Arquitectura de computadoras follada

by Dante

Nota: Las respuestas están separadas por modulos

Preguntas de finales:

2015/04

1. ¿Qué es una pila? Describir el comportamiento con anidamiento de múltiples procedimientos/funciones utilizando pila.
2. Justifique el uso de dos niveles de caché.
3. ¿Cómo funciona un módulo de E/S? Describa las características fundamentales de un DMA.
4. ¿Qué es la segmentación de cauce? Explique los atascos producidos por saltos
5. Funcionamiento de un cluster

2014/12/03

1. Finalidad de las interrupciones. Para que se utiliza un controlador de interrupciones.
2. Describir la estructura de un módulo de E/S. ¿Qué es DMA y cómo funciona?
3. ¿Qué es la segmentación de cauce? Describir técnicas para el tratamiento de saltos condicionales.
4. Caché: mencione algoritmos de reemplazo y políticas de escritura.
5. Describa las características que diferencian a los procesadores RISC respecto de los CISC.

2014/11

1. Explique los métodos de pasaje de argumentos a procedimientos o funciones. Describa el funcionamiento y uso de la pila.
2. ¿Qué es una interrupción? Describa cómo funcionan. ¿Cómo se utiliza un controlador de interrupción?
3. ¿Por qué funciona una jerarquía de memoria? Describa las políticas de ubicación y de reemplazo de bloques en memoria caché.
4. ¿Qué ventajas nos brinda un cauce segmentado? Describa las diferentes formas que pueden mejorar el funcionamiento de un cauce cuando ejecuta instrucciones de transferencia de control.
5. ¿Qué características posee un multiprocesador simétrico (SMP)?

2014/10/22

1. ¿Que es un Bus? Describa los diferentes tipos, métodos de arbitraje y técnicas de sincronización. Mencione las principales diferencias entre PCI y SCSI.
2. ¿Cómo es la estructura de un módulo de E/S? Describa las posibles técnicas que puede utilizar una CPU para realizar operaciones de E/S.
3. Describa las técnicas de ubicación de bloques y las políticas de escritura en Caché.
4. ¿Que es la segmentación del cauce de instrucciones? Describa los métodos y técnicas para disminuir o evitar las paradas (stalls) que afectan el funcionamiento de los cauces.
5. Describa las características que diferencian los procesadores RISC de los CISC.

2014/09

1. En un cauce segmentado, con secuencia de instrucciones independientes ¿Qué consecuencias trae el paso de una instrucción de salto? Analice los casos de salto incondicional y condicional. Mencione que posibles soluciones se pueden aplicar para evitar o disminuir las consecuencias.
2. Describa cómo se debe implementar la estructura de pila en un procesador de tipo RISC cuyos registros son genéricos (basarse en MIPS) ¿Cómo se deberá trabajar el anidamiento de procesos / funciones?
3. Describa las funciones que se utilizan en la política de ubicación de bloques en memoria caché. Analice las políticas de escritura de datos desde el punto de vista de la coherencia de los mismos en la jerarquía.
4. ¿Qué características definen un procesador como superescalar? Describa las políticas de emisión de instrucciones en un cauce segmentado.
5. ¿Cuáles son las arquitecturas que pueden encontrarse en la configuración MIMD de la taxonomía de Flynn?

2014/08/06

1. Explique el mecanismo de interrupción. Describa las distintas fuentes de interrupción que conozca y el tratamiento a realizar cuando hay múltiples interrupciones.
2. ¿Cómo es la estructura interna de un módulo de E/S? Describa las características funcionales del acceso directo a memoria - DMA.
3. Describa las funciones de correspondencia entre memoria principal y caché. Analice las politicas de escritura desde el punto de vista de la coherencia de datos.
4. ¿Qué entiende por segmentación de cauce? ¿Qué ventajas proporciona su implementación?
5. Describa tres (3) diferentes causas o motivos que pueden retardar un cauce de instrucciones segmentado
6. ¿Qué características describen un cluster de computadoras?

2014/02/26

1. Mecanismo de interrupción. Fuentes de interrupción y tratamiento de interrupciones múltiples.
2. Estructura interna del módulo de E/S. Características funcionales de DMA
3. Funciones de correspondencia entre memoria y memoria caché. Políticas de escritura desde el punto de vista de la coherencia de datos.
4. Qué es la segmentación de cauce. Tres motivos de retardo de cauce.
5. Características de los clusters

2014/02

1. Que es un Bus, tipos de buses, temporización y métodos de arbitraje
2. Como es la estructura de un módulo de E/S. Describa las posibles técnicas que pueden utilizarse para realizar las operaciones de E/S
3. Describa las técnicas de reemplazo de bloque, correspondencia y políticas de escritura en memoria caché.
4. Que es la segmentación de cauce. Describa los métodos y técnicas utilizadas para disminuir o evitar las paradas que afectarán al funcionamiento de los cauces.
5. Describa las características que diferencian los SMTP respecto a los clusters

2013/09

1. Explique el mecanismo de interrupción. Describa las características y el funcionamiento de un pic.
2. Describa las características fundamentales de un DMA
3. Describa los diferentes algoritmos de reemplazo de ¿bloques? de una caché. Analice las políticas de escritura desde el punto de vista de la coherencia de datos.
4. ¿Qué entienden por segmentación de cauce? ¿Qué ventajas proporciona su implementación?

5. ¿Qué características describen un cluster de computadoras?

2013/03/26

1. Explique los métodos de pasaje de argumentos a procedimientos o funciones. Describa el comportamiento con anidamiento de múltiples procedimientos/funciones.
2. ¿Cómo es la estructura de un módulo de E/S? Describa las características del controlador de interrupciones PIC.
3. Describa los elementos a tener en cuenta en el diseño de una memoria caché. Anilice ventajas y desventajas de poseer varios niveles de caché.
4. ¿Qué es la segmentación del cauce de instrucción? ¿Cuánto mejora el rendimiento? Describa las dependencias de los datos que puedan afectar un cauce segmentado.
5. ¿Qué características posee un procesador superescalar?

2013/02/27

1. Describir el mecanismo de interrupción. Mencionar cuales son las fuentes de interrupción (tipos de interrupción). Describir el tratamiento de múltiples interrupciones.
2. ¿Cómo es la estructura de un módulo de E/S? (Diagrama). Describir la función del DMA.
3. Mencionar los tipos de correspondencia de la memoria caché. Describir las políticas de escritura (en acierto y en fallo).
4. ¿Qué es la segmentación de instrucciones? ¿Cómo mejora el rendimiento? Describir tipos de dependencia que afectan el funcionamiento de los cauces.
5. ¿Qué características posee un procesador superescalar?

2012/03 - 2do

1. Métodos de Pasaje de argumentos a las funciones/procedimientos. ¿Qué sucede cuando tenemos varias subrutinas anidadas?
2. Estructura de un modulo E/S. Describa el funcionamiento de un controlador DMA(las etapas de transferencia).
3. Describa qué se debe tener en cuenta para diseñar una caché(TODO).
4. Segmentacion de cauce. Describir atascos por dependencia de datos y su solución.
5. PROCESADORES SUPERESCALAR -CARACTERISTICAS.

2012/03 - 1ro

1. ¿Qué es un bus? Describa los tipos, arbitraje y técnicas de sincronización. Mencione diferencias entre bus PCI y bus SCSI.
2. ¿Cómo es la estructura de un módulo E/S? Describa las posibles técnicas que puede utilizar una CPU para realizar operaciones E/S.
3. Describa algoritmos de ubicación y política de escritura en caché.
4. ¿Qué es la segmentación del cauce? Describa tipos de dependencias que afectan el funcionamiento de los cauces y las posibles soluciones para evitarlos.
5. Describa las características que diferencian a los procesadores RISC respecto de los CISC.

Respuestas

Modulo 1

Temas básicos de repaso

Arquitectura Von Neumann

- La unidad central de procesamiento (CPU) está constituida por la unidad de control (UC) y la unidad aritmético-lógica (ALU).
- Datos e instrucciones deben introducirse en el sistema y los resultados se proporcionarán mediante componentes de entrada/salida (E/S).
- Se necesita almacenar temporalmente datos e instrucciones:
    - Memoria Principal

Repertorio de instrucciones

- Es el conjunto completo de instrucciones que se realizan en una CPU.
    - Código máquina
    - Binario
- Representado simbólicamente por un conjunto de códigos de ensamblaje
    - de operaciones:
        - ADD (sumar), SUB (restar), LOAD (cargar datos en un registro)
    - de operandos:
        - ADD BX, PEPE; sumar contenidos de reg BX y dirección PEPE, el resultado se guarda en reg BX

Elementos de una instrucción

- Código de operación (“Cod Op”)
- Referencia a operandos fuentes
- Referencia al operando resultado
- Referencia a la siguiente instrucción

Tipos de instrucciones

- Procesamiento de datos: instrucciones aritmético-lógicas
- Almacenamiento de datos: instrucciones de memoria
- Transferencia de datos: instrucciones de E/S
- Control: instrucciones de testeo y flujo del programa

Subrutinas

¿Qué son?

- Innovación en lenguajes de programación
- Programa auto-contenido
- Puede invocarse desde cualquier punto de un programa (instrucción CALL)
- Brinda economía (código reusable) y modularidad (subdivisión en unidades pequeñas)
- Requiere pasaje de argumentos (parámetros)
    - Por valor (copia de una variable)
    - Por referencia (dirección de la variable)

Segmentación de cauce

Tareas a realizar por un ciclo

- Búsqueda (F, Fetch)

    - Se accede a memoria por la instrucción
    - Se incrementa el PC
- Decodificación (D, Decode)
    - Se decodifica la instrucción, obteniendo operación a realizar en la ruta de datos
    - Se accede al banco de registros por el/los operando/s (si es necesario)
    - Se calcula el valor del operando inmediato con extensión de signo (si hace falta)
- Ejecución (X, Execute)
    - Se ejecuta la operación en la ALU
- Acceso a memoria (M, Memory Access)
    - Si se requiere un acceso a memoria, se accede
- Almacenamiento (W, Writeback)
    - Si se requiere volcar un resultado a un registro, se accede al banco de registros

Subrutinas - Pasaje de parámetros - Pila

Preguntas relacionadas:

1. ¿Qué es una pila? Describir el comportamiento con anidamiento de múltiples procedimientos/funciones utilizando pila.
2. Explique los métodos de pasaje de argumentos a procedimientos o funciones. Describa el funcionamiento y uso de la pila.
3. Explique los métodos de pasaje de argumentos a procedimientos o funciones. Describa el comportamiento con anidamiento de múltiples procedimientos/funciones.

Pasaje de argumentos a procedimientos o funciones

El pasaje de argumentos/parámetros a subrutinas/procedimientos/funciones puede ser de tres formas:

Vía registros:

- El número de registros es la principal limitación
- Es importante documentar qué registros se usan

Ventanas de registros: El uso de un conjunto amplio de registros debería reducir la necesidad de acceder a memoria. Un procedimiento típico emplea solo unos pocos parámetros de llamada y variables locales. Además, la profundidad de activación de procedimientos fluctúa dentro de un rango relativamente pequeño. Para explotar estas propiedades, se usan múltiples conjuntos de registros, cada uno asignado a un procedimiento distinto. Una llamada a un procedimiento hace que el procesador conmute automáticamente a una ventana de registros distinta de tamaño fijo, en lugar de salvaguardar los registros en memoria. Las ventanas de procedimientos adyacentes están parcialmente solapadas para permitir el paso de parámetros.

La ventana se divide en tres áreas de tamaño fijo. Los registros de parámetros contienen parámetros pasados al procedimiento en curso desde el procedimiento que lo llamó, y los resultados a devolver a éste. Los registros locales se usan para variables locales, según la asignación que realice el compilador. Los registros temporales se usan para intercambiar parámetros y resultados con el siguiente nivel más bajo (el procedimiento llamado por el procedimiento en curso). Los registros temporales de un nivel son físicamente los mismos que

los registros de parámetros del nivel más bajo adyacente. Este solapamiento posibilita que los parámetros se pasen sin que exista una transferencia de datos real.

Figura 13.1. Ventanas de registro solapadas.

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 cómo funciona porque la pila (stack) es usada por el usuario y por el sistema
- En x86, SP apunta al último lugar usado

¿Qué es una pila?

Una pila (*stack*) es un **conjunto ordenado de elementos**, en el que solo uno de ellos es accesible en un instante dado: la **cabecera** de la pila. Su **longitud es variable**. Solo se pueden añadir o eliminar elementos en la cabecera/tope de la pila (es conocida como lista LIFO).

¿Cómo funciona la pila?

Operaciones:

- **PUSH**: apila/añade un nuevo elemento en la cabecera de la pila.
- **POP**: desapila/elimina el elemento de la cabecera de la pila.

Es una secuencia de dos acciones:

1. Movimiento de datos Reg-Mem o Mem-Reg
2. Modificación del puntero antes/después de la anterior

Hay que tener en cuenta:

- Dónde apunta el puntero
- Cómo crece la pila

Las **operaciones unarias** operan con el elemento de la cabecera (el operando está de forma implícita) y lo sustituyen por el resultado (ej: NOT).

Las **operaciones binarias** operan con dos elementos de la cabecera de la pila, los elimina y pone el resultado de la operación en la cabecera.

¿Cómo se usa la pila?

En memoria principal se **reserva un bloque de posiciones contiguas** para la pila. Para su correcto funcionamiento se necesitan **3 direcciones**, normalmente **memorizadas en registros** de la CPU:

- **Puntero de pila (SP)**: Dirección de la cabecera de la pila. Si se añade o elimina un elemento de la pila, el mismo incrementa o decrementa para actualizarse.
- **Base de la pila (BP)**: Dirección base del bloque reservado para la pila. Si se hace un POP con la pila vacía, se informa un error.
- **Límite de la pila**: Dirección del otro extremo del bloque reservado. Si se hace un PUSH con la pila llena, se informa un error.

Anidamiento de múltiples funciones utilizando pila

El comportamiento de la pila con anidamiento de procedimientos/funciones/subrutinas:

Antes de llamar a la subrutina se debe apilar los parámetros a pasar y la dirección de retorno, al llamar a la subrutina de debe:

1. **Salvar el estado de BP** (viejo BP): apilar el valor del BP
2. **Salvar el estado de SP** (BP=SP): cargar el valor del BP el del SP
3. **Reservar espacio para datos locales** (opcional): si es que los hay
4. **Salvar valores de otros registros** (opcional): si es que se van a modificar los registros
5. **Acceder a parámetros**: para accederlos se le debe sumar un desplazamiento al BP para acceder a la posición de la pila en la que estén.
    1. **En general el desplazamiento es**: 2 (es el tamaño de BP apilado) + tamaño de dirección de retorno + total de tamaño de parámetros entre el buscado y BP
6. **Escribir sentencias a ejecutar**: Aquí se puede llamar a otra subrutina o regresar a la anterior. Si se vuelve a hacer una llamada se debe apilar los parámetros a pasar a la subrutina y la dirección de retorno. La subrutina llamada debe repetir los pasos anteriores mas los que siguen. Sino se hace otra llamada a subrutina también se deberán seguir los siguientes pasos:
7. **Retornar parámetro** (opcional): si es que tiene que retornar datos
8. **Regresar correctamente del procedimiento**: desapilar todo lo que apilo, volver a cargar a cargar el valor del BP que tenía antes de entrar a la subrutina

Siempre apilo las direcciones de retorno (tantas como llamadas anidadas haya) y voy sacando en orden inverso al que fueron guardadas (con POP, siguiendo la estructura LIFO).

Segmentación de cauce

- ¿Qué es la segmentación de cauce? Explique los atascos producidos por saltos.
- En un cauce segmentado, con secuencia de instrucciones independientes ¿Qué consecuencias trae el paso de una instrucción de salto? Analice los casos de salto incondicional y condicional. Mencione que posibles soluciones se pueden aplicar para evitar o disminuir las consecuencias.
- ¿Qué entiende por segmentación de cauce? ¿Qué ventajas proporciona su implementación?
- Segmentación de cauce. Describir atascos por dependencia de datos y su solución.
- ¿Qué es la segmentación de cauce? Describir técnicas para el tratamiento de saltos condicionales
- ¿Qué ventajas nos brinda un cauce segmentado? Describa las diferentes formas que pueden mejorar el funcionamiento de un cauce cuando ejecuta instrucciones de transferencia de control.
- ¿Que es la segmentación del cauce de instrucciones? Describa los métodos y técnicas para disminuir o evitar las paradas (stalls) que afectan el funcionamiento de los cauces
- Describa tres (3) diferentes causas o motivos que pueden retardar un cauce de instrucciones segmentado
- Qué es la segmentación de cauce. Tres motivos de retardo de cauce.
- ¿Qué es la segmentación del cauce de instrucción? ¿Cuánto mejora el rendimiento? Describa las dependencias de los datos que puedan afectar un cauce segmentado
- ¿Qué es la segmentación del cauce? Describa tipos de dependencias que afectan el funcionamiento de los cauces y las posibles soluciones para evitarlos.

¿Qué es la segmentación de cauce?

La segmentación de cauce (pipelining) es una forma de organizar el hardware de la CPU para realizar más operaciones al mismo tiempo: Supone dividir el ciclo de instrucción en varias etapas/fases separadas que operan secuencialmente, de modo que las instrucciones se mueven a través de estas etapas como en una cadena de montaje. Cada etapa puede estar trabajando en una instrucción diferente al mismo tiempo (ejecución simultánea). Explota el paralelismo entre las instrucciones de un flujo secuencial.

Se llama así porque en un extremo se aceptan nuevas entradas antes de que algunas entradas aceptadas con anterioridad aparezcan como salidas en el otro extremo (como en un pipe o tubería).

Si consideramos el procesamiento de una instrucción como dos etapas independientes, captación y ejecución, habrá períodos en la ejecución de una instrucción en los que no se accede a la memoria principal. Este tiempo podría usarse en captar la siguiente instrucción en paralelo con la ejecución de la actual. Mientras que la segunda etapa ejecuta la instrucción, la primera etapa utiliza algún ciclo de memoria no usado para captar y almacenar la siguiente instrucción (lo llamamos prebúsqueda, precaptación de instrucción o solapamiento de la captación).

¿Qué ventajas proporciona su implementación?

El pipelining es una técnica que mejora las prestaciones a nivel de diseño del hardware, es invisible al programador.

Este proceso acelerará la ejecución de instrucciones. Si las etapas de captación y ejecución fueran de igual duración, el tiempo del ciclo de instrucción se reduciría a la mitad. Pero el tiempo de ejecución suele ser más largo que el de captación (implica lectura y almacenamiento de operandos además de alguna operación) y además una instrucción de bifurcación condicional hace que la dirección de la siguiente instrucción a captar sea desconocida, por lo que la etapa de captación deba esperar hasta recibir la dirección de la misma desde la etapa de ejecución. Para conseguir mayor aceleración, el cauce debe partir las instrucciones en más etapas de duración similar (uniformizar las etapas al tiempo de la más lenta).

El incremento potencial de la segmentación del cauce es proporcional al número de etapas del cauce. Se incrementa la productividad pero no reduce el tiempo de ejecución de la instrucción.

El diseño de procesadores segmentados tiene gran dependencia del repertorio de instrucciones.

Atascos de un cauce (stall)

Problema: No todas las instrucciones necesitan todas las etapas, no todas las etapas pueden ser manejadas en paralelo y no se tienen en cuenta los saltos de control.

Los atascos son situaciones que impiden a la siguiente instrucción que se ejecute en el ciclo que le corresponde. Pueden ser:

- **Estructurales**: Provocados por conflictos por los recursos (dos o más instrucciones necesitan usar el mismo recurso de HW en el mismo ciclo)
- **Por dependencia de datos**: Dos instrucciones se comunican por medio de un dato que no está disponible cuando se necesita en una etapa determinada del cauce.
- **Por dependencia de control (o de instrucciones)**: La ejecución de una instrucción depende de cómo se ejecute otra. Una instrucción que modifica el valor del PC no lo ha hecho cuando se tiene que comenzar la siguiente. Existe una penalización por salto.

Los atascos se pueden solucionar con paradas del cauce, pero disminuye el rendimiento teórico ya que el CPI (Ciclos Por Instrucción)>1, mientras que el máximo rendimiento es completar una instrucción con cada ciclo de reloj.

Atascos estructurales: soluciones

La solución es simple: Replicar, segmentar o realizar turnos para el acceso a las unidades funcionales en conflicto.

- **Duplicación de recursos hardware**: sumadores o restadores además de la ALU
- **Separación en memorias de instrucciones y datos**
- **Turnar el acceso al banco de registros**: escrituras en la 1ra mitad de los ciclos de reloj y lecturas en la 2da mitad de los ciclos de reloj

Atascos por dependencia de datos

Tipos de dependencias de datos:

- Lectura después de Escritura (RAW, dependencia verdadera): Una instrucción genera un dato que lee otra posterior
- Escritura después de Escritura (WAW, dependencia en salida): Una instrucción escribe un dato después que otra posterior. Sólo se da si se deja que las instrucciones se adelanten unas a otras
- Escritura después de Lectura (WAR, antidependencia): Una instrucción modifica un valor antes de que otra anterior que lo tiene que leer, lo lea. No se puede dar en nuestro cauce simple.

Soluciones para riesgos RAW: Se debe determinar cómo y cuándo aparecen esos riesgos. Será necesaria una unidad de detección de riesgos y/o compilador más complejo.

Hay dos soluciones/técnicas:

- **Técnica Hardware (adelantamiento, forwarding o cortocircuito)**: Adelantamiento de operandos. Consiste en pasar directamente el resultado obtenido con una instrucción a las instrucciones que lo necesitan como operando. Si el dato necesario está disponible a la salida de la ALU (XI) se lleva a la entrada de la etapa correspondiente (Xi+1) sin esperar a la escritura (Mi o Wi).
    - Es fácil de implementar si se identifican todos los adelantamientos y se comunican a los registros de segmentación correspondientes.
- **Técnica Software**: Evita los riesgos *reordenando las instrucciones* del código sin afectar los resultados. Es realizada por el compilador:
    - Introducción de instrucciones NOP: Se genera retardo

Reordenación de instrucciones: Máxima separación de instrucciones con dependencia RAW. Cuidado con la ejecución “fuera de orden”.

Atascos por dependencia de control (producidos por saltos).

Las instrucciones de salto pueden ser:

- **Incondicional**: La dirección de destino se debe determinar lo más pronto posible, dentro del cauce, para reducir la penalización.
- **Condicional**: Introduce riesgo adicional por la dependencia entre la condición de salto y el resultado de una instrucción previa

Tratamiento: modificación sencilla de la ruta de datos para reducir la cantidad de paradas a un solo ciclo. Adelantar la resolución de los saltos a la etapa D:

- En ella se decodifica y se sabe que es un salto
- Se puede evaluar la condición de salto (con restador)
- Se puede calcular la dirección de salto (con sumador)

Salto condicional

Cuando hay una bifurcación condicional o salto, hasta que no se ejecuta la instrucción no hay forma de saber qué instrucción vendrá a continuación. Por lo general se carga la siguiente instrucción secuencialmente y continúa. Si el salto no se produce se obtiene el máximo provecho en rendimiento. Si el salto se produce, el cauce debe limpiarse de instrucciones

inútiles y se presenta una penalización en las prestaciones sufrida por no haber podido predecir el salto.

Técnicas para el tratamiento de saltos condicionales:

- **Flujos múltiples**: Duplicar las partes iniciales del cauce y dejar que éste capte las dos instrucciones utilizando los dos caminos. EL problema de esto es que con cauces múltiples hay retardos por la competencia por el acceso a registros y memoria. Además, pueden entrar en el cauce (cualquiera de los dos flujos) instrucciones de bifurcación adicionales antes de que se resuelva la bifurcación original
- **Precaptar el destino del salto**: Se precapta la instrucción del salto además de la siguiente a la bifurcación. Se guarda esta instrucción hasta que se ejecute la instrucción de bifurcación. Si se produce el salto, el destino ya habrá sido precaptado.
- **Buffer de bucles**: Un 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. Tiene tres utilidades:
    - Con la precaptación, se anticipa almacenando algunas instrucciones que secuencialmente están después de la dirección de donde se capta la instrucción actual, y las instrucciones que se capten secuencialmente estarán disponibles sin el tiempo de acceso a memoria habitual
    - Si ocurre un salto a un destino a solo unas pocas posiciones más allá de la dirección de la instrucción de bifurcación, el destino ya estará en el buffer. Es útil para secuencias IF-THEN e IF-THEN-ELSE
    - Si hay un bucle y el buffer de bucles es lo suficientemente grande como para contener todas las instrucciones de un bucle, entonces esas instrucciones solo necesitan ser captadas de la memoria una vez, durante la primera iteración. En las siguientes, ya estará en el buffer
- Predicción de saltos (para evitar la parada): técnica hardware
    - Técnicas estáticas: no dependen de la historia de la ejecución
        - Predecir que nunca se salta: Asume que el salto no se producirá y siempre capta la siguiente instrucción
        - Predecir que siempre se salta: Asume que el salto se producirá y siempre capta la instrucción destino del salto
        - Predecir según el código de operación: El procesador asume que el salto se producirá para ciertos códigos de operación de bifurcación y no para otros
    - Técnicas dinámicas: dependen de la historia de la ejecución, intentan mejorar la exactitud de predicción
        - Conmutador saltar/no saltar: Basado en la historia de las instrucciones, eficaz para los bucles
        - Tabla de historia de saltos (branch-target buffer): Pequeña caché asociada a la etapa de captación de instrucción del cauce. Cada elemento de la tabla consta de tres campos:
            - Dirección de una instrucción de bifurcación
            - Información de la instrucción destino (dirección del destino o instrucción destino)
            - Número de bits de estado (historia de uso de la instrucción)
- Salto retardado (o de relleno de ranura de retardo): técnica software
    - Se pueden mejorar las prestaciones de un cauce reordenando automáticamente las instrucciones de un programa, de forma que las instrucciones de salto tengan lugar después de lo realmente deseado (el compilador introduce instrucciones que se ejecutarán en cualquier caso después de la instrucción de salto).

Modulo 2

Interrupciones

Jerarquía de interrupciones

Si hay múltiples fuentes que pueden solicitar interrupción se establece cuales son mas importantes. Ser consideran

- No Enmascarables: las que NO pueden ignorarse. Indican eventos peligrosos o de alta prioridad.
- Enmascarables: pueden ser ignoradas. Con instrucciones podemos inhibir la posible solicitud.

Interrupciones múltiples

Interrupciones inhabilitadas

- El procesador puede y debe ignorar la señal de petición de interrupción si se produce una interrupción en ese momento.
- Si se hubiera generado una interrupción se mantiene pendiente y se examinará luego una vez que se hayan habilitado nuevamente.
- Ocurre una interrupción, se inhabilitan, se gestiona la misma y luego se habilitan otra vez.
- Por lo tanto las interrupciones se manejan en un orden secuencial estricto.

Definir prioridades

- Una interrupción de prioridad más alta puede 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.

Reconocimiento de interrupciones

Interrupciones multinivel

- Cada dispositivo que puede provocar una interrupción tiene una entrada física de interrupción conectada a la CPU.
- Es muy sencillo, pero muy caro.

Línea de interrupción única

- Una sola entrada física de pedido de interrupción a la que están conectados todos los dispositivos.
- Se debe “preguntar” a cada dispositivo si ha producido el pedido de interrupción (técnica Polling/encuesta).

Interrupciones vectorizadas

- El dispositivo que quiere interrumpir además de la señal de pedido de interrupción, debe colocar en el bus de datos un identificador (vector).
    - Lo coloca el periférico directamente ó
    - Controlador de Interrupciones (que se ocupa de todo).

Interrupciones - PIC

- Finalidad de las interrupciones. Para que se utiliza un controlador de interrupciones.
- ¿Qué es una interrupción? Describa cómo funcionan. ¿Cómo se utiliza un controlador de interrupción?
- Explique el mecanismo de interrupción. Describa las características y el funcionamiento de un pic
- ¿Cómo es la estructura de un módulo de E/S? Describa las características del controlador de interrupciones PIC.

¿Qué son las interrupciones?

Mecanismo mediante el que otros módulos (E/S, memoria) pueden interrumpir el procesamiento normal de la CPU.

Son eventos que indican que existe una condición en algún lugar del sistema, o del programa en ejecución, que requiere la atención del procesador. Generalmente resultan en una transferencia forzada del flujo de ejecución hacia una rutina denominada “manejador de interrupciones”. Las interrupciones se asocian normalmente a eventos de hardware, mientras que las excepciones se producen cuando se detectan ciertas condiciones durante la ejecución, como división por cero, fallos de página, violaciones de segmento, etc.

¿Para qué son las interrupciones? ¿Cómo funcionan?

Las interrupciones proporcionan una forma de mejorar la eficiencia del procesador. Los dispositivos externos suelen ser mucho más lentos que el procesador y esperar por las instrucciones de E/S sería un desperdicio de procesador. Con las interrupciones, el procesador puede dedicarse a ejecutar otras instrucciones mientras una operación de E/S está en curso. La operación de E/S se realiza concurrentemente con Íla ejecución de instrucciones del programa de usuario.

Cuando el dispositivo externo está listo para aceptar más datos del procesador, el módulo de E/S de este dispositivo externo envía una señal de petición de interrupción al procesador. Éste suspende la operación del programa que estaba ejecutando y salta a un programa llamado “gestor de interrupción” que da servicio al dispositivo en cuestión y luego continúa con el programa original.

Una interrupción es una *interrupción* en la secuencia normal de funcionamiento. Cuando la misma se completa, la ejecución sigue: el procesador y el SO son los responsables de detener el programa de usuario y después permitir que prosiga desde el mismo punto.

Mecanismo de interrupción

Se añade al ciclo de instrucción un ciclo de interrupción donde el procesador comprueba si se generó alguna interrupción. Si no hay señal de interrupción se avanza con la instrucción siguiente y si hay señal:

1. Suspende la ejecución del programa en curso y guarda su contexto (el contenido actual del PC y demás)
2. Carga el PC con la dirección de comienzo de una rutina de gestión de interrupción

El mecanismo para el manejo de interrupciones y excepciones en la arquitectura IA-32 permite que éstas sean manipuladas de manera transparente a los programas de aplicación y al mismo

Sistema Operativo. Cuando se genera una interrupción, el procedimiento en ejecución se suspende automáticamente mientras el procesador ejecuta el manejador correspondiente; cuando esta operación se termina, el procesador reanuda la ejecución de la tarea interrumpida. La reanudación del proceso sucede sin pérdida de la continuidad del programa, a menos que el retorno no sea posible o que el evento haya causado la terminación del programa.

Tipos de interrupciones

Las interrupciones pueden ser de dos orígenes:

- **Internas** (generadas por software):
  - Generalmente usadas para hacer llamadas a funciones del SO (esto permite que las subrutinas del sistema se carguen en cualquier lugar)
  - No requieren conocer la dirección de la rutina en tiempo de ejecución
- Traps/Excepciones: Interrupciones por hardware creadas por el procesador en respuesta a ciertos eventos como:
  - Condiciones excepcionales: overflow en ALU de punto flotante.
  - Falla de programa: tratar de ejecutar una instrucción no definida.
  - Fallas de hardware: error de paridad de memoria.
  - Accesos no alineados ó a zonas de memoria protegidos
- **Externas a la CPU** (generadas por hardware):
  - Generadas por dispositivos de E/S
  - Son las “verdaderas” interrupciones
  - El sistema de cómputo tiene que manejar estos eventos externos “no planeados” o “asincrónicos”
  - No están relacionadas con el proceso en ejecución en ese momento
  - Son conocidas como interrupt request

Clases de interrupciones

- **Programa**: Por el resultado de una ejecución de una instrucción (Ej: Overflow, división por 0, intento de ejecución de una instrucción inexistente, intento de acceso fuera del espacio de memoria permitido para el usuario)
- **Temporización**: Clock interno del procesador que permite al SO realizar ciertas funciones de manera regular
- **E/S**: Por una operación de E/S (Ej: indicar la finalización normal de una operación o avisar condiciones de error)
- **Fallo de HW** (Ej: Error de paridad en la memoria, pérdida de energía, etc)

Controlador de interrupciones (PIC)

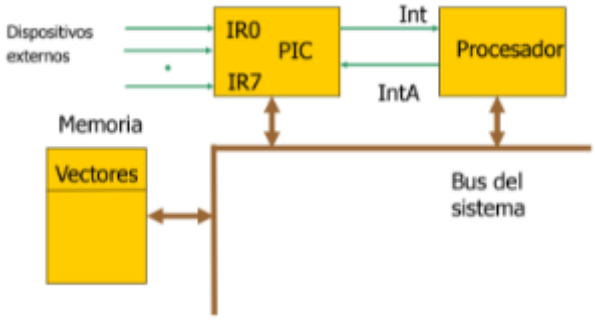
¿Qué es?

El dispositivo controlador programable de interrupciones (PIC) es un chip que sirve si el procesador tiene una única entrada de pedido de interrupciones o si tenemos varios productores de interrupciones.

Es un dispositivo usado para combinar varias fuentes de interrupciones sobre una o más líneas del CPU. Se maneja con prioridades (las de número mas bajo se atienden primero) y tiene un conjunto de registros internos:

- IRR (Interrupt Request Register): Registro de petición de interrupción, indica con bit en 1 las interrupciones demandadas hasta el momento.
- ISR (In Service Register): Registro de interrupción en servicio, indica con bit en 1 cuál es la interrupción que está siendo atendida.

- IMR (Interrupt Mask Register): Registro de maścara de interrupciones, permite el enmascaramiento selectivo de cada una de las entradas de interrupción, indicando con bit en 1. Tras un reset los bits de este registro quedarán en 0. Indica cuáles deben ser ignoradas.
- EOI (End of Interruption): Fin de interrupción. Como consecuencia, se pone en 0 el bit del ISR correspondiente.
- INT0...INT7: 8 registros, donde carga el valor del vector de interrupción correspondiente



Tareas realizadas por el PIC

- Puesto que existen muchos dispositivos que pueden solicitar interrupciones, el PIC debe priorizarlas cuando existen varias IRQ's simultáneas
- Después de enviar una solicitud de interrupción, debe enviar un número de interrupción (número de vector) cuando el procesador indica que está listo para atender la petición
- Mantiene un registro de que se está procesando una interrupción: cuando esto sucede, no envía más peticiones al procesador hasta que este le responde con una señal de EOI (End Of Interrupt), indicando que la rutina de servicio precedente ha terminado o puede aceptar otra interrupción
- Puede enmascarar de forma selectiva cualquiera de las 8 IRQ's que tiene conectadas

¿Cómo funciona?

El controlador de interrupciones puede manejar hasta ocho peticiones de interrupción independientes al mismo tiempo, numeradas de la 0 (INT0) a la 7 (INT7), de las cuales seleccionará una única para presentarla a la entrada de interrupción INT de la CPU.

Si más de una petición de interrupción se producen exactamente al mismo tiempo entonces el PIC las pasa a la CPU en un orden de prioridad, donde la petición por la entrada 0 tiene la prioridad más alta y la de la 7 la menor.

En el ciclo de interrupción, el procesador comprueba si se ha generado alguna interrupción (indicada por una señal -flag- de pedido de interrupción). Si no hay señal, el procesador continúa con el ciclo de captación (capta la instrucción siguiente) y si hay alguna interrupción pendiente:

- Suspende la ejecución del programa en curso y guarda su contexto (dirección de la siguiente instrucción a ejecutar -el contenido del PC- y el estado del procesador)
- Carga el PC con la dirección de comienzo de una rutina de gestión de interrupción
- El procesador prosigue con el ciclo de captación y accede a la primera instrucción del programa de gestión de interrupción que dará servicio a la interrupción.
- Al completarse la rutina de gestión de interrupción, el procesador prosigue la ejecución del programa de usuario en el punto que se interrumpió

Con el uso de interrupciones, el procesador puede dedicarse a ejecutar otras instrucciones mientras una operación de E/S está en curso. El procesador no tiene que comprobar repetidamente el estado del módulo hasta que el mismo pueda transmitir o recibir datos, siendo los dispositivos mucho más lentos que el procesador

RISC vs. CISC

- Describa las características que diferencian a los procesadores RISC respecto de los CISC

RISC

RISC (Reduced Instruction Set Computer): Repertorio reducido de instrucciones

- Repertorio de instrucciones limitado y con un formato fijo (sencillo)
- Número grande de registros o la utilización de un compilador que optimice el uso de éstos
- Énfasis en la optimización del cauce de instrucciones

RISC se presta a una segmentación eficiente, porque hay menos operaciones llevadas a cabo por instrucción y éstas son más previsibles. 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.

Características del RISC

- Una instrucción por ciclo: Se ejecuta una instrucción máquina cada ciclo máquina. Con instrucciones sencillas y de un ciclo, hay poca necesidad de microcódigo y las máquinas pueden estar cableadas (significa que no hay que acceder a la memoria de control de microprograma durante la ejecución de la instrucción)
- Operaciones registro a registro: La mayoría de las operaciones son así, lo que simplifica el repertorio de instrucciones y fomenta la optimización del uso de los registros (los operandos accedidos frecuentemente permanecen en el almacenamiento de alta velocidad)
- Modos de direccionamiento sencillos: se usa principalmente el direccionamiento a registro
- Formatos de instrucción sencillos: Se usan pocos formatos, la longitud de las instrucciones es fija y alineada en los límites de una palabra. Las posiciones de los campos, especialmente la del código de operación, son fijas. Todo esto simplifica la unidad de control.
    - Formato de instrucción fijo: como ventaja, la decodificación del código de operación y el acceso a los operandos en registros puede hacerse de forma simultánea

Pero también: Requiere mayor tiempo/esfuerzo de compilación

Se considera RISC a:

- Un único tamaño de instrucción (típicamente de 4 bytes)
- Pocos modos de direccionamiento de datos, normalmente menos que 5
- No se usa direccionamiento indirecto que requiera efectuar un acceso a memoria
- No hay operandos que combinen carga/almacenamiento con cálculos aritméticos (ej: suma desde o a memoria)
- No se direcciona más de un operando

CISC

CISC (Complex Instruction Set Computer): Repertorio complejo de instrucciones

- Repertorio de instrucción más rico, con un número mayor de instrucciones y más complejas.

Finalidad del CISC

- Facilitar el trabajo del escritor de compiladores
- Mejorar la eficiencia de la ejecución: secuencias complejas de operaciones en microcódigo
- Dar soporte a HLL más complejos

Inconvenientes del CISC

- El software es mucho más caro que el hardware
- El nivel del lenguaje era cada vez más complicado
- Salto semántico: Diferencias entre operaciones HLL y operaciones de la arquitectura
- Todo esto conduce a:
    - Repertorios de instrucciones grandes
    - Más modos de direccionamiento
    - Varias sentencias de HLL implementadas en el hardware (Por ejemplo, el CASE del VAX)

¿Simplificación del compilador?

Las instrucciones máquina complejas son con frecuencia difíciles de aprovechar, ya que el compilador debe descubrir aquellos casos que se ajustan perfectamente a la construcción. Optimizar el código generado para minimizar su tamaño, reducir el número de instrucciones ejecutadas y mejorar la segmentación, es mucho más difícil con un repertorio complejo de instrucciones (CISC).

¿Programas más pequeños?

CISC busca que los programas sean más pequeños (ocupan menos memoria) y rápidos. Pero la memoria hoy en día es barata, por lo que no es una gran ventaja.

Si bien un programa para un CISC expresado en lenguaje máquina simbólico puede ser más corto (tiene menos instrucciones) que el RISC, el número de bits de memoria que ocupa no tiene por qué ser más pequeño. Al haber más instrucciones en un CISC, necesitan códigos de operación más largos (produciendo operaciones más largas). Además, RISC acentúa las referencias a registros en lugar de a memoria y los mismos necesitan menos bits.

¿Programas más rápidos?

CISC al ser propenso a usar instrucciones más sencillas, requiere de una Unidad de Control más compleja y/o la memoria de control del microprograma debe hacerse más grande, de modo que aumenta el tiempo de ejecución de las instrucciones simples.

RISC frente a CISC

RISC vs. CISC

CISC	RISC
Emphasis on hardware	Emphasis on software
Multiple instruction sizes and formats	Instructions of same set with few formats
Less registers	Uses more registers
More addressing modes	Fewer addressing modes
Extensive use of microprogramming	Complexity in compiler
Instructions take a varying amount of cycle time	Instructions take one cycle time
Pipelining is difficult	Pipelining is easy

- No existe una clara barrera diferenciadora
- Muchos diseños incluyen características de ambos criterios (CISC y RISC)
    - Por ejemplo: PowerPC y Pentium II

Controversia RISC y CISC

El trabajo que se ha hecho para evaluar las ventajas de la aproximación RISC se pueden agrupar en dos categorías:

- Cuantitativa: Intentos de comprar el tamaño de los programas y su velocidad de ejecución en máquinas RISC y CISC de similar tecnología
- Cualitativa: Revisión de asuntos tales como soporte de lenguajes de alto nivel y uso óptimo de recursos VLSI

Problemas de las comparaciones:

- No existe un par de máquinas RISC y CISC directamente comparables
- No hay un conjunto de programas de prueba definitivo
- Es difícil separar los efectos del hardware de los del compilador
- La mayoría de los análisis comparativos se han hecho con máquinas de “juguete”, no con productos comerciales
- La mayoría de las máquinas son una mezcla de ambas

MODULO 3 E/S

Identificación del módulo que interrumpe

- Diferentes líneas para cada módulo
    - PC
    - Limita el número de dispositivos
- Consulta software (Poll o encuesta)
    - Ocurrido un pedido de interrupción la CPU consulta a cada módulo para determinar quien fue el demandante.
    - Resulta lento.

- Conexión en cadena (daisy chain) “hard poll”
    - La línea de reconocimiento de interrupción se conecta encadenando los módulos, la línea de pedido es compartida.
    - Una vez enviada la confirmación de parte de la CPU el módulo responderá colocando un vector (palabra), en el bus, que lo identifica.
    - La CPU emplea el vector como puntero para acceder a la rutina de servicio.

Interrupciones múltiples

- Todas las líneas de interrupción tienen un orden de prioridad
- Las líneas con más prioridad pueden interrumpir a las líneas con menor prioridad.
- Si existe un maestro del bus, solo él puede interrumpir.

Análisis

- Las operaciones de E/S mediante interrupciones son más efectivas que las programadas.
- Pero ambas necesitan la intervención directa de la CPU.
    - La velocidad de transferencia es limitada.
    - La CPU permanece ocupada mucho tiempo durante la operación.

Desventajas DMA

- Se puede degradar el rendimiento de la CPU si el DMAC hace uso intensivo del bus
    - Si el bus está ocupado en una transferencia DMA, la CPU no puede acceder a memoria para leer instrucc. / datos
- El problema se reduce con el uso de memoria cache
    - La mayor parte del tiempo, la CPU lee instruc. de la cache, por lo que no necesita usar el bus de memoria.
    - El DMAC puede aprovechar estos intervalos en los que la CPU está leyendo instrucciones de la cache (y por tanto no usa el bus de memoria) para realizar las transferencias.
- En caso de computadores sin cache
    - El procesador no utiliza el bus en todas las fases de la ejecución de una instrucción.
    - El DMAC puede aprovechar las fases de ejecución de una instrucción en las que la CPU no utiliza el bus para realizar sus transferencias.

Tipos de transferencias

- Si el DMAC sólo toma el control del bus durante los intervalos de tiempo en los que la CPU no hace uso del mismo el rendimiento del sistema no sufrirá degradación alguna
- Se distinguen dos tipos de transferencias:
    - Por ráfagas (burst)
    - Por robo de ciclo (cycle-stealing)

DMA modo ráfaga

- El DMAC solicita el control del bus a la CPU
- Cuando la CPU concede el bus, el DMAC no lo libera hasta haber finalizado la transferencia de todo el bloque de datos completo.
- VENTAJAS: La transferencia se realiza de forma rápida.
- DESVENTAJAS: Durante el tiempo que dura la transferencia la CPU no puede utilizar el bus con memoria, lo que puede degradar el rendimiento del sistema.

DMA modo robo de ciclo

- El DMAC solicita el control del bus a la CPU.
- Cuando la CPU concede el bus al DMAC, se realiza la transferencia de una única palabra y después el DMAC libera el bus.
- El DMAC solicita el control del bus tantas veces como sea necesario hasta finalizar la transferencia del bloque completo
- VENTAJAS: No se degrada el rendimiento del sistema.
- DESVENTAJAS: La transferencia tarda más tiempo en llevarse a cabo.
- Para la CPU no es una interrupción: El procesador no debe guardar el contexto.
- Si bien el trabajo de la CPU es lento, no será tanto como si ella realizara la transferencia.
- Por lo tanto, para transferencia de E/S de múltiples palabras, es la técnica más eficiente.

Canales de E/S

Los dispositivos de E/S son cada vez más sofisticados. Ej: tarjetas gráficas 3D.

Evolución:

1. La CPU controla directamente los periféricos.

2. Se agrega un módulo de E/S o controlador.

3. Idem 2 más llamado de interrupción.

4. El módulo de E/S provee el acceso directo a memoria (DMA).

5. El módulo de E/S tiene su propio procesador con su pequeño conjunto de instrucciones.

6. El módulo además tiene su memoria local o sea se convierte en una computadora en sí mismo.

Características de Canales de E/S

- Representan una extensión al concepto de DMA
  - Tienen la habilidad de ejecutar instrucciones de E/S
- Completo control de la transferencia de datos
  - por lo tanto la CPU no ejecuta instrucciones de E/S
- Programa almacenado en memoria principal
- La CPU inicia la transferencia de E/S
  - Ordena ejecutar el programa que está en memoria
  - El programa especifica dispositivos, áreas de memoria a usar, prioridades y acciones ante errores

Tipos de canales de E/S

Selector

- Controla varios dispositivos de alta velocidad y uno por vez, por lo tanto el canal se dedica para la transferencia de datos de ese dispositivo.
- El canal selecciona un dispositivo y efectúa la transferencia.
- Los dispositivos son manejados por un controlador o módulo de E/S

- Por lo tanto el canal de E/S ocupa el lugar de la CPU en el control de esos controladores.

Multiplexor

- Puede manejar E/S con varios dispositivos a la vez.
- Multiplexor de bytes: Acepta y transmite caracteres.

- Multiplexor de bloques: Intercala bloques de datos desde distintos dispositivos.

Módulo de E/S

- ¿Cómo funciona un módulo de E/S? Describa las características fundamentales de un DMA.
- Describir la estructura de un módulo de E/S. ¿Qué es DMA y cómo funciona?
- ¿Cómo es la estructura interna de un módulo de E/S? Describa las características funcionales del acceso directo a memoria - DMA
- Estructura interna del módulo de E/S. Características funcionales de DMA
- Describa las características fundamentales de un DMA
- Estructura de un módulo E/S. Describa el funcionamiento de un controlador DMA(las etapas de transferencia)
- ¿Como es la estructura de un módulo de E/S? Describa las posibles técnicas que puede utilizar una CPU para realizar operaciones de E/S.

¿Cómo funciona un módulo de E/S?

El funcionamiento de un módulo de E/S permite que el procesador vea a una amplia gama de dispositivos de una forma simplificada. El módulo de E/S debe ocultar los detalles de temporización, formatos y electromecánicos de los dispositivos externos para que el procesador pueda funcionar únicamente en términos de órdenes de lectura y escritura. Realiza la interfaz entre el procesador y la memoria (bus) y los periféricos.

Estructura de un módulo de E/S

El módulo se conecta al resto del computador a través de un conjunto de líneas (como las líneas del bus del sistema).

- Registro de datos: Los datos que se transfieren a y desde el módulo se almacenan temporalmente en uno o más registros de datos.
- Registro de estado/control: 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 información de control del procesador

La lógica que hay en el módulo interactúa con el procesador a través de una serie de líneas de control. Son las que usa el procesador para proporcionar las órdenes al módulo de E/S. Algunas de las líneas de control pueden ser utilizadas por el módulo de E/S (ej: para las señales de arbitraje y estado). 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 de direcciones. Por último, el módulo de E/S posee la lógica específica para la interfaz con cada uno de los dispositivos que controla.

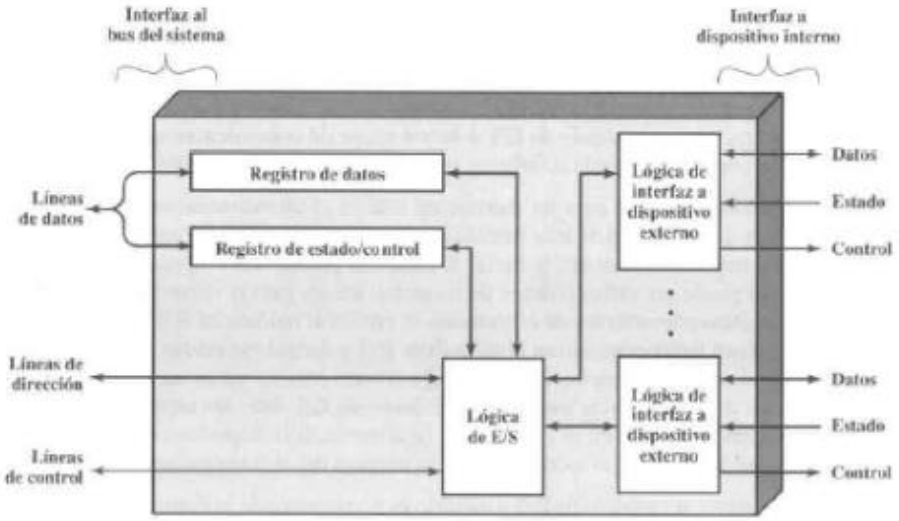


Figura 7.3. Diagrama de bloques de un módulo de E/S.

Técnicas de gestión de E/S

- E/S programada con espera de respuesta:

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: Comprobación de 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 (permanece ociosa, algo no deseable).

Resumen: Cuando el procesador está ejecutando un programa y encuentra una instrucción relacionada con una E/S, ejecuta dicha instrucción mandando una orden al módulo de E/S apropiado. El módulo realiza la acción solicitada y después activa los bits apropiados en el registro de estado, de E/S. El módulo no interrumpe al procesador, sino que éste es responsable de comprobar periódicamente el estado del módulo de E/S hasta que encuentra que la operación ha terminado.

Desventaja: El procesador tiene que esperar un tiempo considerable a que el módulo de E/S esté preparado. Espera comprobando repetidamente su estado, degradando el nivel de prestaciones de todo el sistema

- E/S con 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. Ahí el procesador ejecuta la transferencia de datos como antes y luego continúa con el procesamiento previo. 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 memoria principal en una entrada.

¿Qué hace la CPU? La CPU envía una orden de lectura y continúa su trabajo, mientras el módulo obtiene los datos del periférico. Al final de cada ciclo de instrucción la CPU comprueba las interrupciones. Cuando el módulo E/S emite un pedido de interrupción a la CPU, la misma

lo detecta, guarda el contexto (PC y registros del procesador), interrumpe el proceso y procesa la interrupción. La CPU solicita los datos que el módulo transfiere, y los almacena en memoria. Finalmente recupera el contexto del programa que estaba ejecutando (o de otro) y continúa la ejecución.

Ventaja: No repite la comprobación de los estados de los módulos. Es más eficiente porque elimina las esperas innecesarias.

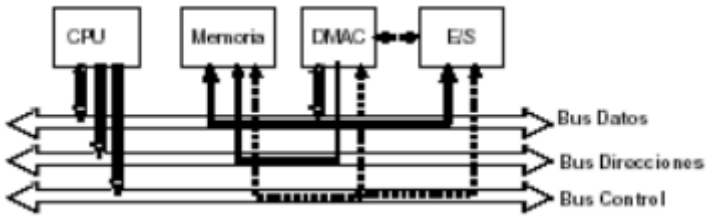
Desventaja: Se consume gran cantidad de tiempo del procesador, porque cada palabra de datos que va desde memoria al módulo de E/S o viceversa, debe pasar por el procesador.

- E/S con acceso directo a memoria (DMA)

A diferencia de los anteriores, acá el módulo de E/S y la memoria principal intercambian datos directamente, sin la intervención del procesador (lo cual implica una velocidad de transferencia limitada y una sobrecarga de la CPU, un gran problema para transmisiones de grandes volúmenes). Para grandes volúmenes de datos, existe el DMA.

El DMA requiere un módulo adicional en el bus del sistema. El módulo de DMA es capaz de imitar al procesador y 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, debe utilizar el bus solo cuando el procesador no lo necesita, o debe forzar al procesador a que suspenda temporalmente su funcionamiento (robo de ciclo o cycle stealing)

Básicamente es capaz de controlar una transferencia de datos entre un periférico y memoria sin intervención de la CPU.



Funcionamiento del DMA

Cuando el procesador desea leer o escribir un bloque de datos, envía una orden al módulo de DMA con la información:

- Si se solicita una lectura o una escritura, usando la línea de control
- La dirección del dispositivo de E/S en cuestión, usando la línea de datos
- La posición inicial de memoria a partir de donde se lee o escribe, usando la línea de datos y almacenada por el DMA en su registro de direcciones
- El número de palabras a leer o escribir, usando la línea de datos y almacenando en el registro de cuenta de datos

Después el procesador continúa con otro trabajo, habiendo delegado la operación de E/S al módulo de DMA. El DMA transfiere el bloque completo de datos, palabra a palabra, directamente desde o hacia memoria sin que pase por el procesador. Cuando la transferencia terminó, el módulo de DMA envía una señal de interrupción al procesador. Así el procesador solo interviene al comienzo y al final de la transferencia.

Nota: No es una interrupción, el procesador no guarda el contexto, sino que espera durante un ciclo de bus. Eso hace que el procesador sea más lento ejecutando programas, aunque para

una transferencia de E/S de varias palabras el DMA es mucho mas eficiente que la E/S mediante interrupciones o programada.

Modulo 4 BUSES

Bus de datos

- Transmite datos.
    - Recuerde que a este nivel no existe diferencia alguna entre “datos” e “instrucciones”.
- El ancho del bus es un factor clave a la hora de determinar las prestaciones.
    - 8, 16, 32, 64 bits.

Bus de dirección

- Identifica la fuente o destino de un ‘dato’
    - cuando el procesador desea leer una palabra de una determinada parte en la memoria.
- El ancho del bus de direcciones determina la máxima capacidad de memoria posible en el sistema.
    - MSX88 tiene un bus de dirección de 16 bits, lo que define un espacio para direcciones de 64K lugares

Bus de control

- Transmite información de señales de control y temporización
    - Señal de escritura/lectura en memoria.
    - Petición de interrupción.
    - Señales de reloj.

Problemas de un único bus

- Conectar gran número de dispositivos a un bus producen Retardos de propagación
    - Si el control del bus pasa de un dispositivo a otro, puede afectar sensiblemente a las prestaciones.
- La mayoría de los sistemas utilizan varios buses para solucionar estos problemas.
    - Jerarquía de buses

Buses

- ¿Que es un Bus? Describa los diferentes tipos, métodos de arbitraje y técnicas de sincronización. Mencione las principales diferencias entre PCI y SCSI.
- Que es un Bus, tipos de buses, temporización y métodos de arbitraje

Qué es un bus?

Un bus es un camino de comunicación entre dos o más dispositivos. Es un medio de transmisión compartido: al bus se conectan varios dispositivos y cualquier señal transmitida por uno de esos dispositivos está disponible para que los otros dispositivos conectados al bus

puedan acceder a ella. Solo un dispositivo puede transmitir con éxito en un momento dado (para que sus señales no se solapen y se distorsionen).

Un bus está constituido por varios caminos de comunicación o líneas. Cada línea es capaz de transmitir señales binarias. En un intervalo de tiempo, se puede transmitir una secuencia de dígitos binarios a través de una única línea. Se pueden usar varias líneas del bus para transmitir dígitos binarios simultáneamente (Por ej: un dato de 8 bits puede transmitirse mediante ocho líneas del bus).

Una computadora tiene distintos tipos de buses que proporcionan comunicación entre sus componentes. El bus que conecta los componentes principales (procesador, memoria y E/S) se llama bus del sistem.

Tipos de buses

Las líneas del bus se pueden dividir en dos tipos genéricos:

- Dedicadas: está permanentemente asignada a una función o a un subconjunto físico de componentes del computador. Ej: el uso de líneas separadas para direcciones y para datos.
    - 16 líneas de direcciones
    - 16 líneas de datos
    - 1 línea de control de lectura o escritura
- Multiplexadas: Uso de las mismas líneas para usos diferentes. Ventaja: uso de menos líneas, lo cual ahorra espacio y costes. Desventaja: Necesita una circuitería más compleja en cada módulo, además de que pueden reducirse las prestaciones debido a que los eventos que deben compartir las mismas líneas no pueden producirse en paralelo.
    - 16 líneas de direcciones ó datos
    - 1 línea de control de lectura o escritura
    - 1 línea de control para definir direcciones o datos

Método de arbitraje

Más de un módulo puede necesitar el control de un bus y como solo una unidad puede transmitir a través del bus en un momento dado, se requiere algún método de arbitraje:

- Centralizados: Un único dispositivo hardware (llamado 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.
- Distribuidos: No existe un controlador central, sino que cada módulo dispone de lógica para controlar el acceso y los módulos actúan conjuntamente para compartir el bus.

En ambos métodos, el propósito es designar un dispositivo, el procesador o un módulo de E/S como maestro del bus. El maestro puede iniciar una transferencia de datos con otro dispositivo que actúa como esclavo en este intercambio concreto.

Técnicas de sincronización/temporización

La temporización es la forma en la que se coordinan los eventos en el bus. Puede ser:

- 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 cual se transmite una secuencia en la que se alternan intervalos regulares de igual duración.

    - Un intervalo (de un 1 seguido de un 0) se conoce como ciclo de reloj o ciclo de bus y define un intervalo de tiempo unidad.
    - Todos los dispositivos del bus pueden leer la línea de reloj y todos los eventos empiezan al principio del ciclo de reloj (suelen sincronizar en el flanco de subida)
    - La mayoría de los eventos se prolongan durante un único ciclo de reloj
- Temporización asíncrona:
    - La presencia de un evento en el bus es consecuencia y depende de que se produzca un evento previo

La síncrona es más fácil de implementar y comprobar, pero es menos flexible que la temporización asíncrona, ya que todos los dispositivos de un bus síncrono deben utilizar la misma frecuencia de reloj y el sistema no puede aprovechar mejoras en las prestaciones de los dispositivos. Con la asíncrona, pueden compartir el bus una mezcla de dispositivos lentos y rápidos.

Diferencias entre PCI y SCSI

El bus PCI (Peripheral Component Interconnect - Interconexión de Componente Periférico) es un bus de ancho de banda elevado, independiente del procesador, que se puede utilizar como bus de periféricos o para una arquitectura de entreplanta. Comparado con otras especificaciones comunes de bus, proporciona mejores prestaciones para los subsistemas de E/S de alta velocidad (ej: los adaptadores de pantalla gráfica, los controladores de interfaz de red, los controladores de disco, 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.

El PCI está diseñado para permitir una cierta variedad de configuraciones basadas en microprocesadores, incluyendo sistemas tanto de uno como de varios procesadores.. Utiliza la temporización síncrona y un esquema de arbitraje centralizado.

SCSI (Small Computer System Interface) sólo se utiliza para dispositivos de almacenamiento y debe tener un controlador de interfaz

Caché

- Justifique el uso de dos niveles de caché
- Caché: mencione algoritmos de reemplazo y políticas de escritura.
- Describa las funciones de correspondencia entre memoria principal y caché. Analice las políticas de escritura desde el punto de vista de la coherencia de datos.
- Mencionar los tipos de correspondencia de la memoria caché. Describir las políticas de escritura (en acierto y en fallo)
- ¿Por qué funciona una jerarquía de memoria? Describa las políticas de ubicación y de reemplazo de bloques en memoria caché.
- Describa las técnicas de ubicación de bloques y las politicas de escritura en Cache.
- Describa las funciones que se utilizan en la política de ubicación de bloques en memoria caché. Analice las políticas de escritura de datos desde el punto de vista de la coherencia de los mismos en la jerarquía.
- Describa las técnicas de reemplazo de bloque, correspondencia y políticas de escritura en memoria cache.

Dos niveles de caché

La caché de dos niveles consta de:

- la caché interna, el nivel 1 (L1)

- la caché externa, el nivel 2 (L2)

Si no hay caché L2 y el procesador hace una petición de acceso a una posición de memoria que no está en la caché L1, el procesador debe acceder a la DRAM o la ROM a través del bus. Debido a la lentitud usual del bus y a los tiempos de acceso de las memorias, se obtienen bajas prestaciones.Pero si se usa una caché L2 SRAM, entonces con frecuencia la información que falta puede recuperarse fácilmente. Si la SRAM es rápida, los datos pueden accederse con cero estados de espera, el tipo de transferencia de bus más rápido.

Actualmente se estila separar la cache en dos: una dedicada a instrucciones y otra a datos. La ventaja de caché unificada es tener una tasa de aciertos mayor que una partida, ya que nivela automáticamente la carga entre captación de instrucciones y de datos. Además, solo se necesita diseñar e implementar una caché.

La ventaja de caché partida es que elimina la competición por la caché entre el procesador de instrucciones y la unidad de ejecución, importante en diseños que cuentan con segmentación de cauce de instrucciones (pipelining)

En resumen: el aumento de densidad de integración permitió tener una caché en el mismo chip del procesador. Esto reduce la actividad del bus externo del procesador y por lo tanto reduce los tiempos de ejecución, e incrementa las prestaciones globales del sistema. Actualmente se incluyen tanto cache on-chip como internos. La estructura más sencilla se denomina caché de dos niveles. En general, el uso de caché multinivel mejora las prestaciones, pero aumenta la complejidad del diseño.

Algoritmos de reemplazo

Algoritmos de reemplazo / políticas de reemplazo / algoritmos de sustitución:

Cuando se introduce un nuevo bloque en la caché, debe sustituirse uno de los bloques existentes.

Para correspondencia directa, solo hay una posible línea para cada bloque particular, por lo que no hay elección y se sustituye por el que ocupa el lugar del nuevo.

Para las técnicas asociativas se requieren algoritmos de sustitución que deben implementarse en hardware para conseguir alta velocidad. Los más comunes:

- LRU: Sustituye el bloque que se ha mantenido en la cache por mas tiempo sin haber sido referenciado (debería dar la mejor tasa de aciertos). Requiere controles de acceso (Cada línea tiene un flag que se pone en 1 al ser referenciada)
- FIFO: El que ha estado más tiempo en la caché. Requiere controles de tiempo (se puede implementar con una técnica de round robin)
- LFU: Sustituye aquel bloque del conjunto que ha experimentado menos referencias. Requiere controles de uso (se implementan contadores a cada línea)
- Aleatoria: Agarra una línea al azar entre las posibles candidatas.

Políticas de escritura

Antes de que pueda ser reemplazado un bloque que está en una línea de caché, es necesario comprobar si ha sido alterado en caché pero no en memoria principal.

Si no se modificó, puede escribirse sobre la línea de caché. Si se modificó, significa que se ha realizado al menos una operación de escritura sobre una palabra de la línea correspondiente de la caché. Hay dos cosas que hay que tener en cuenta:

- Más de un dispositivo puede tener acceso a la memoria principal
    - Si un módulo de E/S puede escribir/leer directamente en/de memoria, si una palabra ha sido modificada solo en la caché, la correspondiente palabra de memoria no es válida. Y si el dispositivo ha alterado la memoria principal, entonces la palabra de caché no es válida.
- Varios procesadores se conectan al mismo bus y cada uno de ellos tiene su propia caché local
    - Si se modifica una palabra en una de la caché, podría presumiblemente invalidar una palabra de otras cache

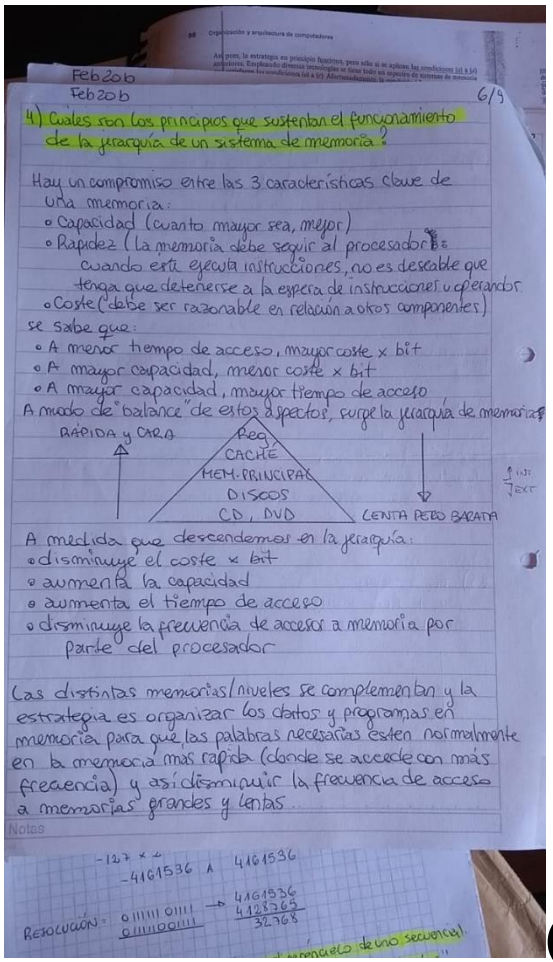
Las técnicas en acierto:

- Escritura inmediata (write-through): Es la más sencilla, todas las operaciones de escritura se hacen tanto en cache como en memoria principal, asegurando que el contenido de la memoria principal siempre es válido. Se actualizan simultáneamente la posición de la caché y de la memoria principal. Desventaja: con múltiples CPU, se genera un tráfico sustancial a memoria que puede originar un cuello de botella y retrasa la escritura. Se debe observar el tráfico a memoria principal para mantener actualizada la caché local
- Post-escritura (write-back): minimiza las escrituras en memoria, ya que las actualizaciones se hacen solo en la caché. Cuando tiene lugar una actualización, se activa un bit ACTUALIZAR (bit de “sucio”) asociado a la línea. Después, cuando el bloque es sustituido, es (post-)escrito en memoria principal si y sólo si el bit ACTUALIZAR está activo. Desventaja: A veces, porciones de memoria principal no son válidas y los accesos por parte de los módulos de E/S sólo podrían hacerse a través de la caché, complicando la circuitería y generando un potencial cuello de botella (la memoria principal puede contener información errónea en algún momento).

Las técnicas en fallo:

- Write allocate: La información se lleva de la memoria principal a la caché. Se sobreescribe en la caché
    - Habitual con write-back
- No-write allocate: El bloque no se lleva a la memoria caché. Se escribe directamente en la memoria principal
    - Habitual con write-through

Jerarquía de memoria



(hoja de Karim)

Políticas de ubicación

Como hay menos líneas de caché que bloques de memoria principal, se necesita un algoritmo que haga corresponder bloques de memoria principal a líneas de caché.

También se requiere algún medio para determinar qué bloque de memoria principal ocupa actualmente una línea dada de caché. La elección de la función de correspondencia determina cómo se organiza la caché. Pueden usarse tres técnicas:

- Correspondencia directa: Hace corresponder cada bloque de memoria principal a sólo una línea posible de caché. Se usa una parte de la dirección como numero de línea, proporcionando así una correspondencia o asignación única. Es simple y poco costosa de implementar. Desventaja: Hay una posición concreta de caché para cada bloque dado y si un programa referencia repetidas veces a palabras de dos bloques diferentes asignados en la misma línea, los bloques se estarían intercambiando continuamente en la caché y la tasa de aciertos sería baja.
    - N° línea caché = N° bloque ref. MOD N° líneas caché
- Correspondencia (totalmente) asociativa: Permite que cada bloque de memoria principal pueda cargarse en cualquier línea de la caché. La lógica de control de la caché interpreta una dirección de memoria como una etiqueta y un campo de palabra. El campo de etiqueta identifica unívocamente un bloque de memoria principal. En esta técnica hay flexibilidad para que cualquier bloque sea reemplazado cuando se va a escribir uno nuevo en la caché. Desventaja: se necesita una circuitería compleja para examinar en paralelo las etiquetas de todas las líneas de caché (para determinar si un bloque determinado está en la caché) y es costoso en tiempo
- Correspondencia asociativa por conjuntos: Un bloque puede almacenarse en un conjunto restringido de lugares en la caché. Un conjunto es un grupo de líneas en la

caché. Se llama correspondencia asociativa por conjuntos de k vías. N° conjunto = N° bloque ref. MOD N° conjuntos caché.

Modulo 5

Cluster vs SMP

- Ambos:
  - dan soporte a aplicaciones de alta demanda de recursos
  - disponibles comercialmente (SMP es mas antiguo)
- SMP:
  - Mas fácil de administrar y configurar
  - Cercano a los sistemas de un solo procesador
  - La planificación (scheduling) es la diferencia principal
  - Menos espacio físico / Menor consumo de potencia
- Cluster:
  - Superior escalabilidad incremental y absoluta
  - Superior disponibilidad
    - Redundancia

Términos UMA, NUMA, CC-NUMA

Todos los procesadores tienen acceso a toda la memoria: Usan ‘load’ y ‘store’

- UMA - Uniform memory access
  - Igual tiempo de acceso a todas las regiones de memoria
  - Igual tiempo de acceso a memoria para los diferentes procesadores
- NUMA - Nonuniform memory access
  - EL tiempo de acceso de un procesador difiere dependiendo de la región de memoria que accede
  - Diferentes procesadores acceden a diferentes regiones de memoria a diferentes velocidades
- CC-NUMA - cache coherente NUMA
  - Es un NUMA que mantiene coherencia de cache entre las cache de los distintos procesadores

Motivación NUMA

- SMP tiene límite práctico en su número de procesadores: entre 16 y 64 por degradación de prestaciones
- En clusters cada nodo tiene su propia memoria principal
  - Aplicaciones no ‘ven’ la memoria global
  - Coherencia de cache mantenida por software no por hardware
- NUMA retiene las características tipo SMP y brinda multiprocesamiento a gran escala
  - ej. SGI Origin de Silicon Graphics es NUMA con 1024 MIPS R10000

Objetivo NUMA: tener una memoria transparente del sistema y permitir nodos, cada uno con su propio bus o sistema de conexión interna.

Operación CC-NUMA

- Cada procesador tiene cache L1 y L2
- Cada nodo tiene su propia memoria principal
- Nodos conectados por algún tipo de red
- Cada procesador 've' un único espacio de direcciones de memoria
- Orden de acceso a memoria:
    - cache L1 (local al procesador)
    - cache L2 (local al procesador)
    - Memoria principal (local al nodo)
    - Memoria remota
        - Petición por red
- Automático y transparente

Procesamiento Multihebra(Multithreading)

- Aumento de paralelismo de instrucciones
    - Sin el aumento de complejidad y consumo de potencia de la segmentación de cauce y los superescalares
- La secuencia de instrucciones se divide en secuencias más pequeñas llamadas hebras (threads) que pueden ejecutarse en paralelo
- Amplia variedad de diseños multihebra.

Términos: Hebra y Proceso

- Concepto de Hebra en procesadores multihebra puede no ser el de S.O. multiprogramados.
- Proceso
    - Un programa 'corriendo' en una computadora
        - Propiedad de Recursos
            - Espacio de direcciones virtuales para almacenar la imágen de un proceso (code, data, stack, etc)
        - Planificación/ejecución
            - Hay camino de ejecución (traza)
- Conmutación de Proceso (process switch)
- Hebra (thread)
    - Unidad de trabajo de un proceso que puede asignarse
    - Incluye un contexto de procesador (incluido PC y SP) y área de datos para su pila (stack)
    - Se ejecuta secuencialmente.
    - Interrumpible. El procesador cambiaría a otra hebra
- Conmutación de hebra (thread switch)
    - Cambio de control del procesador entre hebras de un mismo proceso
        - Usualmente menos costosa que la conmutación de proceso

Multihebra implícito y explícito

- Multihebra explícito

    - Ejecución concurrente de instrucciones de diferentes hebras explícitas
        - Mezcla de instrucciones de diferentes hebras en cauces compartidos
        - Ó por ejecución paralela en cauces paralelos
    - Todos los procesadores comerciales lo usan
- Multihebra implícito
    - Ejecución concurrente de varias hebras extraídas de un único programa secuencial.
        - Definidas estáticamente por el compilador ó dinámicamente por el hardware

Procesador multihebra

- PC (contador de programa) distinto para cada hebra que pueda ejecutarse concurrentemente.
    - Cantidad y tipo de HW para ejecución concurrente
- Se trata cada hebra separadamente
    - Predicción de saltos, renombre de registros y etc para optimizar ejecución.
        - Paralelismo entre hebras
- Aproximaciones con ejecución simultánea real
    - Multihebra simultánea (SMT) – Pentium 4 HT
    - Multiprocesador monochip

Caché

- Funcionamiento de un cluster
- ¿Qué características describen un cluster de computadoras?

Qué es un cluster?

Un cluster es un grupo de computadores completos interconectados que trabajan conjuntamente como un único recurso de cómputo, creándose la ilusión de que se trata de una sóla máquina. Cada computador del cluster se denomina nodo. Son una alternativa a los multiprocesadores simétricos (SMP) para disponer de prestaciones y disponibilidad elevadas. Son atractivos en aplicaciones propias de un servidor.

Los beneficios del cluster son:

- Escalabilidad absoluta: Es posible configurar clusters grandes que incluso superen las prestaciones de los computadores independientes más potentes.
- Escalabilidad incremental: Se pueden añadir nuevos sistemas al cluster en ampliaciones sucesivas.
- Alta disponibilidad: Como cada nodo del cluster es un computador autónomo, el fallo de uno de ellos no significa la pérdida del servicio
- Mejor relación precio/prestaciones: Usa elementos estandarizados que permiten mayor o igual potencia de cómputo que un computador independiente mayor, a menor costo

Procesadores superescalares

- ¿Qué características definen un procesador como superescalar? Describa las políticas de emisión de instrucciones en un cauce segmentado.

¿Qué es un procesador superescalar?

Un procesador superescalar es aquel que usa múltiples cauces de instrucciones independientes. Cada cauce consta 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.

Un procesador superescalar capta varias instrucciones a la vez y a continuación, intenta encontrar instrucciones cercanas que sean independientes entre sí y puedan, por consiguiente, 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, por lo que al identificar estas dependencias, el procesador puede emitir y completar instrucciones en un orden diferente al del código máquina original.

El procesador puede eliminar algunas dependencias innecesarias mediante el uso de registros adicionales y el renombramiento de las referencias a registros en el código original. Para maximizar la utilización del cauce de instrucciones y aumentar el rendimiento no usa saltos retardados (como los RISC) sino que hacen predicción de saltos.

Las instrucciones comunes (aritmética entera y de coma flotante, cargas, almacenamientos y bifurcaciones condicionales) pueden iniciar su ejecución simultáneamente y ejecutarse de manera independiente (en diferentes cauces).

El enfoque superescalar conlleva a la duplicación de algunas o todas las partes de la CPU/ALU:

- Captar múltiples instrucciones al mismo tiempo
- Ejecutar sumas y multiplicaciones simultáneamente
- Ejecutar carga/almacenamiento, mientras se lleva a cabo una operación en ALU

El grado de paralelismo y la aceleración de la máquina aumentan, ya que se ejecutan más instrucciones en paralelo.

Políticas de emisión de instrucciones:

La emisión de instrucciones es el proceso de iniciar la ejecución de instrucciones en las unidades funcionales del procesador. Y la política de emisión de instrucciones es el protocolo usado para emitir instrucciones.

El procesador intenta localizar instrucciones más allá del punto de ejecución en curso, que puedan introducirse en el cauce y ejecutarse. Son importantes 3 ordenaciones:

- Orden en que se captan las instrucciones
- Orden en que se ejecutan las instrucciones
- Orden en que las instrucciones actualizan los contenidos de los registros y de las posiciones de memoria

Para optimizar la utilización de los diversos elementos del cauce, el procesador tendrá que alterar uno o más de estos órdenes con respecto al orden que se encontraría en una ejecución secuencial estricta. La única restricción del procesador es que el resultado sea correcto.

Las políticas pueden ser de tres categorías:

- Emisión en orden y finalización en orden

La más sencilla. Emitir instrucciones en el orden exacto en que lo haría una ejecución secuencial y escribir los resultados en ese mismo orden.

- Emisión en orden y finalización desordenada

La finalización desordenada se usan 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.

La emisión de instrucciones se para cuando hay una pugna por un recurso, una dependencia de datos o una dependencia relativa al procedimiento.

Aparece la dependencia de salida o dependencia escritura-escritura.

La finalización desordenada necesita una lógica de emisión de instrucciones más compleja que la finalización en orden. Además, es más difícil ocuparse de las interrupciones y excepciones.

- Emisión desordenada y finalización desordenada

Con la emisión en orden, el procesador solo decodificara instrucciones hasta que haya conflicto o dependencia, no puede ir más allá del punto del conflicto buscando instrucciones que podrían ser independientes (hasta que el mismo no se resuelva) y que podrían introducirse provechosamente en el cauce

Para permitir la emisión desordenada, es necesario desacoplar las etapas del cauce 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, puede continuar captando y decodificando nuevas instrucciones. 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. Cualquier instrucción puede emitirse, siempre que: necesite la unidad funcional particular que esté disponible y que ningún conflicto ni dependencia la bloqueen.

Así el procesador tiene capacidad de anticipación, permitiendo identificar las instrucciones independientes que pueden introducirse en la etapa de ejecución. Las instrucciones se emiten desde la ventana de instrucciones sin que se tenga en cuenta su orden original en el programa. Aparece una nueva dependencia: la antidependencia o dependencia lectura-escritura, donde en lugar de que la 1ra instrucción produzca un valor que usa la 2da, la 2da instrucción DESTRUYE un valor que usa la 1ra

SMP

- ¿Qué características posee un multiprocesador simétrico (SMP)?

Qué es un SMP?

Características de un computador SMP:

- Hay dos o más procesador similares de capacidades comparables
- 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 (UMA).
- Todos los procesadores comparten los dispositivos de E/S, bien a través de los mismos canales, o bien mediante canales distintos que proporcionan caminos de acceso al mismo dispositivo.
- Todos los procesadores pueden desempeñar las mismas funciones (de ahí el término simétrico)
- El sistema está controlado por un sistema operativo integrado, que proporciona la interacción entre los procesadores y sus programas en los niveles de trabajo, tarea, fichero, y datos

Ventajas potenciales de un SMP con respecto a una arquitectura monoprocesador:

- Mayores prestaciones: Si el trabajo a realizar puede organizarse en paralelo, con varios procesadores será mejor que con uno solo.
- Buena disponibilidad: Un fallo en un procesador no detendrá la computadora, ya que todos los procesadores pueden realizar las mismas funciones
- Crecimiento incremental: Se pueden añadir más procesadores
- Escalado: En función de la cantidad de procesadores de un sistema, hay una gama diferente de productos con prestaciones y precios diferentes

La existencia de varios procesadores es transparente al usuario, el SO es quien sincroniza los procesadores y planifica los hilos o procesos, asignándolos a los distintos procesadores.

CUIDADO: Bus compartido

Taxonomía de Flynn

- ¿Cuáles son las arquitecturas que pueden encontrarse en la configuración MIMD de la taxonomía de Flynn?

¿Qué es la Taxonomía de Flynn?

La taxonomía de Flynn clasifica a los sistemas de varios procesadores según sus capacidades de procesamiento paralelo. Una de las categorías es MIMD: Múltiples secuencias de instrucción, múltiples secuencias de datos.

Significa que un conjunto de procesadores ejecuta simultáneamente secuencias de instrucciones diferentes con conjuntos de datos diferentes, Los SMP, los “clusters” y los sistemas NUMA son ejemplos de esta categoría.

En la organización MIMD, los procesadores son de uso general: cada uno es capaz de procesar todas las instrucciones necesarias para realizar las transformaciones apropiadas de los datos.

Los computadores MIMD se pueden dividir según la forma que tienen los procesadores para comunicarse:

- Memoria compartida (fuertemente acoplada): Los procesadores comparten una memoria común, entonces cada procesador accede a los programas y datos almacenados en la

memoria compartida y los procesadores se comunican unos con otros a través de esa memoria.
    - Multiprocesador simétrico (SMP): Varios procesadores comparten una única memoria mediante un bus compartido u otro tipo de mecanismo de interconexión. El tiempo de acceso a memoria principal es aproximadamente el mismo para cualquier procesador.
    - Acceso no uniforme a memoria (NUMA): El tiempo de acceso a zonas de memoria diferentes puede diferir.
- Memoria distribuida (débilmente acoplada):
    - Clusters: Conjunto de computadores monoprocesadores independientes, o de SMP, que se interconectan. La comunicación entre los computadores es mediante conexiones fijas o mediante algún tipo de red.

(Ver comparación de SMP y Clusters, ademas del termino NUMA (en preguntas anteriores y en la clase 09))