diferentes velocidades comunicarse sin someter a la CPU a una carga masiva de interrupciones.

A diferencia de la E/S programada o el manejo de interrupciones, el DMA se puede utilizar para interconectar o hacer la interfaz a un disco duro sin consumir todos los ciclos del procesador para las operaciones de E/S. Por supuesto, si el procesador también quiere acceder a la memoria, tendrá que esperar cuando ésta esté ocupada efectuando una transferencia de datos con el DMA.

Usando cachés, el procesador puede evitar tener que acceder a la memoria principal la mayor parte del tiempo, de modo que deja en gran medida libre el ancho de banda de la memoria, para uso de los dispositivos de E/S. Al estudiar DMA, debemos tener en cuenta que existe el problema de coherencia de caché, cuya solución requiere un sistema operativo con una funcionalidad superior, es decir más complejo. Otro aspecto importante que tenemos que tener en cuenta al estudiar DMA, es la existencia de direcciones de memoria físicas y virtuales, que originan el problema de la necesidad de traducir esas direcciones virtuales para que el DMA pueda trabajar. Esto es resuelto por el procesador, que sí puede efectuar esa traducción.

• ¿Qué complejidades introduce DMA?

El DMA requiere complejidades adicionales tanto de hardware, como de software sumadas a una codificación extensa que deben tenerse en cuenta a la hora de estudiarlo y evaluar su implementación. A continuación, se detallan las complejidades más destacadas que hay que considerar cuando se estudia DMA:

· En primer lugar, tenemos que decir que para poder utilizar DMA, se requiere de la implementación de manejo de interrupciones, para avisar al procesador cada vez que el controlador DMA termina una operación de E/S o cuando ocurre un error. Por lo tanto, de por sí ya hay que considerar en nuestro sistema la circuitería adecuada para este manejo de interrupciones, como lo son registros, buses, etc. Esto añade complejidades a nivel hardware y a nivel de software, porque el sistema operativo tiene que estar preparado para poder encargarse de coordinar, gestionar y manejar estas interrupciones.

· Al agregar el controlador DMA en sí mismo al hardware, estamos agregando un nuevo componente -el cual tiene un costo de por sí- y, además, se necesitan nuevos buses y cables para comunicar este controlador con los demás dispositivos, por lo cual la circuitería se vuelve más compleja y su costo se incrementa notablemente.

· Se genera una competencia por el bus entre el DMA y el procesador, ya que el controlador DMA puede actuar como maestro del bus y dirigir las lecturas o escrituras entre el mismo y la memoria, por lo tanto, al existir múltiples maestros, se necesita la implementación de un esquema de arbitraje el cual arbitra sobre quien será el próximo en utilizar el bus.

· Ocurren problemas con la memoria cache, ya que cuando el DMA accede a la memoria principal y realiza escrituras en la misma, este no pasa por la caché, y si en esta existen datos desactualizados que fueron modificados solamente en la memoria principal, el procesador va a trabajar con datos incorrectos por lo tanto se generan problemas de coherencia, y posteriormente resultados erróneos en los cálculos realizados por el procesador. A su vez, cuando la CPU actualiza un dato en caché y el DMA realiza una lectura de ese dato en memoria el dato va a ser obsoleto, es el caso inverso del mismo problema detallado anteriormente. Existen varias soluciones para estos problemas, como encaminar las operaciones de E/S a través de la caché, vaciar o invalidar la cache por circuitería o que el sistema operativo se encargue de hacer verificaciones.

· También ocurren problemas con la memoria virtual, porque el controlador va a tener que trabajar con direcciones virtuales y físicas. Trabajar con dirección virtuales implica que la traducción de dirección virtual a física se realiza dentro del controlador DMA. El problema de trabajar con direcciones físicas en una transferencia por DMA es que la transferencia no puede superar el tamaño de página, y si una operación de E/S lo exceder, las direcciones involucradas no estarán contiguas en memoria virtual, con lo cual la transferencia se parte en transferencias pequeñas dentro de una página. El sistema operativo debe asegurarse que no cambie la ubicación de una página mientras se está realizando una transferencia por DMA que involucre a la misma.

Arbitraje de bus: El arbitraje permite decidir qué maestro va a usar el bus para transmitir información en un momento determinado. El concepto tiene sentido cuando hay más de un maestro que requiere utilizar el bus. Existe el arbitraje centralizado, y el arbitraje no centralizado. En el arbitraje centralizado, es un elemento central el que lo efectúa, y en el descentralizado, se realiza de manera conjunta y distribuida.

**Discos y medios de almacenamiento:**

• ¿Cuáles son los parámetros que intervienen en la performance de un disco rígido?

Los parámetros son que intervienen en la performance de un disco rígido son los siguientes:

· La velocidad de rotación de las pistas (expresada en rpm).

· Tiempo de posicionamiento del cabezal, es el tiempo que tarda el cabezal de lectura/escritura en posicionarse sobre la pista donde están los datos a acceder. Si este tiempo es menor, entonces mejora la velocidad de acceso a los datos.

· Latencia de rotación, es el tiempo requerido que tarda el sector deseado en colocarse bajo el cabezal de lectura/escritura. Está relacionado con la velocidad de rotación.

· Tiempo de transferencia, es el tiempo necesario para transferir un bloque de bits.

· Otras prestaciones a considerar son: la capacidad de almacenamiento, la confiabilidad frente a fallos, calidad de los materiales y hermeticidad frente a posibles daños físicos como caídas, golpes, salpicaduras, resistencia a temperaturas externas poco habituales, y demás factores físicos. Algunos otros aspectos a tener en cuenta en este tipo de dispositivos es la posibilidad de recuperar la información frente a errores lógicos o algún tipo de daño interno, la fiabilidad (tiempo promedio del servicio sin fallas), y la disponibilidad.

**Esquemas RAID:**

• ¿Cómo mejora Raid 0 la disponibilidad del sistema? ¿Previene los errores lógicos? ¿Previene los errores fiscos?

RAID 0 no mejora la disponibilidad del sistema porque es un esquema en el cual no existe redundancia, por lo tanto, no proporciona una capacidad recuperación de información si un disco falla, entonces no previene los errores lógicos. Si una información es borrada, no se va a poder recuperar y se perderá. Así también, ante un fallo de algún disco, se perderá la información y el esquema RAID 0 no me permitirá recuperarla. El sistema deja de funcionar ante un fallo. En RAID 0 las capacidades de los discos involucrados se suman y se pueden utilizar ambos en su totalidad. La mejora más sustancial que proporciona RAID 0 es el uso en paralelo de estos discos, lo cual mejora la velocidad de acceso a ellos y por consiguiente, el rendimiento del sistema.

• ¿En qué consiste RAID 1? ¿Cuáles son sus desventajas?

RAID I es el esquema de redundancia más caro. Consiste en disponer de discos que almacenan información en espejo, es decir, lo que se guarda en uno también se guarda en otro. Esto permite aumentar la confiabilidad del sistema de almacenamiento tolerando el fallo de alguno de los discos, de esta manera el sistema seguirá funcionando ante la caída de alguno de los discos utilizando el otro como respaldo y pudiendo recuperar la información del disco caído desde el otro disco espejo en funcionamiento. La capacidad utilizable de este esquema es igual a la capacidad del disco más pequeño y el sistema operativo ve un único disco lógico del tamaño del disco más pequeño.

• ¿En qué casos RAID 5 es útil y en qué casos no?

La utilización de RAID 5 es importante en servidores que requieren una alta confiabilidad en el manejo de grandes cantidades de información, también en casos donde se requiere un equilibrio en costo, rendimiento, redundancia y eficiencia en sistemas de almacenamiento. Con RAID 5, si un disco falla, se puede seguir utilizando el sistema y recuperar la información. Mientras esto sucede, la controladora de disco está sumamente exigida hasta reemplazar el disco dañado. RAID 5 soporta la falla de un disco, pero si falla más de uno ya no se puede sostener el funcionamiento del sistema ni conservar toda la información. Con RAID 5 la controladora de disco debe manejar la información de paridad en todos los discos, esa información es la que se utiliza para recuperar datos si ocurre alguna falla. RAID 5 no es muy eficiente en casos donde las escrituras se efectúan constantemente, porque empeoraría la performance del sistema teniendo que almacenar la paridad en todo momento. No se utiliza en computadoras de uso común como las de escritorio.

• ¿En qué consiste el RAID 5? ¿Cuáles son sus ventajas?

RAID 5 consiste en una colección de discos de trabajan en paralelo para mejorar el rendimiento y aumentar la confiabilidad del sistema de almacenamiento. En el cual se escriben los datos en cada uno de los N-1 discos porque en el disco N se escribe la paridad, este disco N no es siempre el mismo para distribuir la paridad lo cual tiene varias ventajas, al distribuir la paridad en todos los discos si algún disco falla la controladora debe sacar de los demás discos la información de paridad para rearmar la información lo cual si toda esta información está en un mismo disco como en RAID 4 genera un cuello de botella importante, entonces al estar distribuida evito el cuello de botella y mejoro la performance notablemente. Esta estrategia mejora mucho el coste del sistema porque, por ejemplo:

Tengo 5 discos, disco1 de 1TB, disco2 de 1TB, disco3 de 1TB, disco4 de 1TB, disco5 de 1TB.

El sistema operativo ve un disco de 4TB, lo cual mejora muchísimo el coste. Esta estrategia soporta solamente la caída de un disco.

• Explique qué ventajas brinda RAID 5 sobre un disco convencional.

RAID 5 brinda varias ventajas sobre un disco convencional:

Una ventaja es que almacena los datos en varios discos, lo cual al trabajar en paralelo mejora la performance del sistema notablemente. Otra ventaja es que esta estrategia aumenta la confiabilidad y disponibilidad del sistema de almacenamiento porque puede tolerar fallos de disco, si un disco falla tengo la paridad distribuida en cada disco, con lo que la controladora va a obtener la información de paridad de cada disco para rearmar la información y reparar el disco, también me permite utilizar la técnica "hot swap" que si falla un disco me permite cambiarlo en caliente, es decir, no necesito apagar mi sistema y me permite que siga funcionando lo cual es importantísimo.

• ¿Cuáles son las ventajas que RAID en general, qué mejoras incorpora RAID 5?

Las ventajas que introduce RAID son varias, al almacenar los datos en varios discos me permiten trabajar en paralelo y mejorar la performance de mi sistema notablemente, también esta estrategia aumenta la confiabilidad y fiabilidad de mi sistema de almacenamiento tolerando fallos de disco, entonces genera información redundante lo cual al fallar un disco puedo rearmar la información o utilizar el disco redundante y arreglar el disco roto.

• **Sistemas embebidos:** son sistemas que fueron diseñados y preparados para cumplir con propósitos específicos, por lo que sus componentes, capacidades y prestaciones están muy ajustadas a esos propósitos. Al diseñar y fabricar sistemas embebidos con funcionalidades específicas, uno tiene la ventaja de dedicar todos los recursos que va a utilizar en el sistema a las funciones que este tiene que cumplir, entonces se ahorran costos quitando versatilidad al sistema. Las funciones a cumplir por el sistema embebido son delimitadas, por lo tanto, los recursos invertidos en sus componentes y prestaciones también lo son. Por esta razón, se puede decir que los sistemas embebidos son relativamente económicos de producir e implementar.

Sistemas embebidos de baja gama vs alta gama:

|  |  |
| --- | --- |
| **Baja Gama** | **Alta Gama** |
| No posee sistema operativo. | Puede ejecutar sistemas operativos (no es excluyente). |
| No utiliza memoria virtual. | Puede utilizar memoria virtual. |
| Son sistemas más económicos, con características reducidas. | Son sistemas más sofisticados, completos y costosos. |
| Utiliza microcontroladores. | Utilizan microprocesadores. |
| Cuentan con un set de instrucciones acotadas. | Cuentan con un set de instrucciones más amplias. |
| No tiene soporte de video. | Pueden tener soporte de video. |
| No tiene soporte de operaciones de punto flotante. | Puede tener soporte de operaciones de punto flotante. |

• PC vs Servidor:

|  |  |
| --- | --- |
| **Servidores** | **PCs comunes** |
| Trabajan las 24 hs. para poder sostener las múltiples utilidades que dependen de su correcto funcionamiento. | El tiempo de trabajo depende de la utilidad que se le da al equipo, rara vez se utilizan más de 4 o 5 horas seguidas. |
| Presentan redundancia de varios discos, varias placas de red, fuentes de alimentación (funcionando en fases distintas) | Está pensada para alojar un dispositivo de almacenamiento o dos (rara vez más). Lo mismo con la fuente y la placa de red. |
| Están equipados con gran cantidad de memoria RAM | La cantidad de RAM es variable según el propósito de la máquina, pero puede tener una cantidad pequeña como 2 o 4 GB tranquilamente. |
| Sus componentes internos cumplen con estándares de calidad superiores a los de una PC común. | Sus componentes pueden ser de diversas calidades y fabricantes, depende del usuario y el presupuesto. |
| La confiabilidad es un factor clave que debe priorizarse siempre. |  |
| Deben contar con múltiples procesadores. | Puede contar con varios procesadores, pero no es un requisito excluyente para su funcionamiento (puede tener uno sin problemas). |
| Su controladora de disco debe tener un desempeño considerable, y ser compatible con RAID 5. | La controladora de disco puede tener compatibilidad con RAID 0, o RAID 1. |
| Su funcionamiento no debe depender del sistema operativo. | El sistema operativo ocupa un lugar fundamental en el funcionamiento de la PC. |
| Requiere soporte de hardware que permita trabajar con virtualización |  |
| Cuentan con una BIOS más inteligente. |  |
| Están situados en ambientes controlados con una temperatura estable determinada, y demás condiciones de humedad y baja cantidad de partículas en suspensión (polvo). |  |
| Deben estar estrictamente monitoreados con sensores, indicadores y mediciones que evalúen y diagnostiquen su funcionamiento permanentemente. |  |

• Inconvenientes de la implementación monociclo de un procesador:

CPI fijo = 1, para todas las instrucciones. El ciclo de reloj está definido por la instrucción más lenta, por lo que el procesador termina siendo poco eficiente en cuanto a la optimización del tiempo como recurso. No se puede reutilizar ninguna unidad funcional. Si se utiliza una arquitectura más compleja que MIPS R2000, se agravan los inconvenientes y problemas.

Implementación de procesador multiciclo: Se dedican varios ciclos de reloj a cada instrucción. Se necesitan más multiplexores y más registros. Cada dato y resultado se guardará en un registro para que no se pierda una vez terminado cada ciclo. Esta implementación es más cara que la implementación monociclo ya que necesita más conexiones y componentes electrónicos, aunque como ventaja, se puede considerar que es un poco más rápida que la implementación monociclo.

Segmentación/Pipeline: Es una técnica utilizada para optimizar el tiempo de ejecución de procesos que se realizan mediante la repetición de una secuencia de pasos básicos. Permite la ejecución de procesos concurrentemente. A aplicar esta técnica, se intenta separar el proceso en etapas y ejecutar cada etapa en un recurso independiente.

La idea de la segmentación es mejorar la productividad, aumentando el número de instrucciones ejecutadas por unidad de tiempo.

· Funcionamiento: Cuando una etapa del proceso termina, el recurso liberado puede empezar a ejecutar la misma etapa del siguiente proceso.

· Se consigue la ejecución de varios procesos en paralelo, cada uno en una etapa diferente.

· Las etapas son ejecutadas de manera secuencial.

1. Explicar cómo funciona la unidad de detección de riesgo. (Alejandro)

En ciertos casos, además de aplicar anticipación de resultados para evitar riesgos en el pipeline, es necesario bloquear el mismo durante alguna etapa. Por ejemplo, en los casos donde aparece un riesgo de dato de una carga (load-use data hazard), es decir cuando una instrucción situada después de una carga intenta leer el registro en el que se escribe la carga. Ante este tipo de riesgos es que se debe aplicar lo que se llama pipeline stall, también conocido como burbuja o bubble. Este recurso de insertar burbujas es útil para tratar casos difíciles tales como riesgos de datos particularmente intrincados.

Entonces, además de una unidad de anticipación de datos, es necesaria una unidad de detección de riesgos. Es importante que la misma funcione durante la etapa ID de tal manera que pueda insertarse un bloqueo entre el load y la instrucción que lo usa. Para poder ejercer un control de detección de riesgos cuando se tiene que comprobar una instrucción de carga (load) debemos tener estas condiciones en cuenta:

Primero se debe corroborar si la instrucción que se está comprobando es una carga o no. En este punto es donde se requiere una lectura de memoria. Luego se debe comprobar si el campo de registro destino de la carga en la etapa EX es igual a cualquiera de los dos registros fuente de la instrucción en ID. Si la condición se cumple, la instrucción se bloquea durante 1 ciclo. Después de este bloqueo de un ciclo, la lógica de anticipación de datos puede resolver la dependencia y la ejecución continúa. (Si no hubiera anticipación de datos, entonces las instrucciones tendrían que bloquearse otro ciclo). Si la instrucción situada en la etapa ID se bloquea, entonces también debe bloquearse la que esté en IF; de no hacerse así se perdería la instrucción buscada de memoria. Para que las dos instrucciones no sigan adelante, el procesador debe simplemente evitar que cambien el registro PC y el registro de segmentación IF/ID. Si estos valores no cambian, la instrucción en la etapa IF continuará leyendo de memoria usando el mismo PC y se volverán a leer los registros en la etapa ID usando los mismos campos de instrucción en el registro de segmentación IF/ID. A todo esto, la mitad posterior del pipeline que comienza en la etapa EX ejecuta “nops”, las cuales son instrucciones que no tienen ningún tipo de efecto alguno.

Hablando de soluciones de riesgos en el pipeline, estas pueden aplicarse mediante alguna incorporación de algún recurso de hardware, como por ejemplo implementando detección y bloqueos, o bien aplicando alguna modificación de software, como por ejemplo reordenando el código para evitar tener que recurrir a los bloqueos anteriormente mencionados, siendo que el uso de estos da como resultado una ineficiencia en cuanto a los tiempos de procesamiento.

1. Dado un código en CUDA, explicar en qué consiste y qué funcionalidades presenta. (Alejandro)

\_\_global\_\_ matrix\_add\_gpu (fload \*A, float \*B, float \*C, int N){

int i = blockIdx.x \* blockDim.x + threadIdx.x;

int j = blockIdx.y \* blockDim.y + threadIdx.y;

int index = i + j\*N;

if (i<N && j<N){

C[index] = A[index] + B[index];

}

}

int main(){

dim3 dimBlock(blocksize, blocksize);

dim3 dimGrid(N/dimBlock.x, N/dimBlock.y);

matrix\_add\_gpu<<<dimGrid, dimBlock>>>(a, b, c, N);

}

Implementación de suma de matrices en CUDA.

En el código vemos la palabra clave específica de CUDA “global” ante la declaración de la matriz. “global” indica que esta función es un kernel y que hay que llamarlo desde el procesador principal para generar el grid de flujos que ejecutará el kernel en el dispositivo. El código que se ejecuta en el dispositivo (kernel) es la función que ejecutan los diferentes flujos durante la fase paralela, cada uno en el rango de datos que le corresponde. Es importante que recordar que CUDA sigue el modelo SPMD (single-program multiple-data) y, por lo tanto, todos los flujos ejecutan el mismo código.

1. Dado un código en OpenMP, explicar en qué consiste y qué funcionalidades presenta. (Alejandro)
2. Comparar las 3 organizaciones de la caché. (Walter)

· Cache de correspondencia directa: al usar caché de correspondencia directa, a cada bloque de memoria principal se le asigna un lugar específico en la memoria caché. Es muy estricta en el ordenamiento de los bloques. Este tipo de organización es simple y fácil de gestionar, pero es la que tiene la tasa de fallos más alta. Hoy en día no se utiliza con frecuencia.

· Cache asociativa: utilizando caché asociativa, a cada bloque de memoria principal se le asigna aleatoriamente cualquier lugar de la memoria caché, lo cual hace que este tipo de organización sea flexible, pero al mismo tiempo existe una posibilidad de que ocurran colisiones. La memoria caché completamente asociativa es la que tiene la tasa de fallos más baja. No obstante, no es la organización más utilizada ya que es la más cara y compleja de gestionar debido a que examina en paralelo todas las etiquetas para dar con la dirección que corresponde a un determinado bloque de memoria principal.

· Cache asociativa por vías/conjuntos: Resulta ser una solución de compromiso entre la organización de mapeo directo y la completamente asociativa. Un bloque de la memoria principal se ubica de manera determinada en una de las vías de la memoria caché, pero dentro de esa vía puede disponerse en cualquier conjunto. Es una combinación entre la organización de correspondencia/mapeo directo y la completamente asociativa, para ubicar un determinado bloque se busca primero de manera directa, y luego de manera asociativa. Este tipo de organización es muy utilizada, mejora la prestación de la cache directa y aprovecha la flexibilidad de la caché asociativa. Mientras más vías tiene este tipo de organización, más se va a asemejar a la de correspondencia directa, y mientras más conjuntos tenga, más se va a asimilar a la organización completamente asociativa.

1. Explicar cómo se informa la causa de interrupción en MIPS y cómo se decide si se va a tratar o no la misma. (Walter)

Para que el sistema operativo pueda tratar una interrupción, primero debe conocer las razones por las que se ha producido, además de la instrucción que la ha causado. Para esto, MIPS cuenta con un registro de estado denominado registro de causa (cause register), que contiene un campo que indica el motivo de la excepción. Adicionalmente al registro de causa, necesitamos otro registro adicional llamado EPC y también nos conviene incorporar algunas señales de control a nuestra implementación, lo que nos permite ampliar relevantemente el control. A continuación, se detalla la relevancia que toma cada registro que se debe agregar al camino de datos de los que hemos hablado hasta ahora:

· Registro EPC: registro de 32 bits utilizado para guardar la dirección de la instrucción afectada que se ha interrumpido.

· Registro de Causa: registro donde se va a almacenar la causa de la excepción. En MIPS, este registro es de 32 bits, aunque algunos de estos bits normalmente no se utilizan. Se supone que existe un campo de cinco bits para codificar las dos posibles fuentes de excepciones que nos interesan (instrucción no definida y desbordamiento aritmético). El registro de Causa de MIPS registra todas las posibles excepciones que se pueden dar en un ciclo de reloj, por lo que la rutina de gestión de la excepción debe asociar el tipo de la excepción producida a la instrucción que la provoca. Una pista importante para hacerlo es saber qué tipo de excepción puede ocurrir en cada una de las etapas de segmentación.

Las excepciones se recogen en el registro de causa de modo que, una vez resuelta la gestión de la excepción de una instrucción anterior, el hardware se pueda interrumpir en una nueva excepción causada por una instrucción posterior.

Además, pueden aparecer múltiples excepciones de forma simultánea en un mismo ciclo. La solución más común es dar prioridad a las excepciones para determinar cuál de ellas se debe atender primero. En la mayoría de las implementaciones de procesadores MIPS, el hardware ordena las excepciones de modo que sea la instrucción más antigua la que se vea interrumpida. Las peticiones de un dispositivo de E/S y el mal funcionamiento del hardware no están asociados a una instrucción específica, por lo que hay cierta flexibilidad a la hora de tratarlas, como al decidir el momento en el que se debe interrumpir el pipeline. Así, usar el mismo mecanismo que se usa para las otras excepciones funciona perfectamente.

La computadora y el sistema operativo deben trabajar conjuntamente para que las excepciones se puedan atender de manera ordenada, coordinada y sin comportamientos inesperados.

Normalmente, el hardware se encarga de detener la instrucción que causa la excepción en medio del flujo de instrucciones, dejar que las instrucciones anteriores se completen, eliminar todas las instrucciones posteriores, modificar un registro para que muestre la causa de la excepción, guardar la dirección de la instrucción que la ha provocado y entonces saltar a una dirección predefinida.

El sistema operativo es el que evalúa la causa de la excepción y actúa en consecuencia. En el caso de una instrucción no definida, del mal funcionamiento del hardware o de una excepción producida por desbordamiento, el sistema operativo normalmente detiene la ejecución del programa y devuelve un indicador de la razón de esta detención.

• Explique en MIPS cómo se guarda la dirección de retorno en un llamado a interrupción. Identifique cómo se soporta en el datapath.

En un llamado a interrupción, la dirección de retorno se guarda en el registro EPC (registro contador de programa de excepciones). Cuando se termina de atender la interrupción, se chequea el registro EPC para volver a retomar la ejecución del programa. En el datapath se agrega este registro y se modifica la unidad de control agregando una señal EscEPC para escribir la dirección de retorno en un llamado a interrupción.