Arquitectura de computadoras

Preguntas de finales

Fuentes

- Organización y Arquitectura de Computadoras, William Stallings
- Apuntes de clase
- Manual MSX88
- “resumenarqui.pdf” Autor: ns/nc
- “Finales arq Alfonso _resueltos” Autor: ns/nc

Autor

- Karim Emilio (karimemilio98@gmail.com)

Notas:

- 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? 2014/09
- Explique el mecanismo de interrupción. Describa las distintas fuentes de interrupción que conozca y el tratamiento a realizar cuando hay múltiples interrupciones. 2014/08/06
- Mecanismo de interrupción. Fuentes de interrupción y tratamiento de interrupciones múltiples. 2014/02/26
- Describa las características que diferencian los SMTP respecto a los clusters 2014/02
- Describa los elementos a tener en cuenta en el diseño de una memoria caché. Anilice ventajas y desventajas de poseer varios niveles de caché.2013/03/26 y 2012/03 - 2do
- Describir el mecanismo de interrupción. Mencionar cuales son las fuentes de interrupción (tipos de interrupción). Describir el tratamiento de múltiples interrupciones. 2013/02/27

Unidad 1 : Arquitectura y Organización de Computadoras

Concepto de Arquitectura. Relación con Organización de Computadoras. Repaso del modelo de von Neumann. Descripción del funcionamiento de un sistema basado en un microprocesador. Buses, teoría de operación, buses sincrónicos y asincrónicos. Ejemplos. Repaso de ejecución de instrucciones. Ejecución solapada ("pipeline"). Su aplicación en procesadores contemporáneos. Análisis de prestaciones. Arquitecturas reconfigurables: conceptos. Sistemas embebidos: conceptos.

1. **¿Qué es una pila? Describir el comportamiento con anidamiento de múltiples procedimientos/funciones utilizando pila.** (Abril 2015)

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 de la pila (conocida como lista LIFO).

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

La pila es el método más usado para el pasaje de parámetros, considerado el "verdadero pasaje de parámetros". Es independiente de memoria y registros. Es usada tanto por el usuario como por el sistema.

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 del tope o cabecera de pila. Si se añade o elimina un elemento de la pila el mismo incrementa o decrementa para actualizarse.
- Base de la pila: 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 POP con la pila llena, se informa un error.

El comportamiento con anidamiento de procedimientos/funciones/subrutinas es el siguiente:

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:

2. Salvar el estado de BP (viejo BP): apilar el valor del BP
3. Salvar el estado de SP (BP=SP): cargar el valor del BP el del SP
4. Reservar espacio para datos locales (opcional): si es que los hay
5. Salvar valores de otros registros (opcional): si es que se van a modificar los registros
6. 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.
    a. 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
7. Escribir sentencias a ejecutar: Aqui 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:
8. Retornar parámetro (opcional): si es que tiene que retornar datos
9. 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

Apilo las direcciones de retorno, tantas como llamadas anidadas haya y voy sacando en orden inverso al que fueron guardadas (POP, LIFO).

Fuente: "Clase 01" y "Apéndice 9A - Pilas" (Stalling 5ta ed. Pág 353)

2. **¿Qué es la segmentación de cauce? Explique los atascos producidos por saltos** (Abril 2015)
    a. 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. (Septiembre 2014)
    b. ¿Qué entiende por segmentación de cauce? ¿Qué ventajas proporciona su implementación? (Agosto 2014 #06) (Septiembre 2013)
    c. Segmentacion de cauce. Describir atascos por dependencia de datos y su solucion. (Marzo 2012 2do)

La segmentación de cauce supone dividir el ciclo de instrucción en varias etapas separadas que operan secuencialmente, tales como captación de instrucción, decodificación de instrucción y escritura del operando resultado. Las instrucciones se mueven a través de estas etapas como en una cadena de montaje, de modo que, en principio, cada etapa puede estar trabajando en una instrucción diferente al mismo tiempo.

La segmentación de instrucciones es similar al uso de una cadena de montaje en una fábrica, donde se trabaja sobre los productos en varias etapas simultáneamente. Se llama segmentación de cauce o pipelining, 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).

Una instrucción tiene varias etapas que tienen lugar secuencialmente. Si consideramos el procesamiento de una instrucción como dos etapas, 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. El cauce tiene dos etapas independientes: la primera etapa capta una instrucción y la almacena en un buffer. Cuando la segunda etapa está libre, la primera le pasa la instrucción almacenada. 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).

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.

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

Para conseguir mayor aceleración, el cauce debe partir las instrucciones en más etapas de duración similar.

Atascos producidos por saltos

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
- 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

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.

Si el salto es:

- 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.

Fuente: “Clase 04”, “Clase 05” y “Capítulo 11.4 - Segmentación de instrucciones” (Stalling 5ta ed. Pág 405)

3. **¿Qué es la segmentación de cauce? Describir técnicas para el tratamiento de saltos condicionales (Diciembre 2014 #03)**
    a. **¿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. (Noviembre 2014)**

En la instrucción de bifurcación condicional, hasta que la misma no se ejecuta es imposible determinar si el salto se producirá o no.

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).

Fuente: “Clase 05” y “Capítulo 11.4 - Segmentación de instrucciones - Tratamiento de saltos ” (Stalling 5ta ed. Pág 411)

4. **Explique los métodos de pasaje de argumentos a procedimientos o funciones. Describa el funcionamiento y uso de la pila. (Noviembre 2014)**
    a. **Explique los métodos de pasaje de argumentos a procedimientos o funciones. Describa el comportamiento con anidamiento de múltiples procedimientos/funciones. (Marzo 2013 #26) (Marzo 2012 2da fecha)**

Pasaje de argumentos/parámetros a subrutinas/procedimientos/funciones:

- 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

Fuente: “Clase 01” y “Capítulo 12 - Computadores de repertorio reducido de instrucciones ” (Stalling 5ta ed. Pág 443)

5. **¿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** (Octubre 2014 #22) (Febrero 2014)
    a. Describa tres (3) diferentes causas o motivos que pueden retardar un cauce de instrucciones segmentado (Agosto 2014 #06)
    b. Qué es la segmentación de cauce. Tres motivos de retardo de cauce. (Febrero 2014 #26)
    c. ¿Qué es la segmentación del cauce de instrucción? ¿Cuánto mejora el rendimiento? Describa las dependencias de los datos que puedan afectar un cauce segmentado (Marzo 2013 #26) (Febrero 2013 #27)

d. ¿Que es la segmentación del cauce? Describa tipos de dependencias que afectan el funcionamiento de los causes y las posibles soluciones para evitarlos. (Marzo 2012 1ro)

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

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

- Por dependencia de datos: Dos instrucciones se comunican por medio de un dato

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

Hay dos 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 (X_i) se lleva a la entrada de la etapa correspondiente (X_{i+1}) sin esperar a la escritura (M_i o W_i)

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”.

- Por dependencia de control: La ejecución de una instrucción depende de cómo se ejecute otra. Existe una penalización por salto.

Instrucciones de salto:

-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)

VER SOLUCIONES A RIESGOS DE CONTROL MÁS ARRIBA
Fuente: “Clase 05”

Unidad 2: Subsistema Unidad Central de Procesos

Repaso de máquinas que ejecutan instrucciones. Ejemplificación en procesadores típicos: IA32. Análisis del conjunto de instrucciones de procesadores de uso comercial. Concepto de máquinas CISC y RISC. Lineamientos básicos en el diseño de un procesador RISC. Análisis de prestaciones. Ejemplos: procesadores MIPS y ARM. Interrupciones: tratamiento general. Interrupciones por software y por hardware, vectores, descripción y tratamiento particular de cada una. Relación entre las interrupciones y el manejo de operaciones de E/S.

1. **Finalidad de las interrupciones. Para que se utiliza un controlador de interrupciones.** (Diciembre 2014 #03)

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

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 la 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.

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

-1ro: Suspende la ejecución del programa en curso y guarda su contexto (el contenido actual del PC y demás)

-2do: Carga el PC con la dirección de comienzo de una rutina de gestión de interrupción

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
- 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:

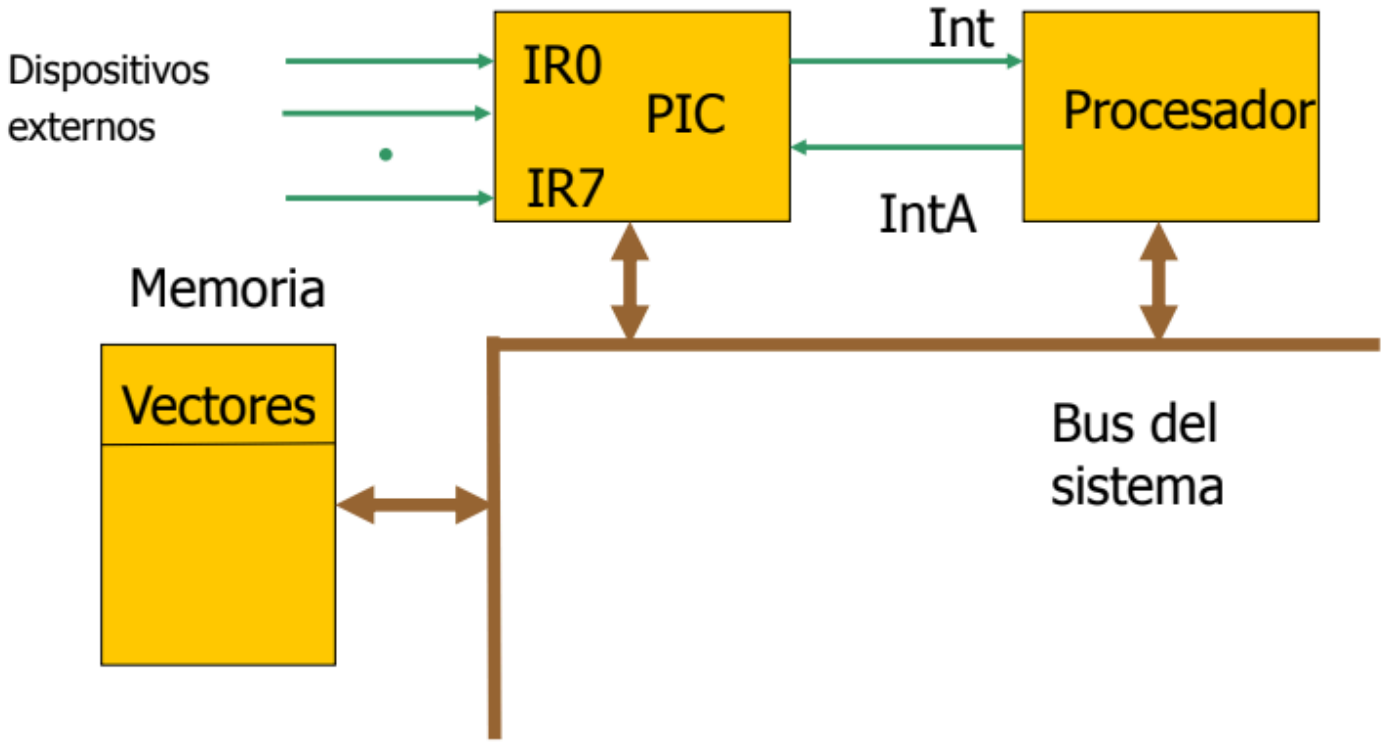
- Programa: Por el resultado de una ejecución de una instrucción (Ej: Overflow, división por 0, intento de ejecución de una instrucción inexistente, intento de acceso fuera del espacio de memoria permitido para el usuario)
- Temporización: Clock interno del procesador que permite al SO realizar ciertas 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:

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

Fuente: "Clase 02" y "Capítulo 3.2 - Funcionamiento del computador - Interrupciones" (Stalling 5ta ed. Pág 58)
Fuente: "7.4 - Controlador de interrupciones (PIC)" (Manual MSX88, Pág 35)
Fuente: "Interrupciones en la Arq IA32" (PDF complementario)

2. **Describa las características que diferencian a los procesadores RISC respecto de los CISC. (Diciembre 2014 #03) (Octubre 2014 #22) (Marzo 2012 1ro)**

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

- Repertorio de instrucciones limitado y con un formato fijo
- 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.

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

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

¿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.

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

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

RISC frente a CISC

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

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

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

Fuente: “Clase 06” y “Capítulo 12 - Computadores de repertorio reducido de instrucciones” (Stalling 5ta ed. Págs 438, 451, 455 y 473)

3. **¿Qué es una interrupción? Describa cómo funcionan. ¿Cómo se utiliza un controlador de interrupción? (Noviembre 2014)**
    a. Explique el mecanismo de interrupción. Describa las caraterísticas y el funcionamiento de un pic. (Septiembre 2013)

b. ¿Cómo es la estructura de un módulo de E/S? Describa las características del controlador de interrpciones PIC. (Marzo 2013 #26)

Las interrupciones 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.
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 (“Interrupciones en la Arq IA32”).

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
Fuente: “Interrupciones en la Arq IA32” (PDF complementario)
Fuente: “Clase 02” y “Capítulo 3.2 - Funcionamiento del computador - Interrupciones” (Stalling 5ta ed. Pág 60)
Fuente: “7.4 - Controlador de interrupciones (PIC)” (Manual MSX88, Pág 35)

4.

Unidad 3 : Subsistema E/S

Concepto de E/S y su relación con la CPU, tipos de puertas. Concepto de puerta de Entrada y Salida paralelo. Concepto de puerta de Entrada y Salida serie. Tipos de transmisión serie. Descripción del formato de transmisión serie asincrónica y sincrónica. Descripción funcional de una puerta de E/S serie asincrónica, acceso a registros internos para control y determinación del estado de operación de la puerta. Mapeado del subsistema E/S y la memoria. Administración de las puertas por encuesta (polling) o por interrupción. Tratamiento de la CPU de las operaciones de E/S, por interrupción o por software. Transferencias de E/S por hardware, DMA, implementación.

1. **¿Cómo funciona un módulo de E/S? Describa las características fundamentales de un DMA.** (Abril 2015)
    a. Describir la estructura de un módulo de E/S. ¿Qué es DMA y cómo funciona? (Diciembre 2014 #03) (Febrero 2013 #27)
    b. ¿Cómo es la estructura interna de un módulo de E/S? Describa las características funcionales del acceso directo a memoria - DMA. (Agosto 2014 #06)
    c. Estructura interna del módulo de E/S. Características funcionales de DMA. (Febrero 2014 #26)
    d. Describa las características fundamentales de un DMA (septiembre 2013)
    e. Estructura de un modulo E/S. Describa el funcionamiento de un controlador DMA(las etapas de transferencia) (Marzo 2012 2da fecha)

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.

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). 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 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.

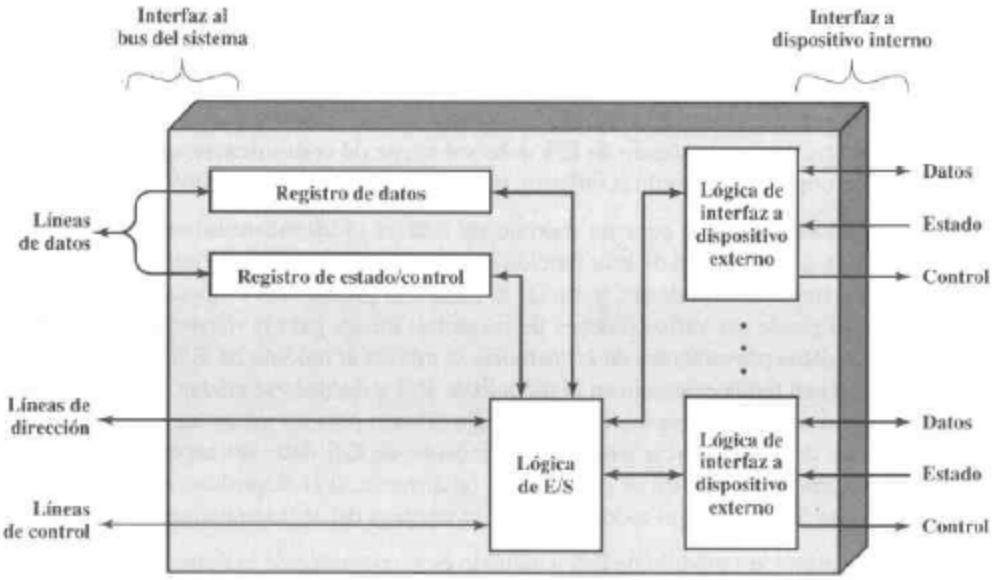
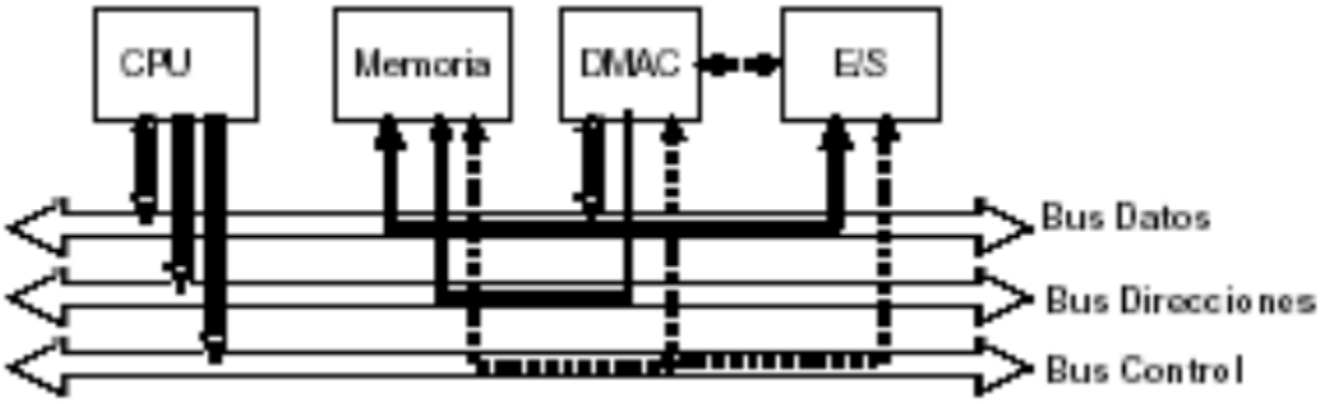


Figura 7.3. Diagrama de bloques de un módulo de E/S.

Funcionamiento del DMA

El DMA (Direct Memory Access) requiere un módulo adicional en el bus del sistema. El módulo 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, el DMA debe usar el bus sólo cuando el procesador no lo necesita o debe forzar al procesador a que suspenda temporalmente su funcionamiento (técnica denominada 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.



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.

Fuente: "Clase 03" y "Capítulo 6.2 - Módulo de E/S" (Stalling 5ta ed. Pág 180)
Fuente: "Clase 03" y "Capítulo 6.5 - Acceso directo a memoria - Funcionamiento del DMA" (Stalling 5ta ed. Pág 195)

2. **¿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**. (Octubre 2014 #22) (Febrero 2014) (Marzo 2012 1ro)

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 volumenes 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) (AMPLIACIÓN EN PREGUNTA #1)

Fuente: “Clase 03” y “Capítulo 6.2 - Módulo de E/S” (Stalling 5ta ed. Pág 183, 186, 195)

Unidad 4 : Subsistema Memoria

Repaso de la organización jerárquica de la memoria, memoria principal y memoria secundaria. Memoria caché, concepto y descripción, análisis de prestaciones, métodos de implementación típicos, múltiples niveles. Ejemplos. Conceptos de memoria virtual.

1. **Justifique el uso de dos niveles de caché.** (Abril 2015)

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 cache.

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.

Fuente: “Clase 07” y “Capítulo 4.3 - Memoria caché” (Stalling 5ta ed. Pág 111)

2. **Caché: mencione algoritmos de reemplazo y políticas de escritura. (Diciembre 2014 #03)**

    a. Describa las funciones de correspondencia entre memoria principal y cache. Analice las politicas de escritura desde el punto de vista de la coherencia de datos. (Agosto 2014 #06) (Febrero 2014 #26) (Septiembre 2013)
    b. Mencionar los tipos de correspondencia de la memoria caché. Describir las políticas de escritura (en acierto y en fallo) (Febrero 2013 #27)

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

Cuando se introduce un nuevo bloque en la cache, 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

Fuente: "Clase 07" y "Capítulo 4.3 - Memoria caché" (Stalling 5ta ed. Pág 121)

3. **¿Por qué funciona una jerarquía de memoria? Describa las políticas de ubicación y de reemplazo de bloques en memoria caché.** (Noviembre 2014)

    a. Describa las técnicas de ubicación de bloques y las politicas de escritura en Cache. (Octubre 2014 #22) (Marzo 2012 1ro)

    b. 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. (Septiembre 2014)

    c. Describa las técnicas de reemplazo de bloque, correspondencia y políticas de escritura en memoria cache. (Febrero 2014)

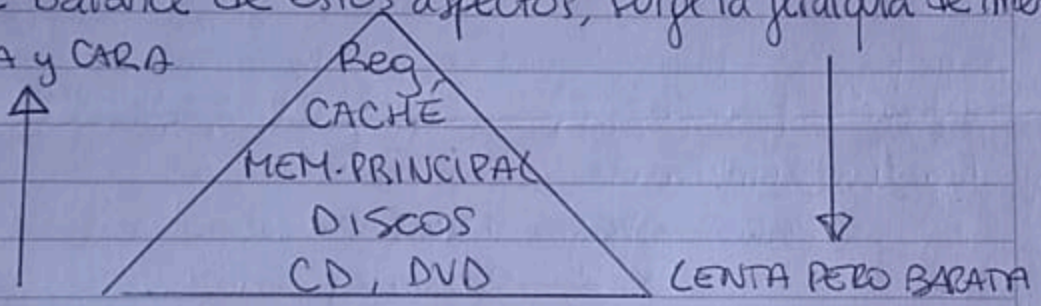
Hay un compromiso entre las 3 características clave de una memoria:

- Capacidad (cuanto mayor sea, mejor)
- Rapidez (la memoria debe seguir al procesador: cuando esta ejecuta instrucciones, no es deseable que tenga que detenerse a la espera de instrucciones u operandos.
- Coste (debe ser razonable en relación a otros componentes)

se sabe que:

- A menor tiempo de acceso, mayor coste x bit
- A mayor capacidad, menor coste x bit
- A mayor capacidad, mayor tiempo de acceso

A modo de "balance" de estos aspectos, surge la jerarquía de memorias:



A medida que descendemos en la jerarquía:

- disminuye el coste x bit
- aumenta la capacidad
- aumenta el tiempo de acceso
- disminuye la frecuencia de accesos a memoria por parte del procesador

Las distintas memorias/niveles se complementan y la estrategia es organizar los datos y programas en memoria para que las palabras necesarias esten normalmente en la memoria mas rapida (donde se accede con más frecuencia) y así disminuir la frecuencia de acceso a memorias grandes y lentas.

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 de 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 numerosos 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.

Fuente: "Clase 07" y "Capítulo 4.3 - Memoria caché" (Stalling 5ta ed. Pág 115) (+ material de Organización de Computadoras)

4. **¿Que es un Bus? Describa los diferentes tipos, métodos de arbitraje y técnicas de sincronización. Mencione las principales diferencias entre PCI y SCSI.** (Octubre 2014 #22) (Marzo 2012 1ro)

    a. Que es un Bus, tipos de buses, temporización y métodos de arbitraje (Febrero 2014)

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

Fuente: “Anexo Clase 07” y “Capítulo 3.4 - Interconexión con buses” (Stalling 5ta ed. Pág 72 y 76)

Unidad 5 : Paralelismo y mejora de prestaciones

Concepto de procesamiento paralelo. Paralelismo a nivel instrucción. Procesadores superescalares. Ejemplos. Clasificación de arquitecturas paralelo: taxonomía de Flynn. Ejemplos de aplicación. Arquitecturas Multiprocesador. Memoria compartida o distribuida. Análisis de prestaciones.

1. **Funcionamiento de un cluster** (Abril 2015)

    a. ¿Qué características describen un cluster de computadoras? (Agosto 2014 #06) (Febrero 2014 #26) (Septiembre de 2013)

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

Fuente: “Clase 09” y “Capítulo 16.4 - Clusters” (Stalling 5ta ed. Pág 619)

2. **¿Qué características posee un multiprocesador simétrico (SMP)?** (Noviembre 2014)

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

Fuente: “Clase 09” y “Capítulo 16.2 - Multiprocesadores simétricos” (Stalling 5ta ed. Pág 604)

3. **¿Qué características definen un procesador como superescalar? Describa las políticas de emisión de instrucciones en un cauce segmentado.** (Septiembre 2014) (Marzo 2013 #26) (Febrero 2013 #27) (Marzo 2012 2do)

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
Fuente: "Clase 08" y "Capítulo 13 - Procesadores superescalares" (Stalling 5ta ed. Pág 480 y 487)

4. **¿Cuáles son las arquitecturas que pueden encontrarse en la configuración MIMD de la taxonomía de Flynn?** (Septiembre 2014)

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))
Fuente: "Clase 09" y "Capítulo 16.1 - Organizaciones con varios procesadores" (Stalling 5ta ed. Pág 601)