Resumen Arquitectura de Computadoras

Subrutinas: innovación en lenguajes de programación. Programa auto-contenido que puede invocarse desde cualquier punto de un programa mediante instrucción CALL. Brinda economía (código usado varias veces) y modularidad (subdivisión en unidades pequeñas). Requiere pasaje de argumentos (parámetros) que pueden ser por valor (copia de una variable) o por referencia (dirección de la variable)

Pasaje de argumentos a subrutinas
●Vía registros: el número de registros es la principal limitación. Es importante documentar que registros se usan
●Vía memoria: se usa un área definida de memoria (RAM). Difícil de estandarizar
●Vía pila (stack): Es el método más ampliamente usado. El verdadero "pasaje de parámetros". Independiente de memoria y registros. Hay que comprender bien como funciona porque la pila (stack) es usada por el usuario y por el sistema. En x86, SP apunta al último lugar usado

PILA

Una pila es un conjunto ordenado de elementos, en el que sólo uno de ellos es accesible en un instante dado. El punto de acceso se denomina cabecera de la pila. El número de elementos de la pila, o longitud, es variable. Sólo se pueden añadir o eliminar elementos en la cabecera de la pila. Por esta razón, una pila también se denomina lista último en entrar-primero en salir (LIFO, last in first out)
Una operación PUSH (apilar) añade un nuevo elemento en la cabecera de la pila. Una operación POP (desapilar) elimina el elemento de la cabecera de la pila. En ambos casos la cabecera experimenta el cambio apropiado.
Una operación unaria realiza una operación con el elemento de la cabecera de la pila. Sustituye el elemento de la cabecera con el resultado.
Una operación binaria realiza una operación con dos elementos de la cabecera de la pila. Elimina de la pila dichos elementos y pone el resultado de la operación en la cabecera de la pila.

La implementación de una pila depende en parte de sus usos potenciales. Si se desean ejecutar operaciones con la pila disponibles para el programador, el repertorio de instrucciones dispondrá de operaciones orientadas al manejo de la pila, incluyendo PUSH, POP y operaciones que utilicen uno o dos elementos de la cabecera de la pila como operandos. Ya que todas estas operaciones hacen referencia a una misma posición, la cabecera de la pila, la dirección del operando u operandos está implícita, y no necesita incluirse en la instrucción. Son instrucciones con 0 direcciones.
Si el mecanismo de pila va a ser utilizado sólo por la CPU, con usos tales como el manejo de procedimientos, en el repertorio de instrucciones no se contemplarían instrucciones orientadas al uso de la pila. En cualquier caso, la implementación de una pila requiere que exista un cierto conjunto de posiciones, utilizadas para almacenar los elementos de la pila. En memoria principal (o virtual) se reserva un bloque de posiciones contiguas para la pila. La mayor parte del tiempo el bloque está parcialmente lleno con elementos de la pila, y el resto del bloque está disponible para que crezca la pila. Para un funcionamiento correcto se necesitan tres direcciones, normalmente memorizadas en registros de la CPU:

• Puntero de pila: contiene la dirección tope o cabecera de la pila. Si se añade o se elimina un elemento de la pila, el puntero se incrementa o decrementa para que contenga la dirección de la nueva cabecera.
• Base de la pila: contiene la dirección base del bloque reservado para la pila. Si se intenta un pop cuando la pila está vacía se informa de un error.

• Límite de la pila: contiene la dirección del otro extremo del bloque reservado. Si se intenta un PUSH cuando el bloque está utilizado en su totalidad, se informa de un error.

Tradicionalmente, en la mayoría de las máquinas actuales la base de la pila coincide con la dirección más alta del bloque reservado para la pila, mientras que el límite se corresponde con la posición más baja. Por lo tanto la pila crece desde direcciones más altas hacia direcciones más bajas. Para acelerar las operaciones con la pila, también los dos elementos de la cabecera se almacenan a menudo en registros.

Interrupciones

Mecanismo mediante el cual se puede interrumpir el procesamiento normal de la CPU (ejecución secuencial de instrucciones de un programa). Pueden ser de origen interno o externo a la CPU.

Las interrupciones proporcionan una forma de mejorar la eficiencia del procesador. Por ejemplo, la mayoría de los dispositivos externos son mucho más lentos que el procesador. Si se estuvieran transfiriendo datos a una impresora, después de cada operación de escritura, el procesador tendrá que parar y permanecer ocioso hasta que la impresora complete la escritura. La longitud de esta pausa puede ser del orden de muchos cientos, o incluso miles, de ciclos de instrucción. Con el uso de las interrupciones, el procesador puede dedicarse a ejecutar otras instrucciones mientras una operación de E/S está en curso.
El programa de usuario llega a un punto en el que realiza una llamada al sistema para realizar una escritura (WRITE). El programa de E/S al que se llama en este caso, está constituido por el código de la preparación y la orden de E/S propiamente dicha. Después de que estas pocas instrucciones se hayan ejecutado, el control se devuelve al programa de usuario. Mientras tanto, el dispositivo externo está ocupado aceptando el dato de la memoria del computador e imprimiéndolo. Esta operación de E/S se realiza concurrentemente con la ejecución de instrucciones del programa de usuario. Cuando el dispositivo externo pasa a estar preparado para actuar, el módulo de E/S envía una señal de petición de interrupción al procesador. El procesador responde suspendiendo la operación del programa que estaba ejecutando y salta a un programa conocido como “gestor de interrupción”, que da servicio a ese dispositivo concreto y prosigue con la ejecución del programa original. Cuando el procesamiento de la instrucción se completa, la ejecución prosigue.

Para permitir el uso de interrupciones, se añade el ciclo de interrupción al ciclo de instrucción. En el ciclo de interrupción el procesador comprueba si se ha generado alguna interrupción, indicada por la presencia de una señal de interrupción. Si no hay señales de interrupción pendientes, el procesador continúa con el ciclo de captación y accede a la siguiente instrucción del programa en curso. Si hay alguna interrupción pendiente, el procesador hace lo siguiente:

1) Suspende la ejecución del programa en curso y guarda su contexto. Esto significa almacenar la dirección de la siguiente instrucción a ejecutar (contenido actual del contador del programa) y cualquier otro dato relacionado con la actividad en curso del procesador.

2) Carga el contador del programa con la dirección de comienzo de una rutina de gestión de interrupción.

A continuación el procesador prosigue con el ciclo de captación y ejecución. Generalmente el programa de gestión de interrupción forma parte del sistema operativo.

Con las interrupciones deben ejecutarse instrucciones extra para determinar el origen de la interrupción y decidir la acción apropiada. No obstante, el procesador se emplea más eficientemente

que esperando la operación de E/S.

Clases de interrupciones

Programa: generadas por alguna condición que se produce como resultado de la ejecución de una instrucción, tal como desbordamiento aritmético (overflow), división por cero, instrucción de máquina inexistente, acceso fuera del espacio de memoria.
Temporización: generadas por un temporizador interno al procesador. Permite al SO realizar ciertas funciones de manera regular.
E/S: generadas por un controlador de E/S para indicar la finalización sin problemas de una operación o error.
Fallo de hardware: generadas por un fallo tal como la falta de potencia de alimentación o un error de paridad en la memoria.

Interrupciones múltiples

Un programa puede estar recibiendo datos a través de una línea de comunicación e imprimir los resultados, por lo que tendría múltiples interrupciones (cada vez que llegue un dato y cada vez que se termine de escribir). Por lo tanto, para tratar las interrupciones múltiples se pueden seguir dos alternativas:
•Desactivar las interrupciones mientras se está procesando una interrupción, por lo que el procesador ignorará la interrupción y la dejará pendiente para cuando termine de atender la interrupción actual (no puede decidir entre interrupciones más importantes que otras). Las interrupciones se manejan en un orden secuencial estricto.
•Definir prioridades para las interrupciones, permitiendo que una segunda interrupción de prioridad más alta pueda interrumpir a un gestor de interrupción de prioridad menor. Cuando se ha gestionado la interrupción de prioridad más alta, el procesador vuelve a las interrupciones previas (de menor prioridad). Terminadas todas las rutinas de gestión de interrupciones se retoma el programa del usuario.

Procesamiento de la interrupción

Cuando se produce una interrupción, se disparan una serie de eventos en el procesador, tanto a nivel hardware como software:

1) El dispositivo envía una señal de interrupción al procesador.
2) El procesador termina la ejecución de la instrucción en curso antes de responder a la interrupción.
3) El procesador comprueba si hay interrupciones, determina que hay una y envía una señal de reconocimiento al dispositivo que originó la interrupción. La señal de reconocimiento hace que el dispositivo desactive su señal de interrupción.
4) El procesador necesita prepararse para transferir el control a la rutina de interrupción. Para empezar, debe guardar la información necesaria para continuar el programa en curso en el punto en que se interrumpió. La información mínima que se precisa es a) el estado del procesador, que se almacena en el registro PSW (program status word) y b) la posición de la siguiente instrucción a ejecutar, que está en el PC.
5) El procesador carga en el PC la posición de inicio del programa de gestión de la interrupción solicitada y se continúa con el ciclo de instrucción siguiente, que empieza con el de la captación de la instrucción. El control se transfiere al programa de gestión de interrupción.
6) Además de guardar el PC y el PSW en la pila, se deben guardar los contenidos de los registros del procesador, ya que pueden ser utilizados por la rutina de interrupción.

7) La rutina de gestión de interrupción puede continuar ahora procesando la interrupción.
8) Cuando el procesamiento de la interrupción ha terminado, los valores de los registros almacenados se recuperan de la pila y se vuelven a almacenar en los registros.
9) El paso final es recuperar los valores del PSW y del PC desde la pila. Como resultado, la siguiente instrucción que se ejecute pertenecerá al programa previamente interrumpido.

Es importante almacenar toda la información del estado del programa interrumpido para que éste pueda reanudarse.

Cuestiones de diseño: Reconocimiento de interrupciones

Múltiples líneas de interrupción: proporciona varias líneas de comunicación entre el procesador y los módulos de E/S. Poco práctico.

Consulta software(software polling): cuando el procesador detecta una interrupción, se produce una consulta a cada módulo de E/S, por una línea específica donde están conectados todos los dispositivos, para determinar el módulo que ha provocado la interrupción. Consume mucho tiempo.

Conexión en cadena (Daisy Chain): todos los módulos de E/S comparten una línea común y cuando el procesador recibe una interrupción, activa el reconocimiento de la interrupción. Esta señal se propaga a través de una secuencia de módulos de E/S encadenados hasta encontrar el que la solicitó. Este módulo responde colocando una palabra en las líneas de datos, llamada vector, que es la dirección del módulo de E/S o algún identificador específico.

Arbitraje de bus: un módulo de E/S debe en primer lugar disponer del control del bus antes de poder activar la línea de petición de interrupción, así solo uno puede activar la línea en un instante. También usa un vector.

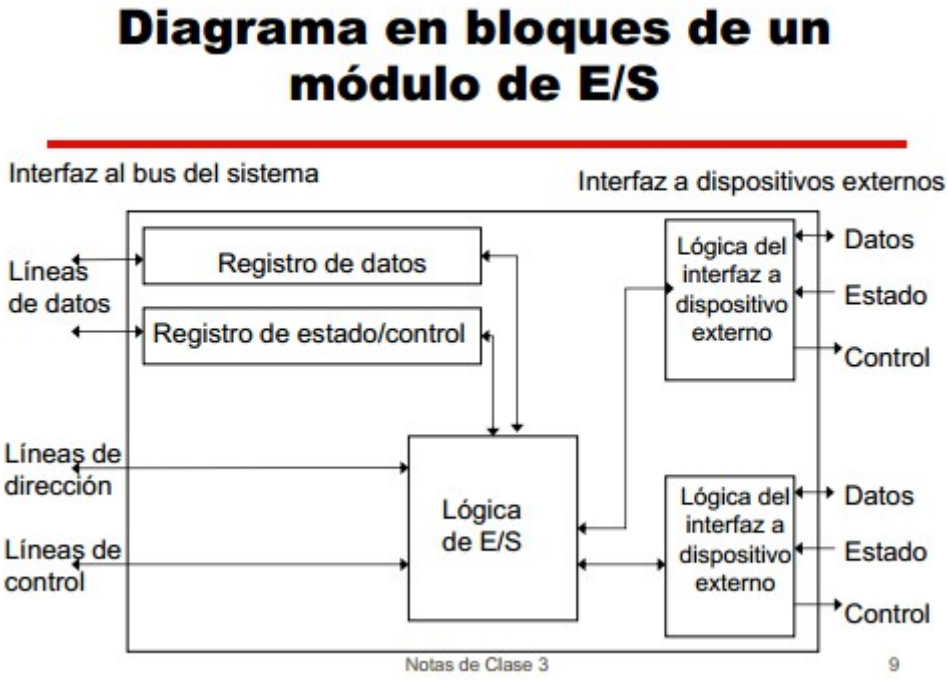
Módulo de E/S

Funciones:

• Control y temporización: coordina el tráfico entre los recursos internos (memoria principal, bus del sistema) y los dispositivos externos.
• Comunicación con el procesador: el módulo de E/S debe decodificar órdenes del procesador (por ej. el controlador de un disco puede recibir leer sector, escribir sector, buscar n° de pista), intercambiar datos con el procesador, informar el estado del dispositivo (BUSY o READY), y reconocer la dirección de cada uno de los periféricos que controla.
• Comunicación con los dispositivos: intercambiar órdenes, información de estado y datos.
• Almacenamiento temporal de datos (data buffering): la velocidad de transferencia desde y hacia memoria principal o procesador es más alta que la de los periféricos, por lo que los datos provenientes de la memoria se envían al módulo de E/S en ráfagas rápidas. Los datos se almacenan temporalmente en el módulo de E/S y se envían al periférico a la velocidad de éste. En sentido contrario, los datos se almacenan para no mantener a la memoria ocupada con una transferencia lenta.
• Detección de errores: debe informar de defectos mecánicos o de errores de transmisión al procesador.

Estructura de un módulo de E/S

El módulo se conecta al resto del computador a través de un conjunto de líneas (por ej, líneas del bus del sistema). Los datos que se transfieren a y desde el módulo se almacenan temporalmente en uno o más registros de datos. Además, puede haber uno o más registros de estado que proporcionan información del estado presente. Un registro de estado también puede funcionar como un registro de control, para recibir la información de control del procesador. La lógica que hay en el módulo interactúa con el procesador a través de una serie de lineas de control. Éstas son las que utiliza el procesador para proporcionar órdenes al módulo de E/S. El módulo también debe ser capaz de reconocer y generar las direcciones asociadas a los dispositivos que controla. Cada módulo de E/S tiene una dirección única o, si controla más de un dispositivo externo, un conjunto único de direcciones. Por último, el módulo de E/S posee la lógica específica para la interfaz con cada uno de los dispositivos que controla.
El funcionamiento de un módulo de E/S permite que el procesador vea una amplia gama de dispositivos de una forma simplificada. El módulo debe ocultar los detalles de temporización, formatos y electromecánica de los dispositivos externos para que el procesador pueda funcionar únicamente en términos de órdenes de lectura y escritura o abrir y cerrar ficheros.
Un módulo de E/S que presenta al procesador una interfaz de alto nivel se denomina canal de E/S, mientras que un módulo simple que requiere un control detallado se denomina controlador de E/S.



Son posibles 3 técnicas para las operaciones de E/S

E/S programada: los datos se intercambian entre el procesador y el módulo de E/S. El procesador ejecuta un programa que controla directamente la operación de E/S, incluyendo la comprobación del estado del dispositivo, el envío de una orden de lectura o escritura y la transferencia del dato. Cuando el procesador envía una orden al módulo de E/S, debe esperar hasta que la operación de E/S concluya. Si el procesador es más rápido que el módulo de E/S, el procesador desperdicia este tiempo.

E/S mediante interrupciones: el procesador proporciona la orden de E/S, continúa ejecutando otras instrucciones, y es interrumpido por el módulo de E/S cuando éste ha terminado su trabajo. Tanto con E/S programada como con interrupciones, el procesador es responsable de extraer los datos de la memoria principal en una salida, y de almacenar los datos en la memoria principal en

una entrada. La alternativa se conoce como **acceso directo a memoria** (DMA), donde el módulo de E/S y la memoria principal intercambian datos directamente, sin la intervención del procesador.

Acceso directo a memoria

La E/S con interrupciones, aunque más eficiente que la sencilla E/S programada, también requiere la intervención activa del procesador para transferir datos entre la memoria y el módulo de E/S, y cualquier transferencia de datos debe seguir un camino a través del procesador. Por tanto se presentan dos inconvenientes:

1) La velocidad de transferencia de E/S está limitada por la velocidad a la cual el procesador puede comprobar y dar servicio a un dispositivo.
2) El procesador debe dedicarse a la gestión de las transferencias de E/S; se debe ejecutar cierto número de instrucciones por cada transferencia de E/S.

Existe un cierto compromiso entre estos dos inconvenientes. Considérese una transferencia de un bloque de datos. Utilizando E/S programada, el procesador se dedica a la tarea de la E/S y puede transferir datos a alta velocidad al precio de no hacer nada más. La E/S con interrupciones libera en parte al procesador a expensas de reducir la velocidad de E/S. No obstante, ambos métodos tienen un impacto negativo, tanto en la actividad del procesador como en la velocidad de transferencia de E/S.
Cuando hay que transferir grandes volúmenes de datos, se requiere una técnica más eficiente: el acceso directo a memoria (DMA)

Funcionamiento del DMA

El DMA requiere un módulo adicional en el bus del sistema. El módulo de DMA es capaz de imitar al procesador y, de hecho, es capaz de recibir el control del sistema cedido por el procesador. Necesita dicho control para transferir datos a, y desde, memoria a través del bus del sistema. Para hacerlo, el módulo de DMA debe utilizar el bus sólo cuando el procesador no lo necesita, o debe forzar al procesador a que suspenda temporalmente su funcionamiento. Esta última técnica es la más común y se denomina robo de ciclo (cycle stealing), puesto que, en efecto, el DMA roba un ciclo de bus, en el que se realiza la transferencia de una única palabra y luego el DMA libera el bus. El DMA solicita el control del bus tantas veces como sea necesario hasta finalizar la transferencia del bloque completo. También está el modo de transferencia ráfaga, en el que el DMA no libera el bus hasta haber finalizado la transferencia de todo el bloque de datos.
Cuando el procesador desea leer o escribir un bloque de datos, envía una orden al módulo de DMA incluyendo la siguiente información:
•Si se solicita una lectura o una escritura, utilizando la línea de control de lectura o escritura entre el procesador y el módulo de DMA.
•La dirección del dispositivo de E/S en cuestión, indicada a través de las líneas de datos.
•La posición inicial de memoria a partir de donde se lee o se escribe, indicada a través de las líneas de datos y almacenada por el módulo de DMA en su registro de direcciones.
•El número de palabras a leer o escribir, también indicado a través de las líneas de datos y almacenado en el registro de cuenta de datos.

Después el procesador continúa con otro trabajo. Ha delegado la operación de E/S al módulo de DMA que se encargará de ella. El módulo de DMA transfiere el bloque completo de datos, palabra a palabra, directamente desde, o hacia, la memoria, sin que tenga que pasar a través del procesador. Cuando la transferencia se ha terminado, el módulo de DMA envía una señal de interrupción al procesador. Así, el procesador sólo interfiere al principio y al final de la transferencia. El procesador no guarda el contexto, sólo espera durante un ciclo de bus. El procesador es más lento ejecutando

programas pero el DMA es mucho más eficiente que la E/S programada o mediante interrupciones para una E/S de varias palabras.
La lógica de DMA puede ser parte de un módulo de E/S o puede ser un módulo separado que controla a uno o más módulos de E/S. Se pueden conectar los módulos de E/S a un módulo DMA mediante un bus de E/S, reduciendo a uno el número de interfaces de E/S en el módulo de DMA, y permite una configuración fácilmente ampliable.

Canales de E/S

El canal de E/S representa una ampliación del concepto de DMA. Un canal de E/S puede ejecutar instrucciones de E/S, lo que le confiere un control completo sobre las operaciones de E/S. En un computador con tales dispositivos, la CPU no ejecuta instrucciones de E/S. Dichas instrucciones se almacenan en memoria principal para ser ejecutadas por un procesador de uso específico contenido en el propio canal de E/S. De esta forma, la CPU inicia una transferencia de E/S, indicando al canal de E/S que debe ejecutar un programa en la memoria. El programa especifica el dispositivo o dispositivos, el área o áreas de memoria para almacenamiento, la prioridad y las acciones a realizar en ciertas situaciones de error. El canal de E/S sigue estas instrucciones y controla la transferencia de datos.
Son comunes dos tipos de canales de E/S. Un *canal selector* controla varios dispositivos de velocidad elevada y, en un instante dado, se dedica a transferir datos a uno de esos dispositivos. Es decir, el canal de E/S selecciona un dispositivo y efectúa la transferencia de datos. Cada dispositivo o pequeño grupo de dispositivos es manejado por un *controlador*, o módulo de E/S, que es similar a los módulos de E/S de los que se ha discutido. Así, el canal de E/S se utiliza en lugar de la CPU para controlar estos controladores de E/S. Un *canal multiplexor* puede manejar las E/S de varios dispositivos al mismo tiempo. Para dispositivos de velocidad reducida, un multiplexor de byte acepta o transmite caracteres tan rápido como es posible a varios dispositivos. Para dispositivos de velocidad elevada, un multiplexor de bloque entrelaza bloques de datos de los distintos dispositivos.

Segmentación de instrucciones

La segmentación de instrucciones es similar al uso de una cadena de montaje en una fábrica de manufacturación. Una cadena de montaje saca partido del hecho de que el producto pasa a través de varias etapas de producción. Disponiendo el proceso de producción como una cadena de montaje, se puede trabajar sobre los productos en varias etapas simultáneamente. A este proceso se hace referencia como segmentación de cauce (pipelining) porque, como en una tubería o cauce (pipeline), en un extremo se aceptan nuevas entradas antes de que algunas entradas aceptadas con anterioridad aparezcan como salidas en el otro extremo.
Para aplicar este concepto a la ejecución de instrucciones, debemos darnos cuenta de que, de hecho, una instrucción tiene varias etapas. Como una aproximación sencilla, consideremos la subdivisión del procesamiento de una instrucción en dos etapas: captación y ejecución de instrucción. Hay períodos en la ejecución en los que no se accede a memoria principal y este tiempo podría utilizarse para captar la siguiente instrucción en paralelo con la ejecución actual (precaptación), acelerando la ejecución de instrucciones.
La segmentación de cauce (pipelining) es una forma particularmente efectiva de organizar el hardware de la CPU para realizar más de una operación al mismo tiempo. Consiste en descomponer el proceso de ejecución de las instrucciones en fases o etapas que permitan una ejecución simultánea. Explota el paralelismo entre las instrucciones de un flujo secuencial. Para conseguir una mejor prestación, el cauce debe tener más etapas.

Características

La segmentación es una técnica de mejora de prestaciones a nivel de diseño hardware y es invisible al programador. El diseño de procesadores segmentados tiene gran dependencia del repertorio de instrucciones. Supone que todas las tareas duran el mismo tiempo, las instrucciones siempre pasan por todas las etapas y todos las etapas pueden ser manejadas en paralelo. Sin embargo las suposiciones no son siempre verdaderas.

Atascos de un cauce (stall)

Situaciones que impiden a la siguiente instrucción que se ejecute en el ciclo que le corresponde.
• Estructurales: Provocados por conflictos por los recursos
• Por dependencia de datos: Dos instrucciones se comunican por medio de un dato
• Por dependencia de control: La ejecución de una instrucción depende de cómo se ejecute otra
Si resolvemos con paradas del cauce, se disminuye el rendimiento.

Soluciones a riesgos estructurales

• Duplicación de recursos hardware
• Separación en memorias de instrucciones y datos
• Turnar el acceso al banco de registros

Soluciones a riesgos de datos

Para riesgos RAW será necesario una unidad de detección de riesgos y/o compilador mas complejo
Dos soluciones:
• Hardware: el adelantamiento de operandos (forwarding) consiste en pasar directamente el resultado obtenido con una instrucción a las instrucciones que lo necesitan como operando sin esperar a la escritura.
• Software: instrucciones NOP o reordenación de código.

Prestaciones de un cauce segmentado

Cuanto mayor es el número de etapas del cauce, mayor es su potencial para conseguir aceleración. Sin embargo, los beneficios potenciales de etapas adicionales en el cauce se contrarrestan por los incrementos en coste, los retardos entre etapas, y el hecho de que se encontrarán saltos que requieran vaciar el cauce. Algo más allá del rango de 6 a 9 etapas, son contraproducentes.

Soluciones a riesgos de control: Tratamiento de saltos

El principal obstáculo es la instrucción de bifurcación condicional. Hasta que la instrucción no se ejecuta realmente, es imposible determinar si el salto se producirá o no. Hay varias aproximaciones en el tratamiento de bifurcaciones condicionales:

•Flujos múltiples: un cauce simple sufre penalización por las instrucciones de bifurcación, porque debe escoger una de las dos instrucciones a captar a continuación, y puede hacer la elección equivocada. Una solución burda es duplicar las partes iniciales del cauce y dejar que éste capte las dos instrucciones utilizando los dos caminos. Esta aproximación tiene dos problemas:

1) con cauces múltiples hay retardos debidos a la competencia por el acceso a los registros y a la memoria
2) pueden entrar en el cauce (en cualquiera de los dos flujos) instrucciones de bifurcación

adicionales antes de que se resuelva la decisión de la bifurcación original. Cada una de esas instrucciones exige un flujo adicional.

•Precaptar el destino del salto: cuando se identifica una instrucción de bifurcación condicional, se precapta la instrucción destino del salto, además de la siguiente a la de bifurcación. Si se produce el salto, el destino ya habrá sido precaptado.

•Buffer de bucles: es una memoria pequeña de gran velocidad, gestionada por la etapa de captación de instrucción del cauce, que contiene, secuencialmente, las n instrucciones captadas más recientemente. Si se va a producir un salto, el hardware comprueba en primer lugar si el destino del salto está en el buffer. En ese caso, la siguiente instrucción se capta del buffer. El buffer de bucles tiene tres utilidades:

1) el buffer se anticipa almacenando algunas instrucciones que secuencialmente están después de la dirección de donde se capta la instrucción actual. Así, las instrucciones que se capten secuencialmente estarán disponibles sin el tiempo de acceso a memoria habitual.
2) si ocurre un salto a un destino a sólo unas pocas posiciones más allá de la dirección de la instrucción de bifurcación, el destino ya estará en el buffer.
3) esta estrategia se acomoda particularmente bien al tratamiento de bucles: si el buffer de bucles es lo suficientemente grande como para contener todas las instrucciones de un bucle, entonces esas instrucciones sólo necesitan ser captadas de la memoria una vez, durante la primera iteración.

•Predicción de saltos: se pueden usar varias técnicas: predecir que nunca se salta, que siempre se salta, predecir según el código de operación, conmutador saltar/no saltar, o tabla de historia de saltos.

Las tres primeras soluciones son estáticas; no dependen de la historia de la ejecución que haya tenido lugar. Las dos últimas aproximaciones son dinámicas: dependen de la historia de la ejecución.
Las dos primeras soluciones son las más simples. Una asume que el salto no se producirá y continuará captando instrucciones secuencialmente, y la otra que el salto se producirá y siempre captará la instrucción destino del salto.
La última aproximación estática toma la decisión basándose en el código de operación de la instrucción de bifurcación. El procesador asume que el salto se producirá para ciertos códigos de operación de bifurcación y no para otros.
Las estrategias de bifurcación dinámicas intentan mejorar la exactitud de la predicción registrando la historia de las instrucciones de bifurcación condicional en un programa. Por ejemplo, a cada instrucción de bifurcación pueden asociarse uno o más bits que reflejen su historia reciente. Estos bits son referenciados como un conmutador saltar/no saltar que dirige al procesador a tomar una determinada decisión la próxima vez que encuentre la instrucción.
La utilización de bits de historia tiene un inconveniente: si la decisión que se toma es efectuar el salto, la instrucción destino no puede captarse hasta que su dirección, que es un operando de la instrucción de bifurcación condicional, sea decodificada. Se podría lograr una mayor eficiencia si la instrucción captada pudiera iniciarse tan pronto como se tome la decisión de salto. Para ello se debe guardar más información en la *tabla de historia de saltos*.
La tabla de historia de saltos es una pequeña caché asociada con la etapa de captación de instrucción del cauce. Cada elemento de la tabla consta de tres campos: la dirección de una instrucción de bifurcación, un número de bits de historia, e información sobre la instrucción destino. La tabla es tratada como una cache. Cada precaptación dispara una búsqueda en la tabla. Si no se encuentra ninguna coincidencia, se usa la dirección siguiente para la captación. Si hay coincidencia, se hace una predicción basada en el estado de la instrucción. Cuando se ejecuta una instrucción de bifurcación, se actualiza el estado de la instrucción en la tabla de historia de saltos.

•Salto retardado: se pueden mejorar las prestaciones de un cause reordenando automáticamente las instrucciones de un programa, de forma que las instrucciones de salto tengan lugar después de lo realmente deseado.

RISC

Los estudios del comportamiento de la ejecución de los programas escritos en lenguajes de alto nivel, proporcionaron la orientación para diseñar un nuevo tipo de arquitectura del procesador: el computador de repertorio reducido de instrucciones (reduced instruction set computer, RISC). Las características fundamentales son:
1) un repertorio de instrucciones limitado y sencillo.
2) un número grande de registros de uso general, o la utilización de un compilador que optimice el uso de éstos: su finalidad es mejorar las prestaciones reduciendo las referencias a memoria para buscar operandos.
3) un énfasis en la optimización de la segmentación de instrucciones.

El repertorio de instrucciones sencillo de un RISC se presta a una segmentación eficiente, porque hay menos operaciones llevadas a cabo por instrucción, y éstas son más previsibles. Una arquitectura de repertorio de instrucciones RISC también se presta a la técnica de salto retardado, en la cual las instrucciones de salto se reubican entre otras instrucciones para mejorar la eficiencia del cauce.

Utilización de un amplio banco de registros

Los registros constituyen el dispositivo de almacenamiento más rápido disponible; más que la memoria principal y que la caché. El banco de registros es pequeño físicamente, está en el mismo chip que la ALU y la unidad de control, y emplea direcciones mucho más cortas que las de la cache y la memoria. Por tanto, se necesita una estrategia que permita que los operandos a los que se accede con mayor frecuencia se encuentren en registros, y que se minimicen las operaciones registro-memoria.

Ventanas de registros

Un procedimiento típico emplea sólo unos pocos parámetros y variables, por lo que se usa un conjunto pequeño de registros, llamado ventana, que se direcciona como si fuera el único conjunto de registros. La ventana se divide en tres áreas de tamaño fijo: registros de parámetros, registros de datos locales y registros temporales.
El banco de registros organizado en ventanas funciona como un buffer pequeño y rápido, que contiene un subconjunto de todas las variables que probablemente se usen mucho. Desde este punto de vista, el banco de registros de comporta de manera muy similar a una memoria cache. Surge por lo tanto la cuestión de si sería más sencillo y mejor, usar una cache y un tradicional banco de registros pequeño.

La aproximación de los registros es claramente superior debido a la sobrecarga de direccionamiento. Para referenciar un dato escalar local en un banco de registros basado en ventanas, se usa un número de registro “virtual” y un número de ventana. Los dos pueden proporcionarse a un decodificador relativamente sencillo para seleccionar uno de los registros físicos. Para referenciar una posición de memoria en la cache, hay que generar una dirección de memoria completa. La complejidad de esta operación depende del modo de direccionamiento. En una cache asociativa por conjuntos, una parte de la dirección se usa para leer un número de palabras y etiquetas igual al tamaño del conjunto. Otra parte de la dirección se compara con las etiquetas, y se selecciona una de las palabras leídas. Incluso en el caso de que la cache sea tan rápida como el banco de registros, el tiempo de acceso será considerablemente más grande. Por consiguiente, desde el punto de vista de las prestaciones, el banco de registros basado en ventanas es superior para datos escalares locales.

CISC

Computador de repertorio complejo de instrucciones. La tendencia CISC ha sido motivada por dos razones:
• el deseo de simplificar los compiladores: la labor del escritor de compiladores es generar una secuencia de instrucciones máquina para cada sentencia de lenguaje de alto nivel (HLL). Si existen instrucciones máquina que se parezcan a sentencias del HLL, la tarea se simplifica. Sin embargo las instrucciones máquina complejas son con frecuencia difíciles de aprovechar.
• el deseo de mejorar las prestaciones: se espera que un CISC produzca programas más pequeños y rápidos

Los programas pequeños tienen dos ventajas: •ocupa menos memoria, así que se ahorra este recurso, pero la memoria hoy en día es tan barata que esa ventaja no es primordial. •hay que captar menos bytes de instrucciones.

Sin embargo, un programa para CISC expresado en lenguaje máquina puede ser más corto, pero el número de bits de memoria que ocupa no tiene por qué ser más pequeño.
Para que un programa sea más rápido, la unidad de control completa debe hacerse más completa, y/o la memoria de control del microprograma ha de hacerse más grande, para proveer un repertorio de instrucciones más rico. Cualquiera de los dos factores aumenta el tiempo de ejecución de las instrucciones simples.

Por lo tanto, no está nada claro que la tendencia hacia repertorios de instrucciones de complejidad creciente sea apropiada.

Características de las arquitecturas RISC

•Una instrucción por ciclo: un ciclo máquina se define como el tiempo que se tarda en captar dos operandos desde dos registros, realizar una operación en la ALU y almacenar el resultado en un registro. Así, las instrucciones máquina de un RISC no deberían ser más complicadas que las microinstrucciones de las máquinas CISC, y deberían ejecutarse más o menos igual de rápido.
•Operaciones registro a registro: excepto LOAD y STORE que acceden a memoria, esta forma de diseño simplifica el repertorio de instrucciones y, por tanto, la unidad de control. RISC incluye una o dos operaciones ADD (suma entera y suma con acarreo) mientras que el VAX tiene 25 instrucciones ADD.
•Modos de direccionamiento sencillos: casi todas las instrucciones RISC usan direccionamiento a registro, simplificando el repertorio de instrucciones y la unidad de control.
•Formatos de instrucción sencillos: generalmente se utiliza uno o unos pocos formatos. La longitud de las instrucciones es fija. Esto simplifica la unidad de control y se optimiza la captación de instrucciones, ya que se captan unidades de una palabra de longitud.

Características CISC frente a RISC

Hay una convicción de que los diseños RISC pueden sacar provecho de la inclusión de algunas características CISC y los diseños CISC pueden sacar provecho de la inclusión de algunas características RISC. El resultado es que los diseños RISC más recientes, especialmente el PowerPC, no son ya RISC “puros”, y que los diseños CISC más recientes, particularmente el Pentium II, incorporan ciertas características RISC.

Bus

Un bus es un camino de comunicación entre dos o más dispositivos. Es un conjunto de conductores eléctricos paralelos con líneas de metal. Una característica clave de un bus es que se trata de un medio de transmisión compartido. Al bus se conectan varios dispositivos, y cualquier señal transmitida por uno de esos dispositivos esta disponible para que los otros dispositivos conectados al bus puedan acceder a ella. Si dos dispositivos transmiten durante el mismo período de tiempo, sus señales pueden solaparse y distorsionarse, por lo que sólo un dispositivo puede transmitir con éxito en un momento dado.
Usualmente, un bus esta constituido por varios caminos de comunicación o líneas. Cada línea es capaz de transmitir señales binarias representadas por 1 y por 0. en un intervalo de tiempo, se puede transmitir una secuencia de dígitos binarios a través de una única línea. Se puede utilizar varias líneas del bus para transmitir dígitos binarios simultáneamente (en paralelo). Por ej, un dato de 8 bits puede transmitirse mediante 8 líneas del bus.
Los computadores poseen distintos tipos de buses que proporcionan comunicación entre sus componentes a distintos niveles dentro de la jerarquía del sistema. El bus que conecta los componentes principales del computador (procesador, memoria y E/S) se denomina bus del sistema. Las estructuras de interconexión más comunes dentro de un computador están basadas en el uso de uno o más buses del sistema.

Estructura del bus

El bus está constituido usualmente por entre 50 y 100 líneas. Hay 3 grupos de líneas

*Líneas de datos: proporcionan un camino para transmitir datos entre los módulos del sistema. Conforman el bus de datos. El número de líneas determina cuántos bits se pueden transferir al mismo tiempo. La anchura del bus de datos es un factor clave a la hora de determinar las

prestaciones del sistema. Por ejemplo, si el bus de datos tiene una anchura de 8 bits, y las instrucciones son de 16, entonces el procesador debe acceder al modulo de memoria dos veces para cada ciclo de instrucción.

*Líneas de dirección: se utilizan para designar la fuente o el destino del dato situado en el bus de datos. La anchura del bus de dirección determina la máxima capacidad de memoria posible en el sistema. También se usan para direccionar los puertos de E/S.
*Líneas de control: se utilizan para controlar el acceso y el uso de las lineas de datos y direcciones. Puesto que las lineas de datos y direcciones son compartidas por todos los componentes, debe existir una forma de controlar su uso. Las señales de control transmiten tanto ordenes como información de temporización entre los módulos del sistema.

Además pueden existir líneas de alimentación para suministrar energía a los módulos conectados al bus.
Si un modulo desea enviar un dato a otro debe hacer dos cosas: (1) obtener el uso de bus, y (2) transferir el dato a través del bus. Si un modulo desea pedir un dato a otro, debe (1) obtener el uso del bus, y (2) transferir la petición al otro modulo mediante las lineas de control y dirección apropiadas. Después debe esperar a que el segundo modulo envíe el dato.

Jerarquía de buses

Si se conecta un gran numero de dispositivos al bus, las prestaciones pueden disminuir, ya que aumenta el retardo de propagación y el bus puede convertirse en un cuello de botella a medida que hay peticiones acumuladas. Por consiguiente, la mayoría de las computadoras utilizan varios buses, normalmente organizados jerárquicamente.
Hay un bus local que conecta el procesador a una memoria cache. El controlador de memoria cache conecta la cache no sólo al bus local sino también al bus del sistema, donde se conectan todos los módulos de memoria principal. Es posible conectar controladores de E/S directamente al bus del sistema, pero es más eficiente utilizar uno o más buses de expansión.

Elementos en el diseño de un bus

Tipos de buses: Las lineas del bus se pueden dividir en dos tipos: dedicadas y multiplexadas. Una linea dedicada esta permanentemente asignada a una función o a un subconjunto físico de componentes del computador. Ej.: lineas separadas para direcciones y datos. Sin embargo, la información de dirección y datos podría transmitirse a través del mismo conjunto de lineas si se utiliza una linea de control. A este método se lo llama multiplexado en el tiempo. La ventaja es que se ahorra espacio y costes. La desventaja es que se necesita una circuitería más compleja.

Método de arbitraje

Puesto que, en un instante dado, sólo una unidad puede transmitir a través del bus, se requiere algún método de arbitraje. Los diversos métodos se pueden clasificar aproximadamente como centralizados o distribuidos. En un *esquema centralizado*, un único dispositivo hardware, denominado controlador del bus o árbitro, es responsable de asignar tiempos en el bus. El dispositivo puede estar en un módulo separado o ser parte del procesador. En un *esquema distribuido*, no existe un controlador central. En su lugar, cada módulo dispone de lógica para controlar el acceso, y los módulos actúan conjuntamente para compartir el bus. En ambos métodos de arbitraje, el propósito es designar un dispositivo, el procesador o un módulo de E/S como maestro del bus. El maestro podría entonces iniciar una transferencia de datos (lectura o escritura) con otro dispositivo, que actúa como esclavo en este intercambio concreto.

Temporización

El término temporización hace referencia a la forma en la que se coordinan los eventos en el bus. Con temporización síncrona, la presencia de un evento en el bus está determinada por un reloj. El bus incluye una línea de reloj a través de la que se transmite una secuencia en la que se alternan intervalos regulares de igual duración a uno y a cero. Un único intervalo a uno seguido de otro a cero se conoce como ciclo de reloj o ciclo de bus y define un intervalo de tiempo unidad ("time slot"). Todos los dispositivos del bus pueden leer la línea de reloj y todos los eventos empiezan al principio del ciclo de reloj.

Con la temporización asíncrona, la presencia de un evento en el bus es consecuencia y depende de que se produzca un evento previo. El procesador sitúa las señales de dirección y lectura en el bus. Después de un breve intervalo para que las señales se estabilicen, activa la señal MSYN (master sync), indicando la presencia de señales de dirección y control válidas. El módulo de memoria responde proporcionando el dato y una señal SSYN (slave sync)

La temporización síncrona es más fácil de implementar y comprobar. Sin embargo es menos flexible que la asíncrona, ya que todos los dispositivos deben utilizar la misma frecuencia de reloj. Con la temporización asíncrona pueden compartir el bus una mezcla de dispositivos lentos y rápidos.

Bus PCI

El bus PCI (Peripheral Component Interconnect, interconexión de componente periférico) es un bus muy popular, de ancho de banda elevado, independiente del procesador, que se puede utilizar como bus de periféricos o bus para una arquitectura de entreplanta. Comparado con otras especificaciones comunes de bus, el PCI proporciona mejores prestaciones para los subsistemas de E/S de alta velocidad (por ej. adaptadores de pantalla gráfica, controladores de disco, de interfaz de red, etc) el PCI ha sido diseñado específicamente para ajustarse económicamente a los requisitos de E/S de los sistemas actuales; se implementa con muy pocos circuitos integrados, y permite que otros buses se conecten al bus PCI. Utiliza temporización sincrónica y un esquema de arbitraje centralizado. Puede utilizarse en sistemas de uno o varios procesadores.

El bus SCSI sólo se utiliza para dispositivos de almacenamiento y debe tener un controlador de interfaz.

Memoria Cache

El objetivo de la memoria cache es lograr que la velocidad de la memoria sea lo más rápida posible, consiguiendo al mismo tiempo un tamaño grande al precio de memorias semiconductoras menos costosas. Hay una memoria principal relativamente grande y más lenta, junto con una memoria cache más pequeña y rápida. La cache contiene una copia de partes de la memoria principal. Cuando el procesador intenta leer una palabra de memoria, se hace una comprobación para determinar si la palabra está en la cache. Si es así, se entrega dicha palabra al procesador. Si no, un bloque de memoria principal, consistente en un cierto número de palabras, se transfiere a la cache y, después, la palabra es entregada al procesador. Debido al fenómeno de localidad de las referencias, cuando un bloque de datos es captado por la cache para satisfacer una referencia a memoria simple, es probable que se hagan referencias futuras a otras palabras del mismo bloque. La cache se conecta con el procesador mediante líneas de datos, de control y de direcciones. Las líneas de datos y de direcciones conectan también con buffers de datos y de direcciones que las comunican con un bus del sistema, a través del cual se accede a la memoria principal. Cuando ocurre un acierto de cache,

los buffers de datos y de direcciones se inhabilitan, y la comunicación tiene lugar sólo entre el procesador y la cache, sin tráfico en el bus, cuando ocurre un fallo de cache, la dirección deseada se carga en el bus del sistema, y el dato es llevado, a través del bus de datos, tanto a la cache como al procesador. En otras formas de organización, frente a un fallo de cache, la palabra es primero leída por la cache y luego transferida desde ésta al procesador.

Elementos de diseño de la cache

•Tamaño de la cache: hay muchas motivaciones para minimizar el tamaño de la cache: cuanto más grande es, mayor es el número de puertas implicadas en direccionar la cache, el resultado es que caches grandes tienden a ser ligeramente más lentas que las pequeñas (incluso estando fabricadas con la misma tecnología de circuito integrado y con la misma ubicación en el chip o en la tarjeta de circuito impreso). El tamaño de cache está también limitado por las superficies disponibles de chip y de tarjeta. Los tamaños óptimos se encuentran entre 1K y 512K palabras, pero depende del tipo de tarea para la que se utilicen.

•Función de correspondencia: se necesita algún algoritmo que haga corresponder bloques de memoria principal a líneas de cache y algún medio para determinar qué bloque de memoria principal ocupa actualmente una línea dada de cache. Pueden usarse 3 técnicas:

*Correspondencia directa: consiste en hacer corresponder cada bloque de memoria principal a sólo una línea posible de cache. Cada dirección de memoria principal se divide en etiqueta, línea y palabra. En una lectura, el número de línea se utiliza como índice para acceder a una línea de la cache, si la etiqueta coincide con el número de etiqueta almacenado en esa línea, se usa el número de palabra se utiliza para seleccionar un byte de esa línea Esta técnica es simple y poco costosa. La desventaja es que hay una posición concreta de cache para cada bloque dado, por lo que si un programa referencia repetidamente a dos bloques diferentes asignados en la misma línea, los bloques se estarían intercambiando continuamente en la cache, y la tasa de aciertos sería baja.
*Correspondencia asociativa: cualquier bloque puede ser reemplazado cuando se va a escribir uno nuevo en la cache. La principal desventaja es la compleja circuitería necesaria para examinar en paralelo las etiquetas de todas las líneas de cache.
*Correspondencia asociativa por conjuntos: es una solución que recoge lo positivo de las correspondencias directa y asociativa sin sus desventajas. La cache se divide en v conjuntos, cada uno de k líneas, un bloque b puede asignarse en cualquiera de k líneas del conjunto. Con la correspondencia totalmente asociativa, la etiqueta en una dirección de memoria es bastante larga, y debe compararse con la etiqueta de cada línea en la cache. Con la correspondencia asociativa por conjuntos de k vías, la etiqueta de una dirección de memoria es mucho más corta, y se comprara sólo con las k etiquetas dentro de un mismo conjunto.

•Algoritmos de sustitución: cuando se introduce un nuevo bloque en la cache, debe sustituirse uno de los bloques existentes. Para el caso de correspondencia directa, sólo hay una posible línea para cada bloque particular y no hay elección posible. Para las técnicas asociativas, se requieren algoritmos de sustitución. Para conseguir alta velocidad, tales algoritmos deben implementarse en hardware. Los 4 más comunes son:

*LRU (least-recently used, utilizado menos recientemente): se sustituye el bloque que se ha mantenido en la cache por más tiempo sin haber sido referenciado. Esto es fácil de implementar para la asociativa por conjuntos de dos vías. Cada línea incluye un bit USO. Cuando una línea es referenciada, se pone a 1 su bit USO y a 0 el de la otra línea del mismo conjunto. Cuando va a transferirse un bloque al conjunto, se utiliza la línea cuyo bit USO es 0. Ya que estamos suponiendo que son más probables de referenciar las posiciones de memoria utilizadas más recientemente, el LRU debería dar la mejor tasa de aciertos.

*FIFO (first-first-out): se sustituye aquel bloque del conjunto que ha estado más tiempo en la cache.
*LFU (least-frecuenty used): se sustituye aquel bloque del conjunto que ha experimentado menos referencias.
*Una técnnica no basada en el grado de utilización consiste simplemente en escoger una línea al azar. Proporciona unas prestaciones sólo ligeramente inferiores a un algoritmo basado en la utilización.

•Política de escritura: antes de que pueda ser reemplazado un bloque que está en una línea de cache, es necesario comprobar si ha sido alterado en cache pero no en memoria principal. Si una palabra ha sido modificada sólo en la cache, la correspondiente palabra de memoria no es válida, y si se ha alterado la memoria principal, entonces la palabra de cache no es válida.
La técnica más sencilla se denomina *escritura inmediata*, que hace todas las operaciones de escritura asegurando que el contenido de la memoria principal siempre sea válido. Un módulo procesador-cache monitorea el tráfico a memoria principal para mantener la coherencia en su propia cache. La principal desventaja es que genera un tráfico sustancial a memoria que puede originar un cuello de botella. Una técnica alternativa, conocida como *post-escritura*, minimiza las escrituras en memoria, ya que las actualizaciones se hacen sólo en la cache. Cuando tiene lugar una actualización, se activa un bit ACTUALIZAR asociado a la línea, y cuando este bloque debe ser sustituido, se escribe en memoria sólo si el bit ACTUALIZAR está activo. La desventaja es una circuitería compleja y cuello de botella potencial.

En una estructura de bus en la que más de un dispositivo (normalmente procesador) tiene una cache, y la memoria principal es compartida, cuando se modifican los datos de una cache se invalida no solamente la palabra correspondiente en memoria principal sino también la de las demás caches. Un sistema que evite este problema, se dice que mantiene la coherencia de cache. Existen 3 aproximaciones:

•Vigilancia del bus con escritura inmediata: cada controlador de cache monitoriza las líneas de direcciones para detectar operaciones de escritura en memoria por parte de otros maestros del bus.
•Transparencia de hardware: se utiliza hardware adicional para asegurar que todas las actualizaciones de memoria principal vía cache queden reflejadas en todas las caches.
•Memoria excluida de cache: sólo una porción de memoria principal se comparte por más de un procesador, y éste se diseña como no transferible a cache.

•Tamaño de línea: cuando se recupera y ubica en cache un bloque de datos, se recuperan además de la palabra deseada, algunas palabras adyacentes. A medida que aumenta el tamaño de bloque, la tasa de aciertos primero aumenta debido al principio de localidad. Sin embargo, la tasa de aciertos comenzará a decrecer cuando el tamaño de bloque se haga aún mayor y la probabilidad de utilizar la nueva información captada se haga menos que la de reutilizar la información que tiene que reemplazarse.

•Número de caches: cuando se introdujeron las caches, un sistema tenía normalmente sólo una cache. Más recientemente, se ha convertido en norma el uso de múltiples caches. Hay dos aspectos de diseño relacionados con este tema: el número de niveles de cache y el uso de cache unificada frente a cache partida.
Con el aumento de densidad de integración, ha sido posible tener una cache en el mismo chip del procesador (on-chip), que reduce la actividad del bus externo del procesador y los tiempos de ejecución, e incrementa las prestaciones globales del sistema. Los diseños actuales incluyen tanto cache on-chip como externa. La estructura resultante se conoce como cache de dos niveles, siendo la cache interna el nivel 1 (L1) y la externa el nivel 2 (L2). La razón por la que se incluye una cache L2 es la siguiente: si no hay cache L2 y el procesador hace una petición de acceso a una posición de memoria que no está en la cache L1, entonces el procesador debe acceder a DRAM o

ROM a través del bus, lo que se hace muy lento. Pero si se utiliza una cache L2 SRAM, entonces, con frecuencia, a información que falta puede recuperarse fácilmente.
La mejora potencial del uso de una cache L2 depende de las tasas de aciertos en ambas caches L1 y L2. Estudios demostraron que el uso de un segundo nivel mejora las prestaciones.
Cuando hicieron su aparición las caches on-chip, muchos de los diseños contenían una sola cache para almacenar las referencias, tanto a datos como a instrucciones. Más recientemente se ha hecho normal separar la cache en dos: una dedicada a instrucciones y otra a datos. Una caché unificada tiene varias ventajas: sólo se necesita diseñar e implementar una caché, y esta suele tener una tasa de aciertos mayor, ya que nivela automáticamente la carga entre captación de instrucciones y de datos. A pesar de estas ventajas, la tendencia es hacia caches partidas, particularmente para máquinas superescalares, tales como el Pentium II y el PowerPC, en las que se enfatiza la ejecución paralela de instrucciones y la pre-captación. La ventaja clave de la cache partida es que elimina la competición por la cache entre el procesador de instrucciones (que precapta instrucciones y las escribe en la cache) y la unidad de ejecución (que carga y almacena datos en la cache), ya que pueden querer utilizar la caché al mismo tiempo. Esto es importante en diseños que cuentan con segmentación de cauce de instrucciones.

Procesadores superescalares

Un procesador superescalar es aquel que usa múltiples cauces de instrucciones independientes. Cada cause costa de múltiples etapas, de modo que puede tratar varias instrucciones a la vez. El hecho de que haya varios cauces introduce un nuevo nivel de paralelismo, permitiendo que varios flujos de instrucciones se procesen simultáneamente. Un procesador superescalar saca provecho de lo que se conoce como “paralelismo a nivel de instrucciones”, que hace referencia al grado en que las instrucciones de un programa pueden ejecutarse en paralelo.
Típicamente, un procesador superescalar capta varias instrucciones a la vez y, a continuación, intenta encontrar instrucciones cercanas que sean independientes entre sí y puedan ejecutarse en paralelo. Si la entrada de una instrucción depende de la salida de una instrucción precedente, la segunda instrucción no puede completar su ejecución al mismo tiempo ni antes que la primera. Una vez que se han identificado tales dependencias, el procesador puede emitir y completar instrucciones en un orden diferente al del código máquina original.
Mientras que los procesadores RISC puros emplean con frecuencia saltos retardados para maximizar la utilización del cauce de instrucciones, este método es menos apropiado para las máquinas superescalares. En lugar de eso, se emplean métodos tradicionales de predicción de saltos para aumentar su rendimiento. La aproximación superescalar se puede usar tanto en RISC como en CISC.

Superescalar frente a supersegmentado

Una solución alternativa para alcanzar mayores prestaciones es la llamada supersegmentación. La supersegmentación aprovecha el hecho de que muchas etapas del cauce realizan tareas que requieren menos de la mitad de un ciclo de reloj. De este modo, se dobla la velocidad de reloj interna, lo que permite la realización de dos tareas en un ciclo de reloj externo. El procesador supersegmentado se queda atrás con respecto al procesador superescalar al comienzo del programa y en cada destino de un salto.

Limitaciones

La aproximación superescalar depende de la habilidad para ejecutar múltiples instrucciones en paralelo. La expresión *paralelismo a nivel de instrucciones* se refiere al grado en el que, en promedio, las instrucciones de un programa se pueden ejecutar en paralelo. Para maximizar el paralelismo a nivel de instrucciones, se puede usar una combinación de optimizaciones realizadas

por el compilador y de técnicas hardware. Las limitaciones fundamentales del paralelismo a las que el sistema tienen que enfrentarse son cinco:

• Dependencia de datos verdadera: sucede cuando una instrucción necesita un dato producido por una instrucción anterior, por lo que no puede ejecutarse y se atrasa tantos ciclos de reloj como sea necesario hasta que se elimine la dependencia. Ej: add r1, r2 move r3, r1.
• Dependencias relativas al procedimiento: la presencia de bifurcaciones en una secuencia de instrucciones complica el funcionamiento del cauce. Las instrucciones que siguen a una bifurcación (en la que se puede saltar o no) tienen una dependencia relativa al procedimiento en esa bifurcación, y no pueden ejecutarse hasta que ésta lo haga.
• Conflictos en los recursos: es una pugna de dos o más instrucciones por el mismo recurso al mismo tiempo. Por ej: memorias, caches, buses, puertos del banco de registros y unidades funcionales (por ej un sumador de la ALU). Se puede solucionar duplicando recursos.
• Dependencia de salida: se da cuando una instrucción sobreescribe un resultado de una instrucción anterior que tarda más en finalizar.
• Antidependencia: inversa a la dependencia verdadera, en lugar de que la primera instrucción produzca un valor que usa la segunda instrucción, la segunda instrucción destruye un valor que usa la primera instrucción.

El paralelismo a nivel de instrucciones depende de la frecuencia de dependencias de datos verdaderas y dependencias relativas al procedimiento que haya en el código. Estos factores dependen a su vez de la arquitectura del repertorio de instrucciones de la aplicación.
El paralelismo de máquina es una medida de la capacidad del procesador para sacar partido al paralelismo a nivel de instrucciones. Depende del número de instrucciones que pueden captarse y ejecutarse al mismo tiempo (número de cauces paralelos), y de la velocidad y sofisticación del mecanismo que usa el procesador para localizar instrucciones independientes.

Políticas de emisión de instrucciones

El procesador intenta localizar instrucciones más allá del punto de ejecución en curso, que puedan introducirse en el cause y ejecutarse. Son importantes tres ordenaciones:

• El orden en que se captan las instrucciones
• El orden en que se ejecutan las instrucciones
• El orden en que las instrucciones actualizan los contenidos de los registros y de las posiciones de memoria.

La única restricción que tiene el procesador es que el resultado debe ser correcto. De este modo, el procesador debe acomodar las diversas dependencias y conflictos. Podemos agrupar las políticas de emisión de instrucciones de los procesadores superescalares en:

• Emisión en orden y finalización en orden: emitir instrucciones en el orden exacto en que lo haría una ejecución secuencial (emisión en orden) y escribir los resultados en ese mismo orden (finalización en orden).
• Emisión en orden y finalización desordenada: se usa en los procesadores RISC escalares para mejorar la velocidad de las instrucciones que necesitan muchos ciclos. Con la finalización desordenada puede haber cualquier número de instrucciones en la etapa de ejecución en un momento dado, hasta alcanzar el máximo grado de paralelismo de la máquina, ocupando todas las unidades funcionales.
• Emisión desordenada y finalización desordenada: con la emisión en orden, el procesador sólo

decodificará instrucciones hasta el punto de dependencia o conflicto. No se decodifican más instrucciones hasta que el conflicto se resuelva. Por consiguiente, el procesador no puede buscar más allá del punto de conflicto. Para permitir la emisión desordenada, es necesario desacoplar las etapas del cause de decodificación y ejecución mediante un buffer llamado ventana de instrucciones. Cuando un procesador termina de decodificar una instrucción, coloca ésta en la ventana de instrucciones, mientras el buffer no se llene. Cuando una unidad funcional de la etapa de ejecución queda disponible, se puede emitir una instrucción desde la ventana de instrucciones a la etapa de ejecución. Así, el procesador tiene capacidad de anticipación, lo que le permite identificar instrucciones independientes que pueden introducirse en la etapa de ejecución. Las instrucciones se emiten desde la ventana de instrucciones sin que se tenga muy en cuenta su orden original en el programa.

Renombramiento de registros

Las dependencias de salida y antidependencias surgen porque los valores de los registros no pueden reflejar la secuencia de valores dictada por el flujo del programa, lo que puede detener alguna etapa del cauce. En la técnica de renombramiento de registros, el hardware del procesador asigna dinámicamente los registros, que están asociados con los valores que necesitan las instrucciones en diversos instantes de tiempo. Cuando se crea un nuevo valor de registro (es decir, cuando se ejecuta una instrucción que tiene un registro como operando destino), se asigna un nuevo registro para ese valor. Las instrucciones posteriores que accedan a ese valor como operando fuente en ese registro, tienen que sufrir un proceso de renombramiento: las referencias a registros de esas instrucciones han de revisarse para referenciar el registro que contiene el valor que se necesita. De este modo, las referencias a un mismo registro original en diferentes instrucciones, pueden referirse a distintos registros reales, suponiendo diferentes valores.

Implementación superescalar

El procesador en la aproximación superescalar requiere hardware con ciertos elementos:

• Estrategias de captación de instrucciones que capten simultáneamente múltiples instrucciones, a menudo prediciendo los resultados de las instrucciones de bifurcación condicional y captando más allá de ellas. Estas funciones requieren la utilización de múltiples etapas de captación y decodificación, y lógica de predicción de saltos.
• Lógica para determinar dependencias verdaderas entre valores de registros, y mecanismos para comunicar esos valores a donde sean necesarios durante la ejecución.
• Mecanismos para iniciar, o emitir, múltiples instrucciones en paralelo.
• Recursos para la ejecución en paralelo de múltiples instrucciones, que incluyan múltiples unidades funcionales segmentadas y jerarquías de memoria capaces de atender múltiples referencias a memoria.
• Mecanismos para entregar el estado del procesador en un orden correcto.

Aunque el concepto de diseño superescalar se asocia generalmente a RISC, también puede aplicarse en una máquina CISC, por ej. el Pentium II.

Procesamiento paralelo (aproximaciones al uso de varios procesadores)

Una manera tradicional de incrementar las prestaciones de un sistema consiste en utilizar varios procesadores que puedan ejecutar en paralelo una carga de trabajo dada. Las organizaciones de

múltiples procesadores más comunes son los multiprocesadores simétricos (SMP) y los “clusters”. Y más recientemente los sistemas de acceso a memoria no uniforme (NUMA).

Un SMP es un computador constituido por varios procesadores similares, interconectados mediante un bus o algún tipo de estructura de conmutación. Su problema más crítico es la coherencia de cache. Cada procesador tiene su propia cache, y es posible que una línea de datos dada esté presente en más de una cache. Si esta línea se altera en una cache, entonces tanto la memoria principal como las otras caches tienen versiones no válidas de dicha línea.

Un “cluster” es un grupo de computadores completos interconectados y trabajando juntos como un sólo recurso de cómputo, proporcionando la ilusión de ser una única máquina. El término *computador completo* significa que puede funcionar autónomamente fuera del cluster.

Un sistema NUMA es un multiprocesador de memoria compartida, en el que el tiempo de acceso de un procesador a una palabra de memoria varía según la ubicación de la palabra en memoria.

Los sistemas SMP y NUMA utilizan memoria compartida (mediante un bus compartido u otro mecanismo de interconexión), en cambio los clusters utilizan memoria distribuida.

Tipos de sistemas de paralelos (categorías de computadores)

• Una secuencia de instrucciones y una secuencia de datos (SISD, Single Instruction, Single Data): un único procesador interpreta una única secuencia de instrucciones, para operar con datos almacenados en una única memoria. Ej: computadores monoprocesador.
• Una secuencia de instrucciones y múltiples secuencias de datos (SIMD): una única instrucción máquina controla paso a paso la ejecución simultánea y sincronizada de un cierto número de elementos de proceso. Cada elemento de proceso tiene una memoria asociada, de forma que cada instrucción es ejecutada por cada procesador, con un conjunto de datos diferentes. Ej: procesadores vectoriales y matriciales.
• Múltiples secuencias de instrucciones y una secuencia de datos (MISD): se transmite una secuencia de datos a un conjunto de procesadores, cada uno de los cuales ejecuta una secuencia de instrucciones diferente. Esta estructura no ha sido implementada.
• Múltiples secuencias de instrucciones y múltiples secuencias de datos (MIMD): un conjunto de procesadores ejecuta simultáneamente secuencias de instrucciones diferentes con conjuntos de datos diferentes. Ej: SMP, clusters y NUMA.

Múltiples procesadores simétricos

El término SMP se refiere a la arquitectura hardware del computador, y también al comportamiento del sistema operativo que utiliza dicha arquitectura (el SO debe explotar el paralelismo). Un SMP tiene las siguientes características:

1) Hay dos o más procesadores similares de capacidades comparables.
2) Estos procesadores comparten la memoria principal y las E/S, y están interconectados mediante un bus u otro tipo de sistema de interconexión, de forma que el tiempo de acceso a memoria es aproximadamente el mismo para todos los procesadores.
3) Todos los procesadores comparten los dispositivos de E/S.
4) Todos los procesadores pueden desempeñar las mismas funciones (por eso son simétricos)
5) El sistema está controlado por un sistema operativo integrado, que proporciona la interacción entre los procesadores y sus programas.

Un SMP tiene las siguientes ventajas potenciales con respecto a una arquitectura monoprocesador:

1) Prestaciones: si el trabajo se puede dividir en partes y realizar en paralelo, un sistema con varios procesadores proporcionará mejores prestaciones.
2) Disponibilidad: un fallo del procesador no hará que el computador se detenga, ya que todos los procesadores pueden realizar las mismas tareas.
3) Crecimiento incremental: se pueden aumentar las prestaciones del sistema añadiendo más procesadores.
4) Escalado: los fabricantes pueden ofrecer una gama de productos con precios y prestaciones diferentes en función del número de procesadores.

La existencia de varios procesadores es transparente al usuario. El SO se encarga de su coordinación.

Bus de tiempo compartidos

El bus de tiempo compartido es el mecanismo más simple para construir un sistema multiprocesador. La estructura y las interfaces son básicamente las mismas que las de un sistema único procesador que utilice un bus para la interconexión. El bus consta de líneas de control, dirección y datos. Las ventajas son:

• Simplicidad: es la aproximación más simple para organizar el multiprocesador.
• Flexibilidad: es generalmente sencillo expandir el sistema conectando más procesadores al bus.
• Fiabilidad: el bus es esencialmente un medio pasivo, y el fallo de cualquiera de los dispositivos conectados no provocaría el fallo de todo el sistema.

Pero la principal desventaja son las prestaciones, ya que todas las referencias a memoria pasan por el bus. En consecuencia, la velocidad del sistema está limitada por el tiempo de ciclo. Para mejorar las prestaciones es deseable equipar a cada procesador con una memoria cache, que reduciría dramáticamente el número de accesos.

Clusters

Los clusters constituyen una alternativa a los multiprocesadores simétricos (SMP) para disponer de prestaciones y disponibilidad elevadas, y son particularmente atractivos en aplicaciones propias de un servidor. Un cluster es un grupo de computadores completos interconectados que se trata de una sola máquina. El término computador completo hace referencia a un sistema que puede funcionar por sí solo, independientemente del cluster. Usualmente, cada computador del cluster se denomina nodo.

Beneficios de la utilización de un cluster:

1) Escalabilidad absoluta: es posible configurar clusters grandes, que incluso superan las prestaciones de los computadores independientes más potentes.
2) Escalabilidad incremental: un usuario puede comenzar con un sistema modesto y ampliarlo a medida que lo necesite, sin tener que sustituir el sistema que dispone.
3) Alta disponibilidad: el fallo de uno de los nodos no significa la pérdida del servicio, ya que son autónomos.
4) Mejor relación precio/prestaciones: gracias a la utilización de elementos estandarizados.

Cluster frente a SMP

La principal ventaja de un SMP es que resulta más fácil de gestionar y configurar que un cluster. El SMP está mucho más cerca del modelo de computador de un sólo procesador, para el que están disponibles casi todas las aplicaciones. El principal cambio que se necesita para pasar de un computador monoprocesador a un SMP se refiere al funcionamiento del planificador. Otra ventaja de un SMP es que necesita menos espacio físico y consume menos energía. Además, los SMP son plataformas estables y bien establecidas. Los clusters son superiores a los SMP en términos de escalabilidad absoluta e incremental, además de disponibilidad.

Acceso no uniforme a memoria

• Acceso uniforme a memoria (UMA): todos los procesadores pueden acceder a toda la memoria principal utilizando instrucciones de carga y almacenamiento. El tiempo de acceso de un procesador a cualquier región es el mismo.

• Acceso no uniforme a memoria (NUMA): todos los procesadores tienen acceso a todas las partes de la memoria principal utilizando instrucciones de carga y almacenamiento. El tiempo de acceso a memoria de un procesador depende de la región a la que se acceda.

• NUMA con coherencia de cache (CC-NUMA): un computador NUMA en el que la coherencia de cache se mantiene en todas las caches de los distintos procesadores.

Motivación

el límite de procesadores en un SMP es uno de los motivos para el desarrollo de los clusters. Sin embargo, en un cluster cada nodo tiene su propia memoria principal privada, y las aplicaciones no “ven” la memoria global. El objetivo de un computador NUMA es mantener una memoria transparente desde cualquier parte del sistema, al tiempo que se permiten varios nodos de multiprocesador, cada uno con su propio bus u otro sistema de interconexión interna.

Organización CC-NUMA

Varios nodos independientes, cada uno de los cuales es un SMP. Cada nodo contiene varios procesadores, con sus caches L1 y L2, más memoria principal. Desde el punto de vista de los procesadores, existe un único espacio de memoria direccionable.