Arquitectura de computadoras UNLP

Unidad 1 - Repaso general

`Filmina 1, esta unidad es de repaso de los temas de organización de computadoras.`

- ¿Cuál es la diferencia entre programación en hardware y software?

  En el pasado se utilizaba la programación mediante **hardware**, por lo tanto al cambiar de hardware había que cambiar de programa también afectando a su lógica y funcionamiento, lo que hacía muy difícil mover un programa de una computadora a otra.

  Los datos de entrada pasan por una secuencia de funciones aritméticas - lógicas y producen resultados.

  En la actualidad se realiza programación sobre **software**, por lo cual los programas son más fáciles de mover entre computadoras, existe un intérprete que traduce las instrucciones de un lenguaje de alto nivel a uno de lenguaje máquina.

- ¿Qué es la arquitectura Von Neuman?

  Esta arquitectura está compuesta por una Unidad central de procesamiento o CPU, una memoria principal, un controlador de entrada y salida, y los buses del sistema, que son tres el de datos, el de direcciónes y el de control. A su vez la cpu

está formada por la unidad de control y la unidad aritmética lógica o ALU. Para que esta arquitectura funcione se necesitan datos para suministrar al sistema y una memoria para almacenar esos datos, con estos se produce un resultado que son enviados al exterior mediante componentes de entrada/salida.

▼ ¿Qué es el repertorio de instrucciones?

Es el conjunto completo de instrucciones que se realizan en la cpu, mediante estas instrucciones el programador puede controlar la cpu. Las operaciones usan un código simbólico para abstraer al programador de tener que escribir las operaciones en binario.

Ejemplo : ADD (sumar), SUB(Restar).

▼ ¿Qué elementos tiene una instrucción?

- Tiene un código de operación que especifica la operación a realizar, la operación se indica por medio de un código binario llamado codop de manera abreviada
- Tiene una referencia a operando fuentes, que son los datos sobre los cuales se va a operar, son los datos de entrada para la instrucción.
- Referencia al operando destino o resultado, la operación produce un resultado y se guarda en esa dirección.
- Referencia a la siguiente instrucción que le dice al procesador de donde captarse la siguiente instrucción tras finalizar la instrucción actual.

Los operandos de las operaciones pueden almacenarse en distintas ubicaciones, puede ser la memoria principal, memoria virtual o caché. En los registros de la cpu o en dispositivos de entrada y salida, como pueden ser los discos externos.

▼ ¿Que tipos de instrucciones existen?

Existen cuatro tipos de instrucciones

Procesamiento de datos, son las operaciones que involucran la aritmética y la lógica.

Almacenamiento de datos.

Transferencia de datos, son las instrucciones que realizan las entradas y salidas.

Control, son las instrucciones encargadas de controlar el flujo del programa y realizar el testo.

▼ ¿Que son las direcciones de instrucción?

Las direcciones que tiene disponible una instrucción pueden variar dependiendo del diseño de la misma.

Más direcciones por instrucción implica que existirán instrucciones más complejas, que habrá más registros y un programa tendrá menos instrucciones, por qué podría hacer más operaciones con menos instrucciones.

Tener menos direcciones por instrucción implica tener instrucciones menos complejas, programas más largos, porque tiene que usar más instrucciones para realizar determinadas operaciones. Pero la captación y ejecución de programas es más rápida.

▼ ¿Qué alternativas de almacenamiento existen?

- Almacenamiento de tipo pila es cuando se usa una estructura especial conocida como pila, para almacenar los datos
- Almacenamiento de tipo acumulador, se utiliza un registro para almacenar un dato.
- Almacenamiento de tipo memoria-memoria, se utilizan dos datos de la memoria como entrada y se produce una salida hacia otra dirección de memoria.
- Almacenamiento de tipo registro-registro, se operan con entradas obtenidas desde los registros y luego se lo almacena en otra dirección de registro.

▼ ¿Qué se debe tener en cuenta al diseñar un repertorio de instrucciones?

- **Repertorio de operaciones**, cuántas y qué operaciones considerar y cuán complejas van a ser estas operaciones
- **Los tipos de datos,** se definen los distintos tipos de datos sobre los cuales se podrán realizar operaciones.
- **El formato de instrucción**, es la longitud que tiene la instrucción en bits, números de direcciones, tamaño de los distintos campos, etc.
- **Los registros**, son la cantidad de registros que pueden ser referenciados por la cpu y su uso.

- **Los modos de direccionamiento**, que es la forma en la que se puede traer a los operandos.

▼ ¿Que tipos de repertorios existen?

El **Risc** y el **Cisc**.

El Risc es una filosofía de diseño de cpu que está a favor de conjunto de instrucciones pequeñas y simples que toman menor tiempo para ejecutarse.

El Cisc que tiene operaciones más complejas y grandes.

▼ ¿Cómo pueden estar ordenados los bits de una palabra?

Mismos datos pueden ser diferentes por la técnica de ordenamiento que se use

El **Big Endian** el byte más significativo en la dirección con valor numérico más bajo

El **Little Endian** guarda el byte menos significativo en la dirección con valor numérico más alto.

▼ ¿Qué formatos de instrucción existen?

Instrucción tipo I

Instrucción tipo R

Instrucción tipo J

▼ ¿Qué modos de direccionamiento hay?

Modos de direccionamiento

Inmediato

Dirección de memoria o absoluto

Dirección de registro

Indirecto de memoria

Indirecto con registro

Indirecto con desplazamiento (Basado, indexado o relativo al pc) o mediante pila

▼ Mencione y describa tipos de operaciones

T**ipos de operaciones:**

- Transferencia de datos, es mover un dato de dirección, para ello se debe especificar la ubicación de los operandos fuentes, las de los operando destino, el tamaño de los datos a ser transferido y el modo de direccionamiento. Existen diferentes tipos de transferencias y cada una tiene su propia instrucción, por ejemplo memoria a memoria o registro a registro o registro a memoria. etc.
- Aritméticas son las operaciones básicas como la suma, la resta, la multiplicación y la división. También en algunos casos se encuentran instrucciones extras como el incremento, la negación, el absoluto o el shift de derecha o izquierda.
- Lógicas son las operaciones de manipulación de bits individualmente, como las operaciones Booleanas, que incluye por ejemplo el and el or o el xor entre otros. También está compuesta por operaciones como el rotate left o right.
- Conversión son las operaciones para cambiar de binario a decimal.
- Entrada/salida son pocas instrucciones como el in, out o move, se puede realizar a través de un dispositivo aparte como el dma
- Control de flujo sirven para controlar los caminos que toma el programa, lo hace modificando el registro de program counter por ejemplo el salto incondicional, el salto condicional el salto con retorno o llamado a subrutina
- Control del sistema

▼ Describa como es el ciclo de instrucción básico

Un ciclo de instrucción está compuesto por dos etapas, en la primera que es la de captación se busca la siguiente instrucción a ejecutar, su dirección se encuentra en el registro contador de programa, luego la unidad de control capta la instrucción desde la memoria y la almacena en el registro de instrucción a continuación el registro contador de programa se incrementa y la unidad de control interpreta la instrucción captada y lleva a cabo la acción requerida.

El paso siguiente es el ciclo de ejecución, en el cual pueden ocurrir diferentes cosas dependiendo de la naturaleza de la instrucción, puede ser que se necesiten transferir datos desde o hacia el cpu desde la memoria o desde el exterior, luego puede que se operan sobre esos datos con operaciones lógicas o aritméticas, o podría ser una operación de control que involucre un salto condicional o incondicional.

▼ ¿Cuales son los estados en un ciclo de instrucción?

Primero se calcula la dirección de la siguiente instrucción, luego se capta esta instrucción llevándola al IR, a continuación se decodifica el código de operación y se determina la acción a realizar. Se calcula la dirección del operando, este paso se puede repetir si la operación involucra varios operandos, se realiza la operación con estos datos. Una vez terminado el cálculo se busca la dirección del operando resultado y se lo almacena, pueden ser varios resultados por lo tanto este paso se puede repetir. Luego, en el caso de que se trate de un vector se calcula la dirección del siguiente operando, en el caso contrario la instrucción se encuentra completada y se vuelve al paso uno, en donde se calcula dirección de la siguiente instrucción.

▼ ¿Qué es una subrutina?

Son una gran innovación para los programas informáticos porque brindan ventajas, como puede ser la reutilización del código y la modularización, se pueden llamar desde cualquier parte del programa usando la instrucción call y se les puede pasar parámetros mediante referencia, pasándole la dirección de la variable o mediante valor enviándole una copia de la variable. El pasaje de argumentos a las subrutinas se puede hacer utilizando registros, utilizando memoria o mediante el stack/pila.

▼ ¿Qué es una pila?

La pila es una estructura que guarda el operando de manera implícita en el tope de la misma, necesita un registro que guarde la dirección del tope. La pila tiene dos operaciones principales, push y pop. para apilar y desapilar.

▼ Describa los pasos para llamar a un procedimiento

Lo primero es salvar el estado del BP, luego se salva el estado del SP, opcionalmente se reserva el espacio para los datos locales y salvar valores de otros registros. Acceder a los parámetros del procedimiento, escribir la sentencia a ejecutar, luego opcionalmente retornar un parámetro y finalmente regresar al procedimiento.

En el primer paso, se establece a BP como puntero de referencia y es usado para acceder a los parámetros y datos locales de la pila. El sp no puede ser usado para este propósito por que no es un registro base o índice. El valor de SP puede cambiar pero BP guarda el mismo valor.

- push BP
- mov BP, SP

Luego de manera opcional se reserva espacio para las variables locales al procedimiento, se hace decrementando el SP en la cantidad de bytes necesarios, Ejemplo : sub SP, 2. En este caso se usan 2 bytes.

De manera opcional también se pueden salvar los valores de un registro, puede que el procedimiento haga uso de determinados registros por lo tanto puede cambiar el valor original de estos haciendo que se pierdan, para prevenir esto, se le hace push a los registros que vayan a ser utilizados por el procedimiento, en el caso de que no use ninguno no se hace este paso. push DI.

En el siguiente paso se accede a los parámetros, para lo cual se debe calcular cuál es la posición en la que se encuentra, la cual es 2 + el tamaño de dirección de retorno + el total de tamaño de los parámetros buscados y BP. Por ejemplo para acceder al parámetro 1, debe ser mov CX, [BP + 8]

Para la salida del procedimiento, primero se debe ir desfilando en el mismo orden, todos los registros salvados. Se debe reponer a SP con el valor guardado dentro de BP y luego usar ret para retornar. Por ejemplo:

- pop DI
- mov SP, BP
- pop BP
- ret

Unidad 2 - interrupciones

`Clase 2 Filminas, hojas 66 a 75 del libro Stallings.`

▼ ¿Qué es una interrupción?

Las interrupciones son un mecanismo para pausar brevemente la ejecución de un programa, ejecutar una subrutina especial, y luego continuar ejecutando el programa.

Desde el punto de vista del usuario o programador, una interrupción no es más que una interrupción del ciclo normal de ejecución de un programa.

▼ ¿Cómo se interrumpe?

Cuando ocurre una interrupción, en la mayoría de los casos, se transfiere el control a otro programa, un gestor, que se encarga de gestionar la interrupción siguiendo una serie de estados

Lo primero es guardar el estado actual del procesador, corrige lo que causó la interrupción, restaura el estado original del procesador y finalmente retorna a la ejecución normal del programa. Luego se tiene que comprobar si es momento de salir de la interrupción o repetir el proceso.

▼ ¿Qué puede causar una interrupción?

- Por el resultado de la ejecución de una instrucción, como puede ser la división por cero, el desbordamiento aritmético, entre otros.
- Por un temporizador interno del procesador que le permite al sistema operativo realizar ciertas funciones de manera regular.
- Por una operación de entrada/salida, como la indicación de la finalización de una operación.
- Por un fallo de hardware, pérdida de energía, fallo en la paridad de datos, etc.

▼ ¿Por que existen las interrupciones?

Las interrupciones son un mecanismo para mejorar el rendimiento del procesador, dado que el procesador se comunica con otros dispositivos de menor velocidad (como pueden ser teclados, mouse, impresoras, etc.), si el procesador debe permanecer ocioso a la espera del segundo dispositivo esto significa una pérdida en velocidad.

▼ ¿Cómo es la jerarquía de interrupciones?

Existen dos tipos, las interrupciones no enmascarables que son las que no pueden ser ignoradas por que representan un alto nivel de prioridad desatadas por un evento peligroso, el procesador la recibe por el pin NMI. En cambio las enmascarables enmascarables pueden ser ignoradas mediante instrucciones, el procesador las recibe por el pin INTR, se pueden ignorar estas interrupciones si se coloca el indicador de habilitación de interrupción en 0.

▼ ¿Qué es una excepción?

Son interrupciones por hardware creadas por el procesador en respuesta a ciertos eventos

▼ ¿Qué las puede causar?

- Condiciones excepcionales como el overflow en punto flotante
- Falla del programa por ejemplo tratar de ejecutar una instrucción que no está definida
- Falla de hardware, como un error de paridad
- Accesos no alineados o zonas de memoria no protegidas

▼ ¿Cómo es el ciclo de instrucción con interrupciones?

Para permitir las interrupciones se añade el ciclo de interrupción al ciclo de instrucción Antes de iniciar la captación de la siguiente instrucción, el procesador comprueba si existe alguna señal de interrupción pendiente, en el caso de que no haya ninguna continúa con el ciclo de instrucción normal. Caso contrario, suspende la ejecución del programa en curso, guarda su contexto que son todos los datos relacionados con el programa en ejecución y salva el contenido del contador de programa. Luego se carga en el contador del programa, la dirección de comienzo de la rutina de gestión de interrupciones. A continuación el procesador continúa con el ciclo de captación y accede a la primera instrucción del programa de gestor de interrupciones, cuando la rutina del gestor de interrupciones finaliza, se continúa con la ejecución normal del programa en el punto en donde se interrumpió.

▼ ¿Qué es una interrupción múltiple?

Durante la ejecución de un programa se pueden producir una interrupción durante la ejecución de otra interrupción.

▼ ¿Qué enfoques existen para resolver este problema?

A raíz de este problema se pueden tomar dos estrategias, la primera es desactivar las interrupciones mientras se está procesando una, esto significa que el procesador debe ignorar las interrupciones y dejarlas pendientes, una vez que se habilitan nuevamente, debe resolverlas. Esta estrategia supone un problema porque no se tienen en cuenta que pueden existir interrupciones con mayor prioridad, o que deben ser resueltas de manera inmediata.

El segundo enfoque consiste en definir prioridades para las interrupciones y permitir que una interrupción de nivel más alto pueda detener a una interrupción de nivel más bajo, cuando termina la interrupción de mayor nivel, se retorna a la anterior.

▼ ¿Cómo se reconoce una interrupción?

Existen tres formas de reconocer a una interrupción

La primera es que cada dispositivo que pueda provocar una interrupción tiene una entrada física de interrupción conectada a la CPU.

Otra alternativa es tener una línea de interrupción única , que es una entrada física en la cual se conectan todos los dispositivos, para lo cual se le deben preguntar a cada dispositivo si ha producido el pedido de interrupción.

La tercera alternativa es teniendo un vector que asocia la interrupción a una subrutina. Además del pedido de interrupción el dispositivo debe enviar un identificador.

▼ ¿Qué es el PIC?

Este dispositivo es conocido como Controlador de interrupciones programable. El PIC es un dispositivo interno con cuatro líneas de interrupción por hardware donde se conectan dispositivos que pueden interrumpir a la CPU, de esta manera el PIC permite utilizar varios dispositivos multiplexando sus pedidos a una única línea de interrupciones.

▼ ¿Cuántos y cuales son los registros del PIC?

Dirección	Registro	Nombre	Propósito	E/S
20h	EOI	Fin de interrupción	Avisa al PIC que se terminó una interrupción	S
21h	IMR	Máscara de interrupciones	Sus bits indican qué líneas de interrupción están habilitadas. Si el bit N vale 1, las interrupciones del dispositivo conectado a la línea INTN serán ignoradas. Si vale 0, las interrupciones del dispositivo serán atendidas en algún momento. Sólo importan los 4 bits menos significativos.	S
22h	IRR	Interrupciones pedidas	Sus bits indican qué dispositivos están solicitando una interrupción. Si el bit N vale 1, entonces el dispositivo conectado a la línea INTN está haciendo una solicitud. Sólo importan los 4 bits menos significativos.	E
23h	ISR	Interrupción en servicio	Sus bits indican si se está atendiendo la interrupción de algún dispositivo. Si el bit N vale 1, entonces el dispositivo conectado a la línea INTN está siendo atendido. Cómo en el MSX88 sólo se puede atender un dispositivo por vez, nunca habrá más de un bit del registro con el valor 1. Sólo importan los 4 bits menos significativos.	E
24h	INT0	ID de Línea INT0	Almacena el ID de la interrupción asociada al dispositivo F10 para buscar en el vector de interrupciones la dirección de comienzo de la subrutina que lo atiende.	S
25h	INT1	ID de Línea INT1	Almacena el ID de la interrupción asociada al dispositivo Timer para buscar en el vector de interrupciones la dirección de comienzo de la subrutina que lo atiende.	S
26h	INT2	ID de Línea INT2	Almacena el ID de la interrupción asociada al dispositivo Handshake para buscar en el vector de interrupciones la dirección de comienzo de la subrutina que lo atiende.	S
27h	INT3	ID de Línea INT3	Almacena el ID de la interrupción asociada al dispositivo CDMA para buscar en el vector de interrupciones la dirección de comienzo de la subrutina que lo atiende.	S

▼ ¿Por que existe el PIC?

Al tener una sola entrada física para interrupciones de dispositivos, surge el problema de que tenemos demasiados dispositivos de entrada y salida que quieren hacer interrupciones pero solo un “enchufe”, por lo tanto necesitamos un dispositivo que haga de intermediario.

Unidad 3 - Entrada y salida

`Hojas 208 a 247 del libro Stallings. Filminas 3.`

▼ ¿Por que los periféricos no se conectan directamente al bus del sistema?

Existen varios motivos:

- Hay una amplia variedad de periféricos con formas de funcionamiento diferentes. Podría ser
  imposible incorporar la lógica necesaria dentro del procesador para controlar tal diversidad de dispositivos.
  • A menudo la velocidad de transferencia de datos de los periféricos es mucho menor que la de la memoria o el procesador. Así, no es práctico utilizar un bus del sistema de alta velocidad para comunicarse directamente con un periférico.
  • Por otro lado, la velocidad de transferencia de algunos periféricos es mayor que la de la
  memoria o el procesador. De nuevo, esta diferencia daría lugar a comportamientos poco eficientes si no se gestionase correctamente.
  • Con frecuencia, los periféricos utilizan datos con formatos y tamaños de palabra diferentes de los del computador a los que se conectan.

▼ ¿Cuáles son las funciones de un modulo de entrada/salida?

Realizar la interfaz entre el procesador y la memoria a través del bus del sistema o un conmutador central.

Realizar la interfaz entre uno o más dispositivos periféricos mediante enlaces de datos específicos.

- Control y temporización.
- Comunicación con el procesador.
- Comunicación con los dispositivos.
- Almacenamiento temporal de datos.
- Detección de errores.

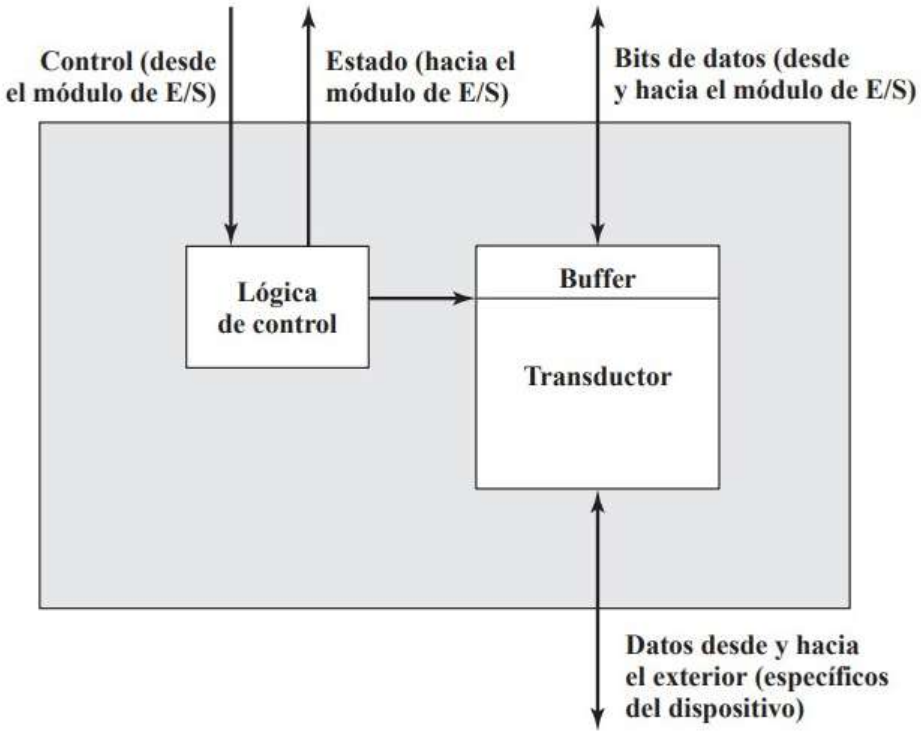
▼ ¿Cuáles son los bloques de un dispositivo externo?

La lógica de control asociada al dispositivo controla su operación en respuesta a las indicaciones del módulo de E/S.

El transductor convierte las señales eléctricas asociadas al dato a otra forma de
energía en el caso de una salida y viceversa en el caso de una entrada

Usualmente, existe un buffer asociado al transductor para almacenar temporalmente el dato que se está transfiriendo entre el módulo de E/S y el

exterior; es común un tamaño de buffer de 8 a 16 bits.



▼ De algunos ejemplos de dispositivos externos.

monitor/pantalla, mouse, teclado.

disco duro, CD, DVD

impresora, escáner

modem, acceso/interfaz de red

micrófono, parlantes

Sensores, alarmas, adquisición de datos

▼ ¿Qué implica la comunicación con el procesador?

- Decodificación de órdenes: el módulo de E/S acepta órdenes del procesador. Estas órdenes generalmente se envían utilizando líneas del bus de control. Por ejemplo, un módulo de E/S para un controlador de disco podría recibir las siguientes órdenes: LEER SECTOR («READ SECTOR»), ESCRIBIR SECTOR («WRITE SECTOR»), BUSCAR número

de pista (SEEK track number), y EXPLORAR IDentificador de registro (SCAN record ID). Cada una de las dos últimas órdenes incluye un parámetro que es enviado a través del bus de datos.
• Datos: el procesador y el módulo de E/S intercambian datos a través del bus de datos.
• Información de Estado: puesto que los periféricos son lentos, es importante conocer el estado del módulo de E/S. Por ejemplo, si se solicita a un módulo de E/S que envíe datos al procesador (lectura), puede que no esté preparado por encontrarse todavía respondiendo a una orden de E/S previa. Esta situación puede indicarse con una señal de estado. Señales de estado usuales son «BUSY» (ocupado) y «READY» (preparado). También puede haber señales para informar de ciertas situaciones de error.
• Reconocimiento de Dirección: igual que cada palabra de memoria tiene una dirección, cada dispositivo de E/S tiene otra. Así, un módulo de E/S puede reconocer una única dirección para cada uno de los periféricos que controla.

▼ ¿Qué es un puerto?

Un puerto es una interfaz a través de la cual se pueden enviar y recibir distintos tipos de datos.

▼ ¿Qué tipos de señales posee?

La señal de control que especifica la función a realizar, por ejemplo: READ, WRITE, OUTPUT.

La señal de estado que indica como se encuentra en dispositivo por ejemplo ready not-ready.

La señal de datos, que son conjuntos de bits que se intercambian.

▼ ¿Cómo es el diagrama de un modulo de entrada y salida?

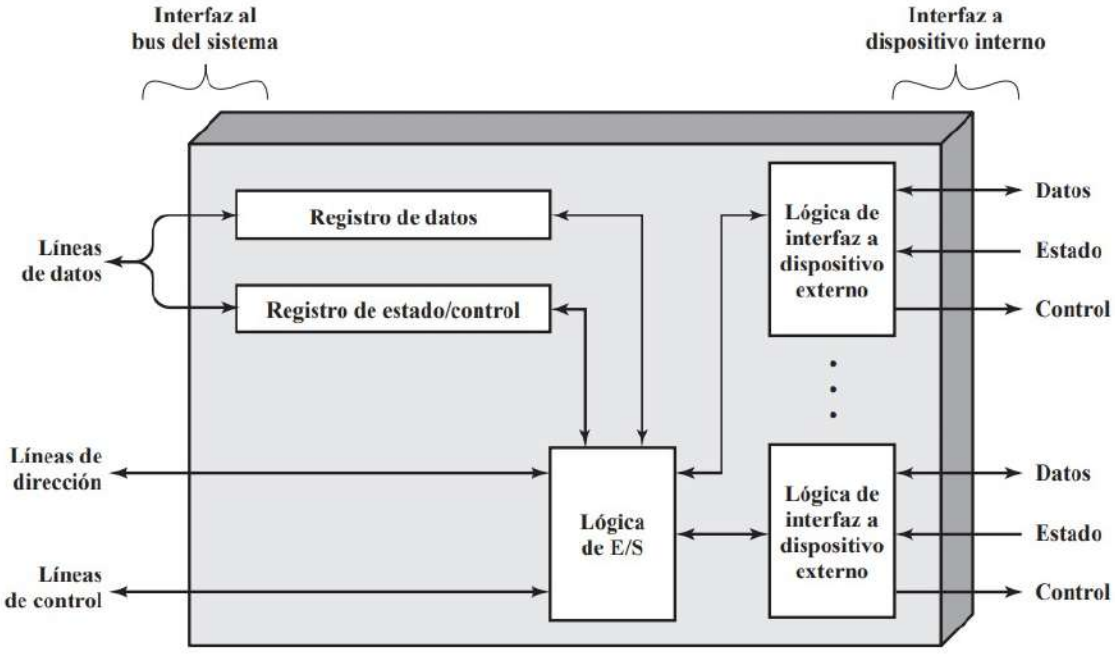


Figura 7.3. Diagrama de bloques de un módulo de E/S.

▼ ¿Cuáles son las capacidades de un modulo de entrada y salida?

• Ocultar las propiedades del dispositivo a la CPU. El funcionamiento de un módulo de E/S permite que el procesador vea a una amplia gama de dispositivos de una forma simplificada. Ante el espectro de posibilidades que pueden darse, el módulo de E/S debe ocultar los detalles de temporización formatos, y electromecánica de los dispositivos externos para que el procesador pueda funcionar únicamente en términos de órdenes de lectura y escritura, y posiblemente órdenes de abrir y cerrar ficheros

• Ocuparse de uno o varios dispositivos.
• Controlar o no las funciones del dispositivo.

▼ ¿Qué requiere una operación de entrada y salida?

Direccionamiento

Transferencia de información

Gestión de la transferencia

▼ ¿Qué técnicas de entrada y salida existen?

- E/S programada con espera de respuesta.
- E/S con interrupciones.
- E/S con acceso directo a memoria.

▼ ¿Cómo funciona la entrada y salida programada?

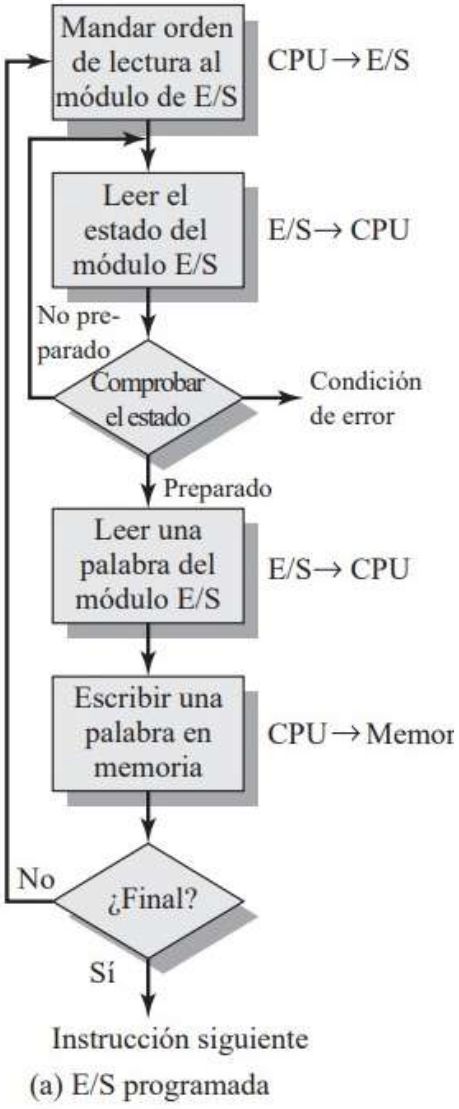
Con la E/S programada, los datos se intercambian entre el procesador y el módulo de E/S. El procesador ejecuta un programa que controla directamente la operación de E/S, incluyendo la comprobación del estado del dispositivo, el envío de una orden de lectura o escritura y la transferencia del dato.

▼ ¿Qué desventaja tiene este método?

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.

▼ Describa los pasos que hace la CPU

La CPU solicita la operación de E/S al módulo . • El módulo E/S realiza la operación. • El módulo E/S activa los bits de estado del dispositivo direccionado y espera. • La CPU comprueba periódicamente el estado de esos bits, hasta que detecta que la operación fue completada. • En caso contrario la CPU espera y vuelve a comprobarlo más tarde.



(a) E/S programada

▼ ¿Qué tipo de ordenes entrada/salida puede enviar el procesador?

Al ejecutar una instrucción relacionada con una E/S, el procesador proporciona una dirección, especificando el módulo de E/S particular y el dispositivo externo, y una orden de E/S. Hay cuatro tipos de órdenes de E/S que puede recibir un módulo de E/S cuando es direccionado por el procesador:

• Control: se utiliza para activar el periférico e indicarle qué hacer. Por ejemplo, puede indicarse a una unidad de cinta magnética que se rebobine

o que avance al registro siguiente. Estas órdenes son específicas del tipo particular de periférico.

• Test: se utiliza para comprobar diversas condiciones de estado asociadas con el módulo de E/S y sus periféricos. El procesador podrá comprobar si el periférico en cuestión está conectado y disponible para su uso. También podrá saber si la operación de E/S más reciente ha terminado y si se ha producido algún error.

• Lectura: hace que el módulo de E/S capte un dato de un periférico y lo sitúe en un buffer interno. Después, el procesador puede obtener el dato solicitando que el módulo de E/S lo ponga en el bus de datos.

• Escritura: hace que el módulo de E/S capte un dato (byte o palabra) del bus de datos y posteriormente lo transmita al periférico.

▼ ¿Qué modos de direccionamiento existen cuando el procesador, la memoria principal y las E/S comparten un bus común?

Existen el asignado en memoria y el aislado

▼ ¿Cómo funciona cada uno?

Con las E/S asignadas en memoria, existe un único espacio de direcciones para las posiciones de memoria y los dispositivos de E/S. El procesador considera a los registros de estado y de datos de los módulos de E/S como posiciones de memoria y utiliza las mismas instrucciones máquina para acceder tanto a memoria como a los dispositivos de E/S. Así, por ejemplo, con diez líneas de dirección, se puede acceder a un total de 1024 posiciones de memoria y direcciones de E/S, en cualquier combinación.

Con las E/S asignadas en memoria, se necesita una sola línea de lectura y una sola línea de escritura en el bus. Alternativamente, el bus puede disponer de líneas de lectura y escritura en memoria junto con líneas para órdenes de entrada y salida. En este caso, las líneas de órdenes especifican si la dirección se refiere a una posición de memoria o a un dispositivo de E/S. El rango completo de direcciones está disponible para ambos. De nuevo, con diez líneas de dirección, el sistema puede soportar ahora 1 024 posiciones de memoria y 1 024

direcciones de E/S. Puesto que el espacio de direcciones de E/S está aislado del de memoria, éste se conoce con el nombre de E/S aislada.

▼ ¿Cómo funciona la E/S con interrupciones?

Una alternativa, a la e/s programada, consiste en que el procesador, tras enviar una orden de E/S a un módulo, continúe realizando algún trabajo útil. Después, el módulo de E/S interrumpirá al procesador para solicitar su servicio cuando esté preparado para intercambiar datos con él. El procesador ejecuta entonces la transferencia de datos, como antes, y después continúa con el procesamiento previo. Estudiemos cómo funciona, primero desde el punto de vista del módulo de E/S. Para una entrada, el módulo de E/S recibe una orden READ del procesador. Entonces, el módulo de E/S procede a leer el dato desde el periférico asociado. Una vez que el dato está en el registro de datos del módulo, el módulo envía una interrupción al procesador a través de una línea de control. Después, el módulo espera hasta que el procesador solicite su dato. Cuando ha recibido la solicitud, el módulo sitúa su dato en el bus de datos y pasa a estar preparado para otra operación de E/S.

▼ ¿Qué desventajas tiene este método?

Las E/S con interrupciones todavía consumen gran cantidad del tiempo del procesador puesto que cada palabra de datos que va desde la memoria al módulo de E/S o viceversa debe pasar a través del procesador.

▼ Describa los pasos que hace el procesador

Consideremos con más detalle el papel del procesador en las E/S. Cuando se produce una interrupción se disparan una serie de eventos en el procesador, tanto a nivel hardware como software.

1. El dispositivo envía una señal de interrupción al procesador.
2. El procesador termina la ejecución de la instrucción en curso antes de responder a la interrupción.
3. El procesador comprueba si hay interrupciones, determina que hay una, y envía una señal de reconocimiento al dispositivo que originó la interrupción. La señal de reconocimiento hace que el dispositivo desactive su señal de interrupción.

4. Ahora el procesador necesita prepararse para transferir el control a la rutina de interrupción. Para empezar, debe guardar la información necesaria para continuar el programa en curso en el punto en que se interrumpió. La información mínima que se precisa es (a) el estado del procesador, que se almacena en un registro llamado Palabra de Estado del Programa (PSW, Program Status Word), y (b) la posición de la siguiente instrucción a ejecutar, que está contenida en el contador de programa.
5. Después, el procesador carga el contador de programa con la posición de inicio del programa de gestión de la interrupción solicitada. Según sea la arquitectura del computador y el diseño del sistema operativo, puede haber un solo programa, uno por cada tipo de interrupción, o uno por cada dispositivo y cada tipo de interrupción. Si hay más de una rutina de gestión de interrupción, el procesador debe determinar a qué programa llamar. Esta información puede haber sido incluida en la señal de interrupción original, o el procesador puede tener que enviar una solicitud al dispositivo que originó la interrupción para que este responda con la información que se precise. Una vez que el contador de programa se ha cargado, el procesador continúa con el ciclo de instrucción siguiente, que empieza con la captación de instrucción. Puesto que la instrucción a captar, viene determinada por el contenido del contador de programa, el control se transfiere al programa de gestión de interrupción. La ejecución de este programa da lugar a las siguientes operaciones:
6. Hasta este momento, se han almacenado en la pila del sistema el contador de programa y el PSW del programa interrumpido. Sin embargo, hay otra información que se considera parte del «estado» de un programa en ejecución. En concreto, se deben guardar los contenidos de los registros del procesador puesto que estos registros pueden ser utilizados por la rutina de interrupción. Usualmente, la rutina de gestión de interrupción empezará almacenando en la pila los contenidos de todos los registros. En este caso, un programa de usuario es interrumpido después de la instrucción de la posición N. Los contenidos de todos los registros junto con la dirección de la siguiente instrucción (N 1) se introducen en la pila. El puntero de la pila se

actualiza para que apunte a la nueva cabecera de la pila, y el contador de programa se actualiza para que apunte al comienzo de la rutina de servicio de interrupción.

7. La rutina de gestión de la interrupción puede continuar ahora procesando la interrupción. Esto incluirá el examen de la información de estado relativa a la operación de E/S o a cualquier otro evento que causara la interrupción. También puede implicar el envío al dispositivo de E/S de órdenes o señales de reconocimiento adicionales.
8. Cuando el procesamiento de la interrupción ha terminado, los valores de los registros almacenados se recuperan de la pila y se vuelven a almacenar en los registros.
9. El paso final es recuperar los valores del PSW y del contador de programa desde la pila. Como resultado, la siguiente instrucción que se ejecute pertenecerá al programa previamente interrumpido.

Obsérvese que es importante almacenar toda la información del estado del programa interrumpido para que este pueda reanudarse. Esto se debe a que la interrupción no es una llamada a una rutina realizada desde el programa. En cambio, la interrupción puede producirse en cualquier momento y por consiguiente en cualquier punto de la ejecución del programa de usuario. Una interrupción es impredecible. De hecho, como se verá en el siguiente capítulo, los dos programas pueden no tener nada en común y pueden pertenecer a distintos usuarios.

▼ ¿Qué cuestiones de diseño hay que tener en cuenta?

En la implementación de las E/S mediante interrupciones aparecen dos cuestiones. Primero, puesto que casi invariablemente habrá múltiples módulos de E/S, cómo determina el procesador qué dispositivo ha provocado la interrupción. Y segundo, si se han producido varias interrupciones, cómo decide el procesador la que debe atender.

▼ ¿Cómo se puede identificar el dispositivo?

Existen cuatro técnicas, la primera es tener múltiples líneas de interrupción, la segunda es la consulta de software o software poll, la

tercera es conexión en cadena o Daisy chain. y la ultima es el arbitraje de bus o vectorizada.

La aproximación más directa al problema consiste en proporcionar varias líneas de interrupción entre el procesador y los módulos de E/S. Sin embargo, no resulta práctico dedicar más de unas pocas líneas del bus o terminales del procesador a ser líneas de interrupción. En consecuencia, incluso si se utilizan varias líneas, es probable que a cada una se conecten varios módulos de E/S. Por eso, se debe utilizar alguna de las otras tres técnicas en cada línea.

Una alternativa es la consulta software. Cuando el procesador detecta una interrupción, se produce una bifurcación a una rutina de servicio de interrupción que se encarga de consultar a cada módulo de E/S para determinar el módulo que ha provocado la interrupción. Entonces, el procesador lee el estado del registro de cada módulo de E/S para identificar el módulo que solicitó la interrupción. Una vez identificado el módulo, se produce una bifurcación para que el procesador ejecute la rutina de servicio específica para ese dispositivo. La desventaja de la consulta software está en el tiempo que consume.

Una técnica más eficiente consiste en utilizar la conexión en cadena (daisy chain) de los módulos de E/S que proporciona, de hecho, una consulta hardware. La línea de reconocimiento de interrupción se conecta encadenando los módulos uno tras otro. Cuando el procesador recibe una interrupción, activa el reconocimiento de interrupción. Esta señal se propaga a través de la secuencia de módulos de E/S hasta que alcanza un módulo que solicitó interrupción. Normalmente este módulo responde colocando una palabra en las líneas de datos. Esta palabra se denomina vector y es la dirección del módulo de E/S o algún otro tipo de identificador específico. En cualquier caso, el procesador utiliza el vector como un puntero a la rutina de servicio de dispositivo apropiada. Así se evita tener que ejecutar una rutina de servicio general en primer lugar. Esta técnica se conoce con el nombre de interrupción vectorizada.

Hay otra técnica que hace uso de las interrupciones vectorizadas, y se trata de el arbitraje de bus. Con el arbitraje de bus, un módulo de E/S

debe en primer lugar disponer del control del bus antes de poder activar la línea de petición de interrupción. Así, solo un módulo puede activar la línea en un instante. Cuando el procesador detecta la interrupción, responde mediante la línea de reconocimiento de interrupción. Después, el módulo que solicitó la interrupción sitúa su vector en las líneas de datos.

▼ ¿Cómo se gestiona una interrupción múltiple?

Con varias líneas de interrupción, el procesador simplemente selecciona la línea con más prioridad. Con la consulta software, el orden en el que se consultan los módulos determina su prioridad. De igual forma, el orden de los módulos en la conexión en cadena (daisy chain) determina su prioridad. Si existe un maestro del bus, solo él puede interrumpir.

▼ ¿Cómo funcionan las interrupciones en el procesador 8086 de Intel?

El Intel 8086 tiene sólo una línea de petición de interrupción (INTReq) y por lo tanto una sola de confirmación (INTAck). Se deberá utilizar un árbitro o gestor de interrupciones externo, el 8259A (PIC) Este chip tiene 8 líneas de interrupción, por lo tanto podrá manejar 8 módulos de E/S. Usando conexión en cascada se puede gestionar hasta 64 módulos.

▼ Compare las e/s programada con la e/s de interrupciones

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

1. La velocidad de transferencia de E/S está limitada por la velocidad a la cual el procesador puede comprobar y dar servicio a un dispositivo.

2. El procesador debe dedicarse a la gestión de las transferencias de E/S; se debe ejecutar cierto número de instrucciones por cada transferencia de

E/S

▼ ¿Cómo funciona la entrada y salida con acceso directo a memoria? (DMA)

El DMA requiere un módulo adicional en el bus del sistema. El módulo o controlador de DMA es capaz de imitar al procesador y, de hecho, 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. Cuando el procesador desea leer o escribir un bloque de datos, envía una orden al módulo de DMA, incluyendo la siguiente información: •
Si se solicita una lectura o una escritura, utilizando la línea de control de lectura o escritura entre el procesador y el módulo de DMA.
• La dirección del dispositivo de E/S en cuestión, indicada a través de las líneas de datos.
• La posición inicial de memoria a partir de donde se lee o se escribe, indicada a través de las líneas de datos y almacenada por el módulo de DMA en su registro de direcciones.
• El número de palabras a leer o escribir, también indicado a través de las líneas de datos y almacenado en el registro de cuenta de datos.

Después, el procesador continúa con otro trabajo. Ha delegado la operación de E/S al módulo de DMA, que se encargará de ella. El módulo de DMA transfiere el bloque completo de datos, palabra a palabra, directamente desde o hacia la memoria, sin que tenga que pasar a través del procesador. Cuando la transferencia se ha terminado, el módulo de DMA envía una señal de interrupción al procesador. Así pues, el procesador solo interviene al comienzo y al final de la transferencia.

▼ ¿Cómo es el diagrama de un modulo de DMA?

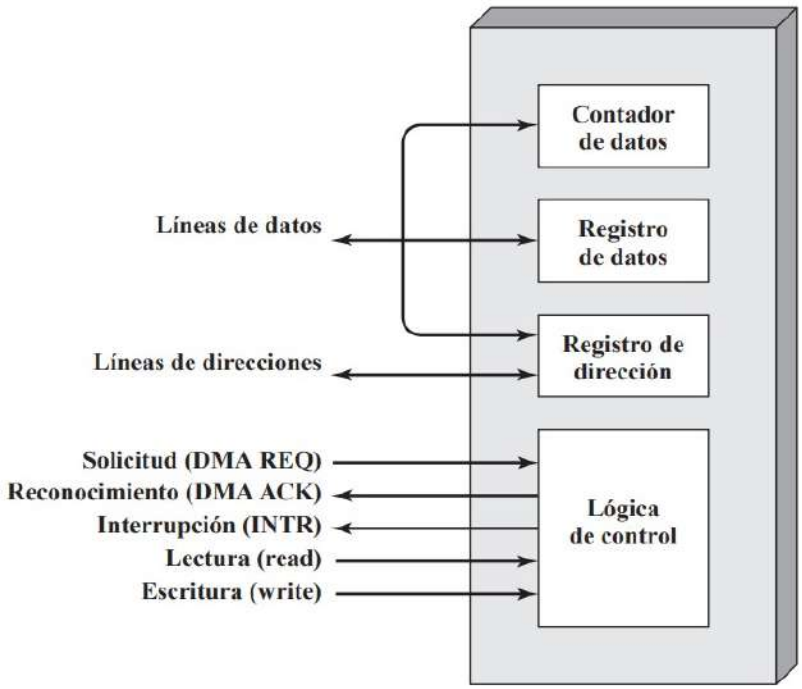
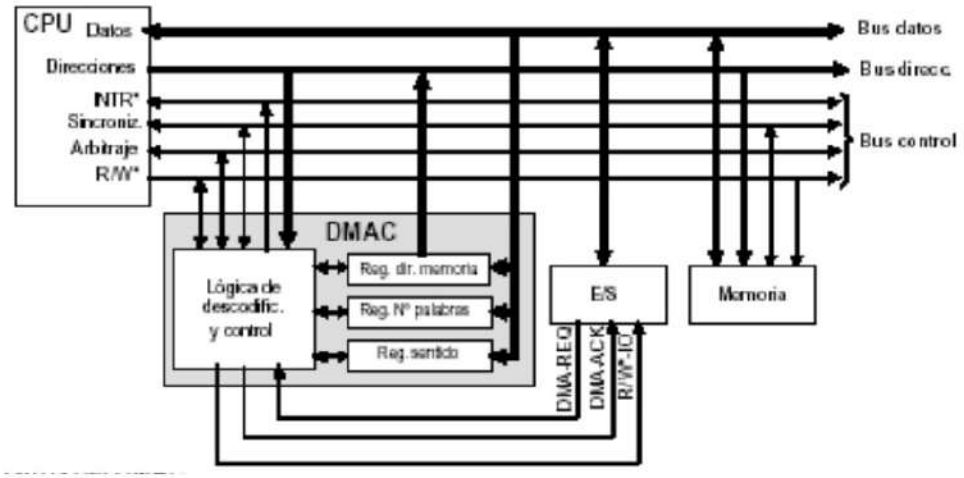


Figura 7.11. Diagrama de bloques típico de un módulo de DMA.

▼ ¿Qué es el DMAC?

El DMAC es el controlador de DMA, que debe actuar como maestro del bus durante la transferencia DMA y debe ser capaz de, solicitar el uso de bus mediante señales y la lógica de arbitraje, especificar la dirección de memoria sobre la cual se hace la transferencia y generar las señales de control de bus.

▼ ¿Cómo es su diagrama?



▼ ¿Cuáles son las etapas de una transferencia DMA?

1. Inicialización de transferencia: la CPU envía los parámetros de la transferencia: inicializa interfaz del periférico: nº de bytes a transferir, tipo de transferencia (R/W)… inicializa DMA: nº bytes/palabras, tipo de transferencia (R/W), dirección de memoria inicial para transferencia; nº de canal, si DMA tiene más de un canal…

2. La CPU retorna a su tarea y se olvida de la evolución de la transferencia.

3. Se realiza la transferencia: Cuando el periférico está preparado se lo avisa a DMAC. El DMAC pide el control del bus: actúa como master del bus durante la transferencia, especifica la dirección de memoria sobre la que realiza la transferencia, y realiza transferencia entre periférico y memoria. Genera las señales de control: Tipo de operación (R/W). Señales de sincronización de la transferencia. 4

. Finaliza la transferencia: El DMAC libera el bus y devuelve el control a la CPU (generalmente con una interrupción)

▼ ¿Qué problemas pueden encontrarse al usar 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 instrucciones o datos. Este problema se puede solucionar en gran medida con el uso de la cache por que la mayor parte del tiempo, la CPU lee instrucciones 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. y por lo tanto el DMAC puede aprovechar las fases de ejecución de una instrucción en las que la CPU no utiliza el bus para realizar sus transferencias.

▼ ¿Qué métodos de transferencias puede realizar el DMAC?

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.

Existen dos tipos de transferencias, **Por ráfagas** y **por robo de ciclo.**

DMA modo ráfagas: 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. La ventaja que tiene es que la transferencia se realiza muy rápido, pero degrada el rendimiento del sistema por que durante ese tiempo la CPU no puede hacer uso del bus de memoria.

DMA por 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. Tiene como ventaja que no se degrada el rendimiento del sistema, pero la transferencia tarda mas tiempo. Para la CPU no es una interrupción. por lo tanto 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 ende para transferencia de E/S de múltiples palabras, es la técnica más eficiente.

▼ ¿Cómo fueron las etapas de evolución del funcionamiento de las entradas y salidas?

1. La CPU controla directamente al periférico. Esta situación se observa en los dispositivos simples controlados por microprocesadores.

2. Se añade un controlador o módulo de E/S. La CPU utiliza E/S programada sin interrupciones. De esta forma, la CPU se independiza de los detalles específicos de las interfaces de los dispositivos externos.

3. Se utiliza la misma configuración del paso 2, pero ahora se emplean interrupciones. La CPU no necesita esperar a que se realice la operación de E/S, incrementándose la eficiencia.

4. El módulo de E/S tiene acceso directo a la memoria a través del DMA. Ahora se puede transferir un bloque de datos a, o desde, la memoria sin implicar a la CPU, excepto al comienzo y al final de la transferencia.

5. El módulo de E/S se mejora haciendo que se comporte como un procesador en sí, con un repertorio especializado de instrucciones orientado a las E/S. La CPU hace que el procesador de E/S ejecute un programa de E/S en memoria. El procesador de E/S capta y ejecuta sus instrucciones sin intervención de la CPU. Esto permite que la CPU pueda especificar una secuencia de actividades de E/S y ser interrumpida cuando se haya completado la secuencia entera.

6. El módulo de E/S tiene una memoria local propia y es, de hecho, un computador en sí. Con esta arquitectura, se puede controlar un conjunto grande de dispositivos de E/S con la mínima intervención de la CPU. Un uso común de este tipo de arquitectura ha sido la comunicación con terminales interactivos. El procesador de E/S se ocupa de la mayoría de las tareas correspondientes al control de los terminales.

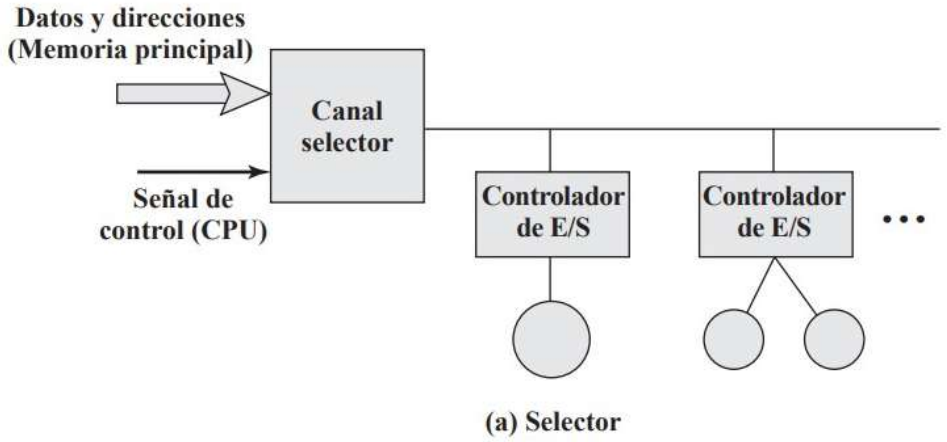
▼ ¿Cuáles son las características de los canales de entrada y salida?

El canal de E/S representa una ampliación del concepto de DMA. Un canal de E/S puede ejecutar instrucciones de E/S, lo que le confiere un control completo sobre las operaciones de E/S. En un computador con tales dispositivos, la CPU no ejecuta instrucciones de E/S. Dichas instrucciones se almacenan en memoria principal para ser ejecutadas por un procesador de uso específico contenido en el propio canal de E/S. De esta forma, la CPU inicia una transferencia de E/S indicando al canal de E/S que debe ejecutar un programa de la memoria. El

programa especifica el dispositivo o dispositivos, el área o áreas de memoria para almacenamiento la prioridad, y las acciones a realizar en ciertas situaciones de error. El canal de E/S sigue estas instrucciones y controla la transferencia de datos.

▼ ¿Qué tipos de canales de e/s existen?

El Selector que 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.



(a) Selector

Un canal multiplexor que puede manejar las E/S de varios dispositivos al mismo tiempo. Para dispositivos de velocidad reducida, un multiplexor de byte acepta o transmite caracteres tan rápido como es posible a varios dispositivos. Por ejemplo, la cadena de caracteres resultante a partir de tres dispositivos con diferentes velocidades y cadenas individuales A1A2A3A4...,B1B2B3B4..., y C1C2C3C4..., podría ser A1B1C1A2C2A3B2C3A4, y así sucesivamente. Para dispositivos de velocidad elevada, un multiplexor de bloque entrelaza bloques de datos de los distintos dispositivos.

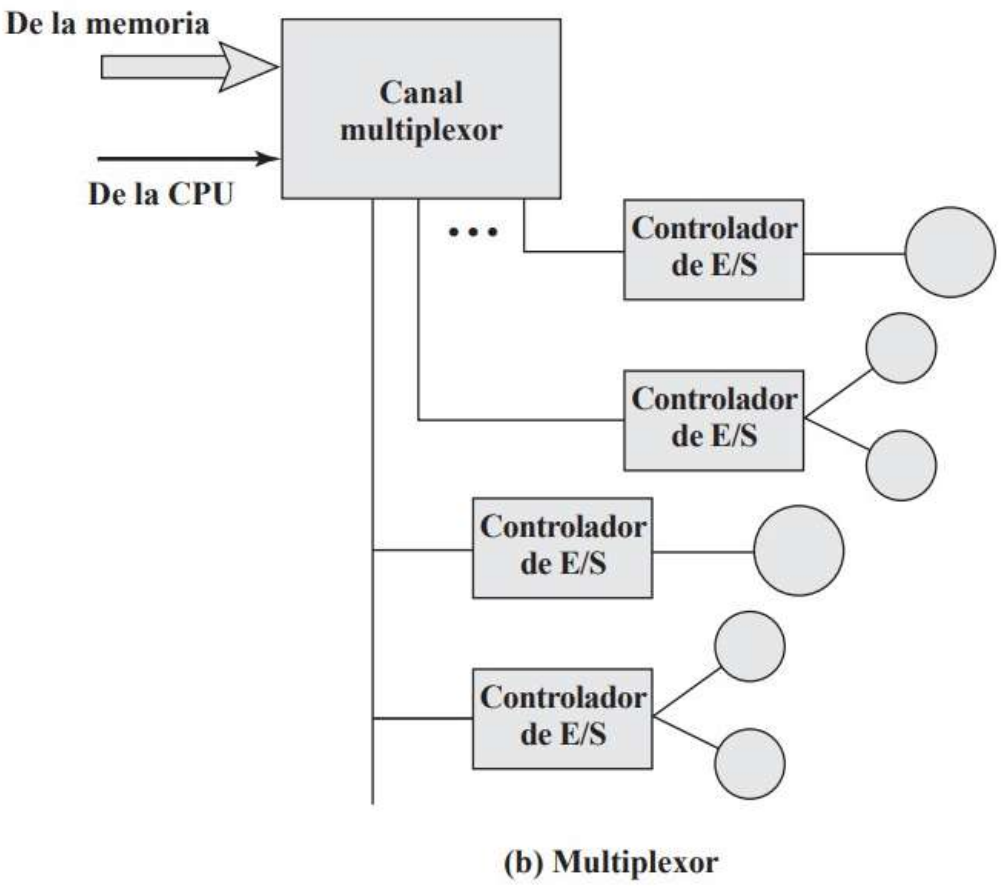


Figura 7.15. Arquitectura de un canal de E/S.

Unidad 4 - Segmentación de Instrucciones

`Clase 4 filminas, Hojas 449 a 464 del libro Stallings. Hojas 35 a 42 libro de Marta Beltrán Pardo.`

▼ ¿Qué es el CPI?

El CPI es el acronimo de ciclos por instrucción que mide el rendimiento de un procesador en base a la cantidad de ciclos de media que le toma a la CPU terminar con una instrucción

▼ Cual es el CPI ideal?

El CPI ideal de un procesador segmentado debe ser 1, de manera que se finaliza una nueva instrucción cada ciclo y en media, es como si cada

instrucción sólo necesitara 1 ciclo para ejecutarse. Es decir, se consigue el CPI de un procesador secuencial mono-ciclo.

▼ Cual es la diferencia entre un procesador monociclo y uno multiciclo?

El procesador monociclo completa cada instrucción en un único ciclo de reloj. Mientras que el procesador multiciclo puede tardar mas de un ciclo para terminar una instrucción, tantos como sea necesario.

Para que funcione el procesador monociclo se debe adaptar al procesador para que en la instrucción mas larga pueda ser resuelta en solo un ciclo de reloj, esto significa que el tiempo de ciclo de reloj se hace tan largo como la instrucción mas lenta. Esto es poco optimizable y poco eficiente.

▼ ¿En que etapas se puede descomponer una instrucción?

• Captar instrucción: leer la supuesta siguiente instrucción en un buffer.

• Decodificar instrucción: determinar el código de operación y los campos de operando.

• Calcular operandos calcular la dirección efectiva de cada operando fuente. Esto puede involucrar direccionamiento mediante un desplazamiento, indirecto a través de registro, indirecto, u otras formas de calcular la dirección.

• Captar operandos: captar cada operando que resida en memoria. Los operandos en registros no tienen que ser captados.

• Ejecutar instrucción: realizar la operación indicada y almacenar el resultado, si lo hay, en la posición de operando destino especificada.

• Escribir operando: almacenar el resultado en memoria.

▼ ¿Qué tareas se realizan por 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

▼ ¿Qué es la segmentación de cause?

La segmentación de cause es un mecanismo que tiene la CPU para organizar el hardware para realizar mas de una operación al mismo tiempo. Para esto las operaciones se dividen en múltiples pasos; para poder entender este concepto mas fácil se lo puede comparar con una fabrica, en la cual se puede dividir el producto en distintos componentes, construir cada uno por separado y luego ensamblarlo.

▼ Que características tiene la segmentación de cause?

Aplica una mejora al rendimiento.

Es invisible para el programador.

Necesita que las instrucciones sean uniformes y estén dividas por etapas.

▼ ¿Qué dificultades tiene la implementación de la segmentación de cause?

La principal dificultad que conlleva la implementacion de la segmentación de cause esta relacionada con los saltos condicionales, no siempre se puede proveer en que instrucción continuara el programa, por lo tanto se pierde rendimiento. También, se puede encontrar la dificultad relacionada a los registros, puede que al momento de captar los operandos necesarios, estos se encuentren siendo utilizados para alguna otra operación. Ademas no todas las instrucciones necesitan todas las etapas. Algunas de estas dificultades toman el nombre de **riesgos**.

▼ ¿Qué tipos de riesgos existen?

Riesgos estructurales. Se producen cuando dos o más instrucciones necesitan utilizar el mismo recurso hardware al mismo tiempo.

Riesgos de datos. Se producen cuando dos o más instrucciones presentan dependencias de datos entre sí que, si no se resuelven correctamente, podrían llevar a la obtención de resultados erróneos en la ejecución del código por realizar operaciones de lectura y escritura en un orden diferente al indicado por la secuencia de instrucciones.

Riesgos de control. Se producen cuando una instrucción que modifica el valor del PC(program counter) todavía no lo ha hecho cuando se tiene que comenzar la ejecución de la siguiente instrucción.

▼ ¿Qué tipos de riesgo de datos existen?

— RAW o Read After Write (Lectura después de Escritura).

— WAR o Write After Read (Escritura después de Lectura).

— WAW o Write After Write (Escritura después de Escritura).

Las dependencias RAW son las reales, ya que las WAW y las WAR, aunque pueden provocar riesgos, se deben a la reutilización de los registros visibles para el programador, pero no existe un flujo real de información entre las instrucciones que provocan la dependencia.

▼ ¿Cómo se puede estimar el tiempo que tardara una segmentación de cause?

PRESTACIONES DE UN CAUCE SEGMENTADO

En esta subsección desarrollamos algunas medidas sencillas de las prestaciones de un cauce segmentado y del incremento de velocidad relativo (basadas en la discusión de [HWAN93]). El tiempo de ciclo τ de un cauce de instrucciones es el tiempo necesario para que un conjunto de instrucciones avance una etapa a través del cauce; cada columna de las Figuras 12.10 y 12.11 representa un tiempo de ciclo. El tiempo de ciclo puede determinarse como

$$\tau = \max_i[\tau_i] + d = \tau_m + d \qquad 1 \leq i \leq k$$

donde

τ_i = retardo de tiempo de la circuitería de la i-ésima etapa del cauce
τ_m = máximo retardo de etapa (retardo a través de la etapa que experimenta el mayor retardo)
k = número de etapas del cauce de instrucciones
d = retardo de tiempo de un registro *latch*, necesario para que avancen las señales y datos de una etapa a la siguiente

▼ ¿Cómo se puede solucionar un riesgo?

Existe una técnica universal para resolver estos tres tipos de riesgos, la introducción de paradas en la ruta de datos. Con esta técnica se asegura la obtención de resultados correctos y se evitan los conflictos por recursos y por incertidumbres en cuanto al valor del PC. Es la unidad de control la encargada de realizar la detección de riesgos en la etapa D (con una lógica específica para esta tarea que evalúa estos riesgos cuando se decodifica la instrucción) y

de inhibir el avance de las instrucciones por la ruta de datos hasta que estos queden resueltos.

- ¿Como se pueden solucionar los riesgos estructurales?

  La resolución de estos riesgos suele ser la más sencilla ya que basta con duplicar los recursos hardware que provocan los conflictos, segmentarlos o realizar turnos para acceder a ellos. Son las soluciones que se han utilizado en el diseño del nanoMIPS para evitar los riesgos estructurales por la memoria y por el banco de registros, separando entre las memorias de instrucciones y datos y realizando turnos para leer y escribir en el banco de registros (las escrituras siempre se hacen en la primera mitad de los ciclos de reloj y las lecturas, en la segunda).

- ¿Como se pueden solucionar los riesgos de datos?

  Para resolver los riesgos de datos RAW que aparecen en el nanoMIPS existe una técnica hardware y otra software. La técnica hardware se denomina adelantamiento o cortocircuito y consiste en pasar directamente el resultado obtenido con una instrucción a las instrucciones que lo necesitan como operando. Es decir, estas instrucciones no tienen que esperar a que la que produce el resultado lo escriba en el registro destino, sino que reciben su valor en cuanto está disponible. Esta técnica puede mejorar mucho el rendimiento de un procesador segmentado y es fácil de imple- mentar, ya que lo único que hay que hacer es identificar todos los posibles adelantamientos necesarios para el repertorio que ejecuta el procesador y comunicar los registros de segmentación involucrados. Pero no siempre se puede aplicar esta técnica evitando con ello que se produzcan paradas. Si se estudia en profundidad el caso del nanoMIPS tenemos las siguientes posibilidades para riesgos RAW.

  LW seguido de SW: El operando está disponible a la salida de la etapa M de la instrucción LW y se necesita a la entrada de la etapa M de la instrucción SW.

- ¿Como se puede solucionar un riesgo de control?

  De nuevo existen dos tipos de técnicas para resolver estos riesgos, una hardware y otra software. Además, se puede modificar ligeramente la ruta de datos del procesador, para que, en el caso que no haya otra solución

además de la parada de la ruta de datos, por lo menos ésta sea lo más corta posible. Se ha mostrado que al resolver un riesgo de control en el nanoMIPS mediante paradas, son necesarios tres ciclos de espera hasta que la instrucción de salto BEQ carga el valor adecuado en el PC y se puede continuar con la ejecución. Existe una sencilla modificación de la ruta de datos del nanoMIPS que puede reducir esta parada a un único ciclo. La idea es sencilla: adelantar tanto como sea posible la resolución de los saltos. Como es en la etapa D en la que se decodifica la instrucción y se sabe que es un salto, en esta misma etapa se añade el hardware necesario para evaluar la condición del salto (un restador, el módulo Zero?) y para calcular la dirección destino del salto (un sumador). La señal de control Branch pasa a generarse directamente para ser utilizada en la etapa D, y la señal de control ALUSrcA para la etapa X ya no es necesaria. Además, si se realiza esta modificación en la ruta de datos, aparece un nuevo adelantamiento a la etapa D (cuando hay un riesgo de datos RAW con uno de los registros necesarios para la evaluación de la condición del salto), lo que exige añadir el hardware necesario para realizar los adelantamientos también en esta etapa si se utiliza esta técnica para resolver los riesgos de datos. Con la ruta de datos modificada del nanoMIPS, sólo es necesario un ciclo de parada para resolver el riesgo de control

---

Unidad 5 - Posibles soluciones a atascos.

`Clase 5 filminas,  Hojas 43 a 63 libro de Marta Beltrán Pardo. Hojas 449 a 464 del libro Stallings. Nota: algunas preguntas son parecidas a la unidad 4.`

▼ ¿Qué técnica universal soluciona los atascos?

Existe una técnica universal para resolver estos tres tipos de riesgos, la introducción de paradas en la ruta de datos. Con esta técnica se asegura la obtención de resultados correctos y se evitan los conflictos por recursos y por incertidumbres en cuanto al valor del PC. Es la unidad de control la encargada de realizar la detección de riesgos en la etapa D y de inhibir el avance de las instrucciones por la ruta de datos hasta que estos queden resueltos.

▼ ¿Cuál es el problema de esta técnica?

El problema de esta técnica es que los riesgos de datos y de control son muy comunes, y si se resuelven siempre mediante la introducción de paradas, el CPI del procesador se aleja mucho del valor ideal de 1 que se busca con la segmentación.

▼ ¿Qué técnicas permite solucionar los riesgos de datos?

Para solucionar los riesgos de datos primero hay que analizar el código para determinar como y cuando se producen.

Existen dos tipos de soluciones, una del lado del hardware y otra del lado del software.
La técnica **hardware** se denomina adelantamiento o cortocircuito y consiste en pasar directamente el resultado obtenido con una instrucción a las instrucciones que lo necesitan como operando. Es decir, estas instrucciones no tienen que esperar a que la que produce el resultado lo escriba en el registro destino, sino que reciben su valor en cuanto está disponible. Esta técnica puede mejorar mucho el rendimiento de un procesador segmentado y es fácil de implementar, ya que lo único que hay que hacer es identificar todos los posibles adelantamientos necesarios para el repertorio que ejecuta el procesador y comunicar los registros de segmentación involucrados. Pero no siempre se puede aplicar esta técnica evitando con ello que se produzcan paradas.

La alternativa **software** para la resolución de riesgos de datos es responsabilidad del compilador, y en realidad no es una técnica que resuelva los riesgos sino que más bien los evita. Se trata de reordenar el código, de manera que la reordenación no afecte a los resultados y evitar así las combinaciones de instrucciones que provocan paradas en la ruta de datos. La norma general es separar lo más posible las instrucciones con dependencia RAW que generan el riesgo (se puede usar la instrucción nop para separar, que generan un retardo).

▼ ¿Qué técnicas permiten solucionar los riesgos estructurales?

Es el mas fácil de solucionar de los tres, solo basta con duplicar los recursos de hardware, segmentarlos o realizar turnos para acceder a estos.

▼ ¿Qué técnicas permiten solucionar los riesgos de control?

De nuevo existen dos tipos de técnicas para resolver estos riesgos, una **hardware** y otra **software**. Además, se puede modificar ligeramente la ruta de datos del procesador, para que, en el caso que no haya otra solución además de la parada de la ruta de datos, por lo menos ésta sea lo más corta posible.
Existe una sencilla modificación de la ruta de datos del nanoMIPS que puede reducir esta parada a un único ciclo. La idea es sencilla: adelantar tanto como sea posible la resolución de los saltos. Como es en la etapa D en la que se decodifica la instrucción y se sabe que es un salto, en esta misma etapa se añade el hardware necesario para evaluar la condición del salto y para calcular la dirección destino del salto. Aún con esta modificación, cada vez que se ejecuta una instrucción de salto, el procesador tiene que parar un ciclo, y las instrucciones de salto son muy frecuentes, por lo que estas paradas empeoran significativamente el rendimiento de la segmentación.
La técnica **hardware** que intenta evitar esta parada se basa en realizar una predicción para el salto. Las predicciones siempre consiguen una reducción de la penalización por salto cuando aciertan. Las predicciones más sencillas son las estáticas, en las que el procesador está diseñado para predecir en todos los casos que el salto se va a tomar o que el salto no se va a tomar. Es decir, la predicción es siempre la misma para cualquier salto. Con la predicción de salto no tomado en el nanoMIPS se consigue que la penalización sea de un ciclo para saltos tomados y de ningún ciclo para saltos que no se toman. Mientras se decodifica y se hace efectivo el salto, se va buscando la siguiente instrucción. Si finalmente el salto no se toma y la predicción se ha acertado, se continúa normalmente y no se ha perdido ningún ciclo. Si por el contrario el salto se toma y la predicción falla, se busca la instrucción destino y se ha perdido un ciclo haciendo un trabajo que no era necesario, pero que se hubiera perdido igualmente si no se hubiera realizado la predicción. Es decir, gracias a la predicción de salto no tomado sólo se tiene penalización por salto cuando los saltos se toman (la predicción falla).

En cuanto a la técnica **software** para la resolución de los riesgos de control, de nuevo es responsabilidad del compilador y se suele denominar técnica de salto retardado o de relleno de ranura. Con esta técnica se evita la penalización por salto introduciendo justo a continuación de la instrucción de salto, instrucciones que se van a ejecutar en cualquier caso. Es decir, la idea básica es conseguir que el procesador realice trabajo útil mientras se resuelve el salto. . Esta instrucción no puede ser otro salto y puede ser una instrucción anterior a la de salto, la instrucción

destino del salto o una instrucción del camino que se sigue cuando el salto no se toma.

▼ ¿Qué predicciones estáticas existen?

• El opcode de la instrucción de salto, ya que hay estudios que demuestran que algunos tipos de saltos se toman con más probabilidad que otros (esto tiene que ver con el estilo de programación de los desarrolladores y con el diseño de los compiladores y cómo traducen ciertas estructuras de alto nivel a lenguaje ensamblador).

• La condición de la instrucción de salto (si un registro es igual a 0 o diferente de 0, si dos registros son iguales o diferentes, etc), ya que hay estudios que demuestran que algunas condiciones se cumplen en muchas más ocasiones que otras (por los mismos motivos mencionados para el opcode).

• La longitud y la dirección del desplazamiento del salto, ya que esto permite detectar los saltos al final de los bucles fácilmente. La norma general suele ser que los saltos hacia atrás (los que están al final del bucle) se predicen como tomados, y los saltos hacia delante, se predicen como no tomados.

• La estructura del programa, proporcionando un interfaz al compilador para predecir los saltos basándose en una comprensión a alto nivel de la estructura de la aplicación. Un ejemplo son las denominadas heurísticas de Ball y Larus, un conjunto de nueve reglas que permiten tener en cuenta los lenguajes de programación habituales y los comportamientos más usuales de las aplicaciones para que el compilador pueda predecir con la información de alto nivel de la estructura del programa si los saltos condicionales se van a tomar o no.

▼ ¿Qué alternativas existen para tratar los saltos condicionales?

- Flujos múltiples.

  ▼ ¿Cómo funcionan los flujos múltiples?

  Es una solución burda, por que consiste es duplicar las partes iniciales del cauce y dejar que este capte las dos instrucciones, utilizando los dos caminos. Esta aproximación tiene dos problemas:

  • Con cauces múltiples hay retardos debidos a la competencia por el acceso a los registros y a la memoria.

• Pueden entrar en el cauce (en cualquiera de los dos flujos) instrucciones de salto adicionales antes de que se resuelva la decisión del salto original. Cada una de esas instrucciones exige un flujo adicional.

- Pre-captar el destino del salto.

▼ ¿Cómo funciona la pre-captación de destino?

Cuando se identifica un salto condicional, se precapta la instrucción destino del salto, además de la siguiente a la de salto. Se guarda entonces esta instrucción hasta que se ejecute la instrucción de salto. Si se produce el salto, el destino ya habrá sido precaptado.

- Buffer de bucles.

▼ ¿Cómo funciona el 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. En ese caso, la siguiente instrucción se capta del buffer.

▼ ¿Qué utilidades tiene el buffer de bucles?

El buffer de bucles tiene tres utilidades:

1. Con el uso de precaptación, el buffer de bucles contendrá algunas instrucciones que secuencialmente están después de la dirección de donde se capta la instrucción actual. De este modo, las instrucciones que se capten secuencialmente estarán disponibles sin necesitar el tiempo de acceso a memoria habitual.
2. Si ocurre un salto a un destino a solo unas pocas posiciones más allá de la dirección de la instrucción de salto, el destino ya estará en el buffer. Esto es útil para el caso bastante común de las secuencias IF-THEN e IF-THEN-ELSE
3. Esta estrategia se acomoda particularmente bien al tratamiento de bucles, o iteraciones, de ahí el nombre de buffer de bucles. Si el buffer de bucles es lo suficientemente grande como para contener

todas las instrucciones de un bucle, entonces esas instrucciones solo han de ser captadas de la memoria una vez, durante la primera iteración. En las siguientes iteraciones, todas las instrucciones necesarias se encuentran ya en el buffer.

El buffer de bucles es similar en principio a una caché de instrucciones. Las diferencias son que el buffer de bucles solo guarda instrucciones consecutivas y que es mucho más pequeño en tamaño, y por tanto de menor coste.

- Predicción de saltos.

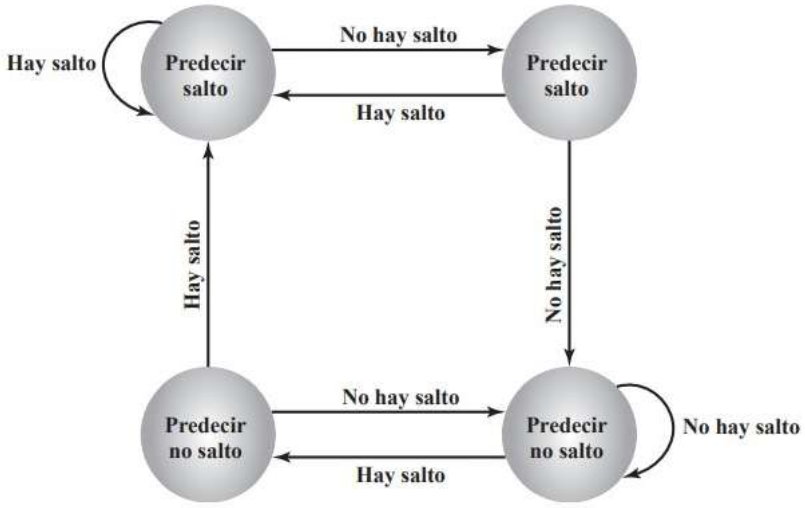
▼ ¿Cómo funciona la predicción de salto?

Se predice en base a una técnica a donde se hará el salto.

▼ ¿Qué técnicas existen? Explique cada una.

- Predecir que nunca se salta.
- Predecir que siempre se salta.
- Predecir según el código de operación: dependiendo del codop el procesador predice lo que sucederá.
- Conmutador saltar/no saltar: se guarda en un bit si el resultado del anterior salto se realizo o no. Con un único bit de historia, ocurrirá un error en la predicción dos veces en cada uso del bucle: una vez cuando se entra al bucle y otra cuando se sale de él. Si se usan dos bits, se pueden emplear para registrar el resultado de las dos últimas veces que se ejecutó la instrucción asociada, o para registrar el estado de alguna otra forma.

▼ ¿Cómo es el diagrama de estados para el conmutador de saltar/no saltar?



- Tabla de historia de saltos: La tabla de historia de saltos es una pequeña memoria caché asociada a la etapa de captación de instrucción del cauce. Cada elemento de la tabla consta de tres campos: la dirección de una instrucción de salto, un determinado número de bits de historia que guardan el estado de uso de esa instrucción, e información sobre la instrucción destino. En la mayoría de los proyectos e implementaciones, este tercer campo contiene la dirección de la instrucción destino. Otra posibilidad es que el tercer campo contenga la propia instrucción destino.

▼ ¿Cuáles son estáticas y cuales son dinámicas?

Las tres primeras soluciones son estáticas: no dependen de la historia de la ejecución que haya tenido lugar hasta la instrucción de salto condicional. Las dos últimas aproximaciones son dinámicas: dependen de la historia de la ejecución.

- Salto retardado.

▼ ¿Cómo funciona el salto retardado?

Se pueden mejorar las prestaciones de un cauce reordenando automáticamente
las instrucciones de un programa, de manera que las instrucciones de salto tengan lugar después de lo realmente deseado.

▼ ¿Cómo es la segmentación en el Intel 80486?

El cause del Intel 80486 tiene cinco etapas:

• Captación (Fetch): las instrucciones se captan de la caché o de la memoria externa y se colocan en uno de los dos buffers de prebúsqueda de 16 bytes. El objetivo de la etapa de captación es llenar los buffers de prebúsqueda con nuevos datos tan pronto como los viejos sean tratados por el decodificador de instrucciones. Dado que las instrucciones son de longitud variable (de uno a once bytes sin contar prefijos), el estado del prebuscador relativo a las otras etapas del cauce varía de instrucción en instrucción. En promedio se captan alrededor de cinco instrucciones con cada carga de 16 bytes [CRAW90]. La etapa de captación opera independientemente de las otras etapas para mantener llenos los buffers de prebúsqueda.

• Etapa de decodificación 1: el código de operación y la información referente al modo de direccionamiento se decodifican en la etapa D1. La información necesaria, así como la información sobre la longitud de la instrucción, está contenida a lo sumo en los tres primeros bytes de la instrucción. De ahí que se pasen tres bytes desde los buffers de prebúsqueda a la etapa D1. El decodificador D1 puede entonces indicar a la etapa D2 que capture el resto de la instrucción (desplazamiento y dato inmediato), que no está involucrado en la decodificación de D1.

• Etapa de decodificación 2: la etapa D2 convierte cada código de operación en señales de control para la ALU. También controla el cálculo de los modos de direccionamiento más complejos.

• Ejecución (Execute, EX): esta etapa incluye operaciones de la ALU, acceso a caché y actualización de registros.

• Escritura (Write Back, WB): esta etapa, cuando es necesaria, actualiza los registros e indicadores de estado modificados durante la etapa de ejecución precedente. Si la instrucción en curso actualiza la memoria, el valor calculado se envía al mismo tiempo a la caché y a los buffers de escritura de la interfaz del bus. Mediante el uso de dos etapas de decodificación, el cauce puede mantener una productividad cercana a una instrucción por ciclo de reloj. Las instrucciones complejas y los saltos condicionales pueden reducir esta velocidad.

▼ ¿Qué se debe cambiar para implementar operaciones en punto flotante a la segmentación de cause vista?

Primero hay que añadir la lógica para soportar la latencia de la división en punto flotante y que se pueda completar en la etapa X, dado que es una operación particularmente lenta. Ahora las operaciones no serán mas lineales sino que tardaran dependiendo del tipo de operación que se realiza. También se necesitan añadir nuevas piezas de hardware: un sumador y restador en coma flotante, un multiplicador y un divisor, esto implica que se deben incluir nuevos registros de segmentación y que se añaden nuevas conexiones entre todos los registros para implementar todos los adelantamientos que sean necesarios.

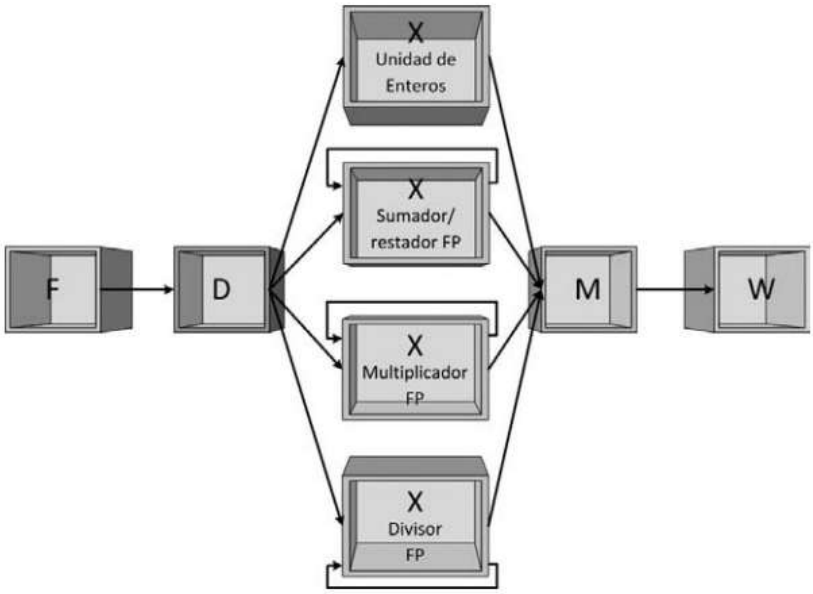


FIGURA 1.37

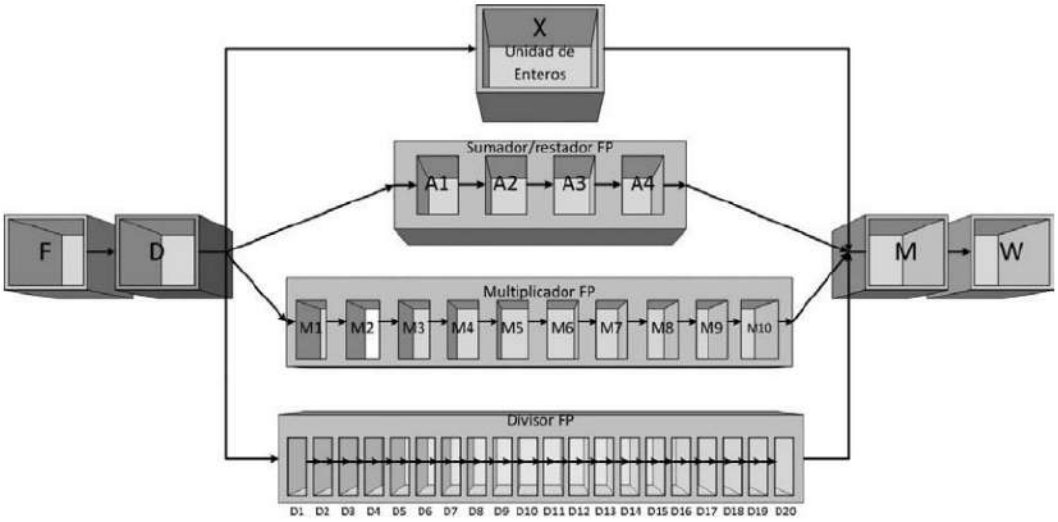
▼ ¿Qué dificultad ve en la implementación anterior?

Este tipo de diseño es sencillo, pero plantea un problema de rendimiento con los riesgos estructurales por las unidades funcionales en coma flotante. Si se tiene que el sumador/restador tiene una latencia de 4 ciclos, el multiplicador de 10 ciclos y el divisor de 20 ciclos, ¿qué ocurre si en un código aparecen dos sumas en coma flotante seguidas? Obviamente, la segunda suma tendrá que esperar 4 ciclos (en lugar de 1) a que termine la primera.

▼ ¿Qué solución tiene este problema?

Para evitar todas estas paradas por riesgos estructurales debidos a las unidades funcionales, se suele optar por incluir unidades funcionales de coma flotante segmentadas dentro de la ruta de datos.

Es decir que se añaden mas etapas dependiendo de la operación a realizar.



Unidad 6 - Computadoras de repertorio reducido de instrucciones.

`Clase 6 filminas, Hojas 103 a 133 del libro Stallings`

▼ ¿Cuáles son los avances mas importantes desde el nacimiento de la computadora?

- El concepto de familia, que fue introducido por IBM en su System/360 en 1964. El concepto de familia hace referencia a separar la arquitectura de una maquina de su implementación. Ofreciendo así un conjunto de computadores, con distintas características en cuanto a precio y prestaciones, pero se le da al usuario la misma arquitectura.
- Unidad de control microprogramada, que fue introducida por IBM en 1964 en la linea 360. Facilitando así la tarea de diseñar e implementar la unidad de control y da soporte al concepto de familia
- Memoria cache, que se introdujo en los modelos 85 del s/360 de IBM en 1968. Mejorando las prestaciones de las computadoras de una manera espectacular.

- La segmentación de cause, una manera de introducir paralelismo en la naturaleza esencialmente secuencial de un programa constituido por instrucciones máquina.
- Múltiples procesadores
- Arquitectura de computador de repertorio reducido de instrucciones.

▼ ¿Qué es la arquitectura RISC?

Risc es Reduced Instruction Set Computer o arquitectura de computador de repertorio reducido de instrucciones.

▼ ¿Qué características posee?

- Un gran numero de registros de uso general o el uso de tecnologías para mejorar la utilización de los registros.
- Un repertorio de instrucciones limitado y sencillo.
- Un énfasis en la optimización de la segmentación de instrucciones.

▼ ¿Qué dificultades enfrentaba el software?

Conforme el coste del hardware ha disminuido, el coste relativo del software ha ido creciendo. Junto con esto, la escasez permanente de programadores ha hecho subir los costes del software en términos absolutos. De ahí que el coste principal del ciclo de vida de un sistema sea el software, no el hardware. Otro elemento que se añade al coste y a otros inconvenientes es la falta de fiabilidad: es frecuente que los programas, tanto de sistema como de aplicación, continúen mostrando nuevos errores después de años de funcionamiento.

▼ ¿Qué solución se encontró?

La respuesta de los investigadores y de la industria fue desarrollar lenguajes de programación de alto nivel más potentes y complejos. Estos lenguajes de alto nivel (High-Level Languages, HLL) permiten al programador expresar los algoritmos de manera más concisa, se encargan de gran parte de los pormenores, y a menudo favorecen de manera natural la programación estructurada o el diseño orientado a objetos.

▼ ¿Qué problema originaron los HLL?

Desgraciadamente, la solución ocasionó otro problema, conocido como el salto semántico, la diferencia entre las operaciones que proporcionan los HLL y las que proporciona la arquitectura del computador. Los supuestos síntomas de este salto o hueco incluían ineficiencia de la ejecución, tamaño excesivo del programa en lenguaje máquina, y complejidad de los compiladores.

▼ ¿Qué es arquitectura CISC?

CISC significa Computador con Conjunto de Instrucciones Complejas. Los procesadores basados en CISC, tienen un conjunto de instrucciones que se caracteriza por ser muy amplio y permitir operaciones complejas entre operandos situados en la memoria o en los registros internos.

▼ ¿Qué finalidades tiene?

- Facilitar el trabajo de los compiladores.
- Mejorar la eficiencia de ejecución, dado que las secuencias complejas de operaciones pueden ser implementadas por microcódigo.
- Dar soporte la lenguajes de alto nivel mas sofisticados.

▼ ¿Qué características se estudiaron en la relacion de los HLL y los repertorios de instrucciones?

- Operaciones realizadas: determinan las funciones que lleva a cabo el procesador y su interacción con la memoria.
- Operandos usados: los tipos de operandos y su frecuencia de uso determinan la organización de memoria para almacenarlos y los modos de direccionamiento para acceder a ellos.
- Secuenciamiento de la ejecución: determina la organización del control y del cauce segmentado.

▼ ¿A que conclusiones se llego?

Primero se descubrió que las asignaciones, los elementos condicionales como if y loop eran muy usados, además que el llamado y retorno de procesos que además consumían mucho tiempo para cargarse/descargarse. Con respecto a los operandos, las principales variables usadas eran escalares locales por eso la optimización debía focalizarse en el acceso a variables locales.

Finalmente en cuanto al secuenciamianto de ejecucion, se concluyo que los procedimientos eran muy utilizados y poco eficientes, consumian mucho tiempo y dependian del numero de parametros tratados y del nivel de anidamiento. Se encontraron algunos patrones, La mayoría de los programas no tienen una larga secuencia de llamadas seguida por la correspondiente secuencia de retornos. La mayoria de variables son locales, y las referencias a operandos estan muy localizadas.

▼ Consecuentemente a esto, ¿Qué habia que cambiar en las arquitecturas?

Se puede ofrecer mejor soporte para los HLL optimizando las prestaciones de las características más usadas y que más tiempo consumen.

• Usar un gran número de registros:
• Optimizar las referencias a operandos
• Prestar cuidadosa atención al diseño de los cauces de instrucciones:
• Predicción de bifurcaciones, etc.
• Es recomendable un repertorio con instrucciones simples (reducido)

▼ ¿Por que los registros son importantes?

Los resultados relacionados a los procedimientos arrojaron que es muy deseable lograr un acceso rápido a los operandos. Además hay un número significativo de accesos a operandos por cada sentencia de HLL. Si juntamos estos resultados con el hecho de que la mayoría de los accesos se hacen a datos escalares locales, se puede pensar en una fuerte dependencia del almacenamiento en registros.

La razón de que esté indicado dicho almacenamiento en registros es que estos constituyen el dispositivo de almacenamiento más rápido disponible, más que la memoria principal y que la caché.

▼ ¿Qué aproximaciones se tomaron para mejorar su rendimiento?

La implementación de un banco de registros que es pequeño físicamente, está en el mismo chip que la ALU y la unidad de control, y emplea direcciones mucho más cortas que las de la caché y la memoria. Por tanto, se necesita una estrategia que permita que los operandos a los que se acceda con mayor frecuencia se encuentren en registros y se minimicen las operaciones registro-memoria.

Son posibles dos aproximaciones básicas, una basada en software y la otra en hardware. La aproximación por software consiste en confiar al compilador la maximización del uso de los registros. El compilador intentará asignar registros a las variables que se usen más en un periodo de tiempo dado. Esta solución requiere el uso de sofisticados algoritmos de análisis de programas. La aproximación por hardware consiste sencillamente en usar más registros de manera que puedan mantenerse en ellos más variables durante períodos de tiempo más largos.

▼ ¿Qué es una ventana de registro?

En primer lugar, un procedimiento típico emplea solo unos pocos parámetros de llamada y variables locales. En segundo lugar, la profundidad de activación de procedimientos fluctúa dentro de un rango relativamente pequeño. Para explotar estas propiedades, se usan múltiples conjuntos pequeños 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 actual desde el procedimiento que lo llamó, y los resultados a devolver a este. 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.

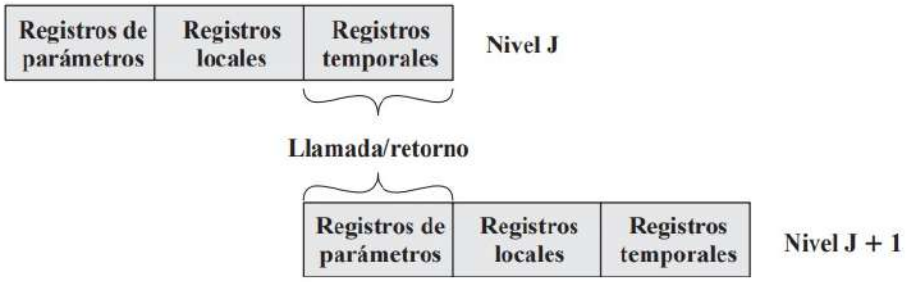
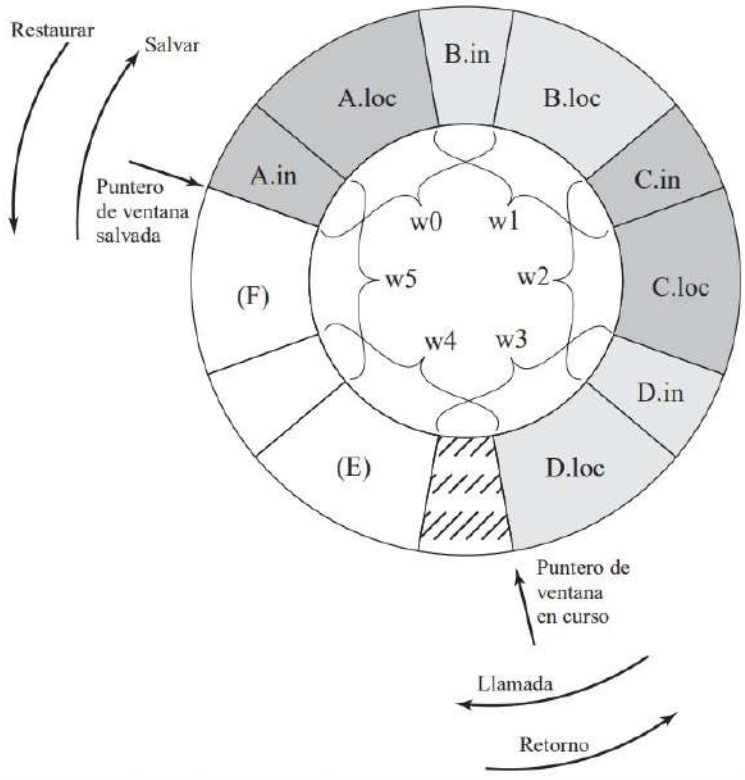


Figura 13.1. Ventanas de registro solapadas.

▼ ¿Qué es un buffer circular?

Almacena las distintas ventanas de los procesos en una estructura circular.



▼ ¿Qué pasa si no se tiene mas lugar?

Se produce una interrupción y se guardan los datos mas viejos en el buffer en memoria, una vez ocurre el retorno se produce otra interrupción que restaura los datos guardados.

- ¿Cómo se solucionan las variables locales?

  Con el esquema de ventanas, propuesto en la pregunta anterior.

- ¿Cómo se solucionan las variables globales?

  Existen dos alternativas, la primera es que el compilador asigne posiciones de memoria a las variables declaradas como globales en un HLL, y que todas las instrucciones máquina que referencien esas variables usen operandos referenciados en memoria. Esto es sencillo desde los puntos de vista hardware y software (compilador). No obstante, para variables globales a las que se accede frecuentemente, este esquema es ineficiente. Una alternativa es incorporar al procesador un conjunto de registros globales. Estos registros serían fijos en cuanto a su número y accesibles por todos los procedimientos. Se puede usar un esquema de numeración unificado para simplificar el formato de instrucción. Esto conlleva un hardware añadido, que se encarga de adaptar la división en el direccionamiento de los registros. Además, el compilador ha de decidir qué variables globales deben ser asignadas a registros.

- ¿Cuáles son las diferencias entre el cache y un banco de registros?

  Como conclusión el banco de registros es mejor por que sus datos se pueden conseguir sin tener que realizar un direccionamiento a memoria.

Banco de registros amplio	Caché
Todos los datos escalares locales	Datos escalares locales recientemente usados
Variables individuales	Bloques de memoria
Variables globales asignadas por el compilador	Variables globales usadas recientemente
Salvaguarda/restauración basadas en la profundidad de anidamiento	Salvaguarda/restauración basadas en el algoritmo de reemplazo
Direccionamiento a registros	Direccionamiento a memoria

- ¿Qué optimizaciones de los registros se hace con el compilador?

Se utiliza la optimización “coloreo de grafos”.

Por lo general se sigue el siguiente enfoque. Cada cantidad del programa candidata para residir en un registro se asigna a un registro simbólico o virtual. El compilador entonces asigna el número ilimitado de registros simbólicos a un número fijo de registros reales. Los registros simbólicos cuya utilización no se solape pueden compartir el mismo registro real. Si en una parte concreta del programa hay más cantidades a tratar que registros reales, algunas de las cantidades se asignan a posiciones de memoria. Para colocar temporalmente las cantidades en los registros durante las operaciones de cálculo se usan instrucciones de carga y almacenamiento.

▼ ¿Por que el CISC no parecía el camino adecuado?

Uno de los objetivos de la existencia del CISC era simplificar la tarea del compilador, quien en teoría al tener instrucciones parecidas a los lenguajes de alto nivel se sentiría mas cómodo y se simplificaría la tarea. Pero los investigadores del RISC encontraron que esto no era verdad, por que las instrucciones a menudo eran tan complejas que eran difíciles de ser aprovechadas al máximo por el compilador. El compilador tenia que descubrir cuales eran los casos en los que se justificaba a la perfección el uso de cada instrucción.

También las tareas de optimización y reducción del código eran muy complicadas con un repertorio CISC.

La ventaja que daba el CISC era tener programas mas pequeños, lo cual dejo de ser una ventaja a medida que los costos en memoria disminuían. Era mas importante tener un programa pequeño para reducir los fallos de paginas. Pero los programas en CISC estaban lejos de ser pequeños ni mucho menos, por que por mas que parecían muy pequeños en su representación simbólica (cantidad de líneas que ocupaba) eran muchos mas los bites que utilizaba.

Otro ventaja que ofrecía en teoría el CISC era que tenia programas mas rápidos, lo cual a priori era cierto, pero a su vez se necesitaba una unidad de control completa mas compleja y/o la memoria de control del microprograma más grande, para acomodar un repertorio de instrucciones más rico. También se encontró la prueba de que las instrucciones no eran mas rápidas por que fueran complejas, sino por que residían en almacenamiento de control rápido (cache de instrucciones).

▼ ¿Cuáles son las características de las arquitecturas de repertorio reducido de instrucciones?

Aunque ha habido diferentes aproximaciones a la arquitectura de repertorio reducido de instrucciones, hay ciertas características comunes a todas ellas:
• Una instrucción por ciclo.
• Operaciones registro a registro.
• Modos de direccionamiento sencillos.
• Formatos de instrucción sencillos.
• Diseño cableado (sin microcódigo).
• Formato de instrucción fijo.
• Mayor tiempo/esfuerzo de compilación.

▼ ¿Qué significa una instrucción por ciclo?

La primera característica enumerada es que se ejecuta una instrucción máquina cada ciclo máquina. Un ciclo máquina se define como el tiempo que se tarda en captar dos operandos desde dos registros, realizar una operación de la ALU y almacenar el resultado en un registro. Así, las instrucciones máquina de un RISC no deberían ser más complicadas que las microinstrucciones de las máquinas CISC, y deberían ejecutarse más o menos igual de rápido.

▼ ¿Qué significa operaciones registro a registro?

Una segunda característica es que la mayoría de las operaciones deben ser del tipo registro a registro con la excepción de sencillas operaciones LOAD y STORE para acceder a memoria. Esta forma de diseño simplifica el repertorio de instrucciones y por tanto la unidad de control. . Otra ventaja es que tal arquitectura fomenta la optimización del uso de registros, ya que los operandos accedidos frecuentemente permanecen en el almacenamiento de alta velocidad.

▼ ¿Qué significa que tenga modos de direccionamiento sencillos?

Una tercera característica es el uso de modos de direccionamiento sencillos. Casi todas las instrucciones RISC usan direccionamiento sencillo a registro. Se pueden incluir varios modos adicionales, como el desplazamiento y el relativo al contador de programa. Otros modos más complejos se pueden sintetizar por

software a partir de los simples. Nuevamente, esta característica de diseño simplifica el repertorio de instrucciones y la unidad de control.

- ¿Qué significa que tenga formatos de instrucciones sencillos?

  Una última característica común es el uso de formatos de instrucción sencillos. Generalmente, solo se usa un formato o unos pocos. 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. Este tipo de diseño tiene varias ventajas. Con campos fijos, la decodificación del código de operación y el acceso a los operandos en registros puede tener lugar simultáneamente. Los formatos sencillos simplifican la unidad de control. Se optimiza la captación de instrucciones ya que se captan unidades de una palabra de longitud. La alineación en el límite de una palabra también significa que una única instrucción no traspasa los límites de una página.

- ¿Existen los modelos de RISC puros?

  No. Se mantienen las bases del modelo RISC, pero se agregan algunas operaciones que pueden ser mas complejas.

- ¿A que hace referencia la controversia entre RISC y CISC?

  Se intento comparar a los CISC y RISC desde dos puntos de vista principales, el cuantitativo que hace referencia al tamaño del programa y a su velocidad de ejecución y el cualitativo que es la forma en la que dan soporte a los lenguajes de alto nivel y que tan optimo es el uso de los recurso del sistema. A partir de esto surge el problema de las comparaciones, y es que no existe un par de maquinas de RISC y CISC directamente comparables, primero por que no hay un conjunto de programas de prueba definitivo, segundo existe una dificultad para separar los efectos del hardware, de los efectos logrados por las optimizaciones realzadas por el compilador y finalmente por que no existen maquinas reales en donde comparar, por que las que existen son en su mayoría de pruebas o simplemente “juguetes”, no son productos comerciales. La mayoría de las maquinas son una mescla entre RISC y CISC.

  En los años más recientes, la controversia RISC frente a CISC se ha sosegado en gran parte. Ello se debe a que ha habido una convergencia progresiva de las tecnologías. Conforme la densidad de integración y la velocidad bruta del hardware han aumentado, los sistemas RISC se han vuelto más complejos. Al mismo tiempo,

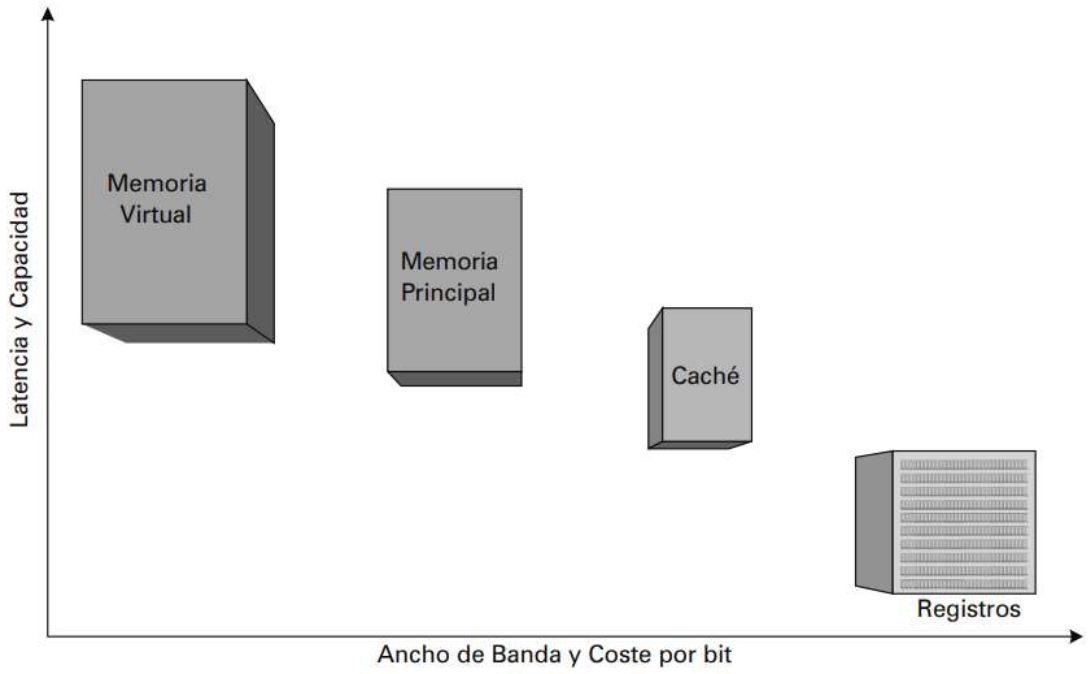
en un esfuerzo por exprimir las prestaciones al máximo, los diseños CISC se han concentrado en cuestiones asociadas tradicionalmente a los RISC, tales como un mayor número de registros de propósito general y un énfasis creciente en el diseño del cauce de instrucciones.

Unidad 7 - memoria

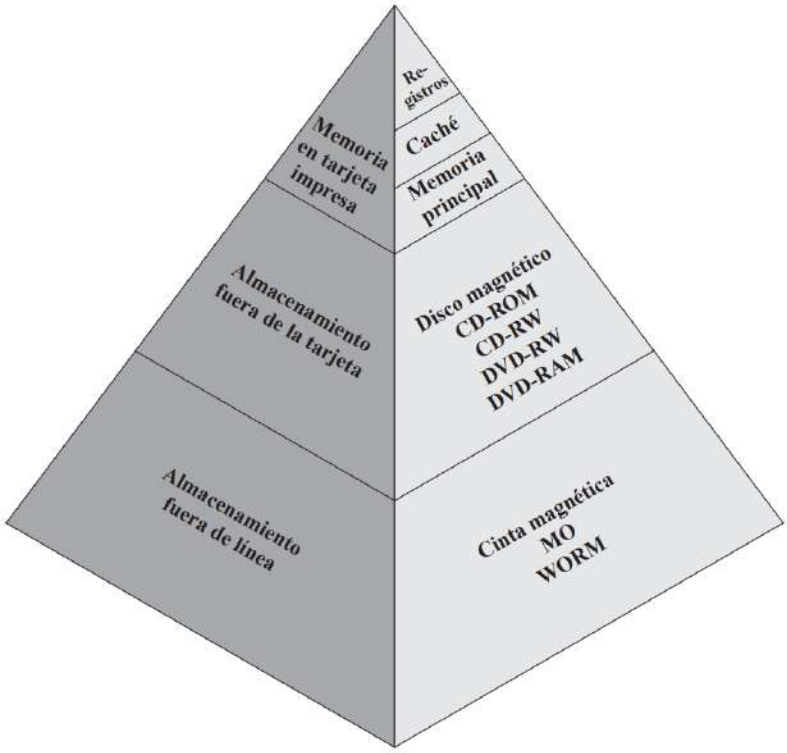
`Clase 7 filminas, Hojas 485 a 523 del libro Stallings`

▼ ¿Qué es la jerarquía de memoria?

Los programadores desean acceder a cantidades ilimitadas de memoria rápida, lo cual es muy costoso, para poder alcanzar la velocidad del procesador y poder tener el mejor desempeño. Para llegar a un punto de eficiencia al precio mas optimo, lo que se hace es combinar memorias rápidas con memorias mas lentas. Esto es la jerarquía de memoria que organiza en distintos niveles la memoria física.



Este cuadro muestra como se comportan los distintos elementos de la jerarquía. A menor latencia, mayor es la capacidad y menor el costo. A menor latencia menor es la capacidad y mayor el costo.



▼ ¿Cuál es el objetivo de esta estructura organizacional?

El objetivo es conseguir una estructura de memoria de gran capacidad, con el coste mas bajo de la jerarquía, pero con la latencias comparables al del nivel mas bajo.

▼ ¿Qué propiedades debe de cumplir la jerarquía?

Son dos las propiedades que debe cumplir para poder funcionar.

- La primera, denominada inclusión, implica que cualquier información que se contenga en un nivel, también debe estar disponible en los niveles inferiores a el. Por ejemplo un dato que reside en un registro debe de estar en la memoria principal, pero un dato que reside en memoria principal, puede o no estar en un registro.
- La segunda, coherencia, implica que todos los niveles que contengan copias de la misma información deben ser coherentes, es decir tener almacenado el mismo valor.
- Además debe existir una correspondencia directa de direcciones entre los distintos niveles.

▼ ¿Qué es un fallo?

Un fallo se produce cuando buscamos un dato de información y no se encuentra en un dado nivel de la jerarquía, entonces debemos descender de nivel para buscarla.

▼ Entonces... ¿Por que funciona?¿Que principios sigue?

Sigue dos principios para poder funcionar.

- El primero, la localidad espacial, que dicta que el procesador tiene tendencia a referenciar elementos de memoria cercanos a los últimos que han sido referenciados.
- El segundo, el de localidad temporal, señala una tendencia del procesador a referenciar elementos de memoria que han sido referenciados recientemente.

▼ ¿Cuáles son las etapas que forman un acceso a memoria completo?

- El procesador genera la dirección virtual de la palabra que se debe leer o escribir.
- Se traduce esa dirección virtual a un dirección física comprensible para la jerarquía de memoria. Si se logra esta traducción es por que la palabra que se esta buscando se encuentra en memoria principal, con esta dirección se puede acceder a la memoria cache, si la palabra buscada se encuentra en ese nivel, entonces el acceso a memoria finaliza.
  - Si por el contrario se produce un fallo entonces se busca la palabra en el siguiente nivel mas bajo.

  ▼ ¿Qué tipos de fallo de memoria cache existen?

  Existen 3:

  - Los iniciales, son fallos obligatorios, dado que la primera vez que se referencie a una palabra esta no estará en cache.
  - De capacidad, producido por la falta de espacio en la cache, cuando esto sucede se remplazan los bloques que sean necesarios.
  - De conflicto, producto de que distintos bloques tengan la misma localización en memoria cache y se van remplazando unos a otros

debido de esta colisión.

- Sea cual sea el tipo de fallo, su resolución es buscar el dato/palabra en memoria principal.
- Para esto hay que consultar al controlador de la memoria principal que mapea la dirección física con la ubicación física en los chips de dram. Si esto se puede resolver, se mueve todo el bloque a cache y se finaliza con éxito. En el caso contrario..
- Significa que la pagina o palabra no se ubica en memoria principal, por lo tanto se realiza una interrupción y el procesador continua con su trabajo, hasta que el dato es enviado desde memoria virtual.
  - En el caso de existir dos niveles de cache, el método de acceso es el mismo , salvo que cuando se produce un fallo en el nivel primero se intenta acceder a la palabra en el siguiente nivel. Si la palabra que ocasiono el fallo en el nivel uno, se encuentra en el nivel dos, entonces se la envía al nivel uno. Caso contrario se busca en memoria.

▼ ¿Qué es la memoria cache?

Es una memoria pequeña de alta velocidad, que se encuentra entre el procesador y la memoria principal. También podría ubicarse dentro del procesador. Y puede tener múltiples niveles.

▼ ¿Cuál es la unidad de transferencia entre el CPU y la cache?

Se transfieren palabras.

▼ ¿Cuál es la unidad de transferencia entre la cache y la memoria principal?

Bloques de múltiples palabras.

▼ ¿Cómo esta organizada la memoria cache?

La cache esta divida en líneas/marcos, cada lineal puede contener un determinado numero de palabras mas la etiqueta.

▼ ¿Y la memoria principal?

La memoria principal cuenta con 2 a la n palabras direccionables, teniendo cada palabra una única dirección de n bits, la memoria esta divida en un

numero de bloques de longitud fija, de determinado numero de palabras por bloque. El numero de líneas de la memoria cache es considerablemente menor a la cantidad de bloques de memoria principal.

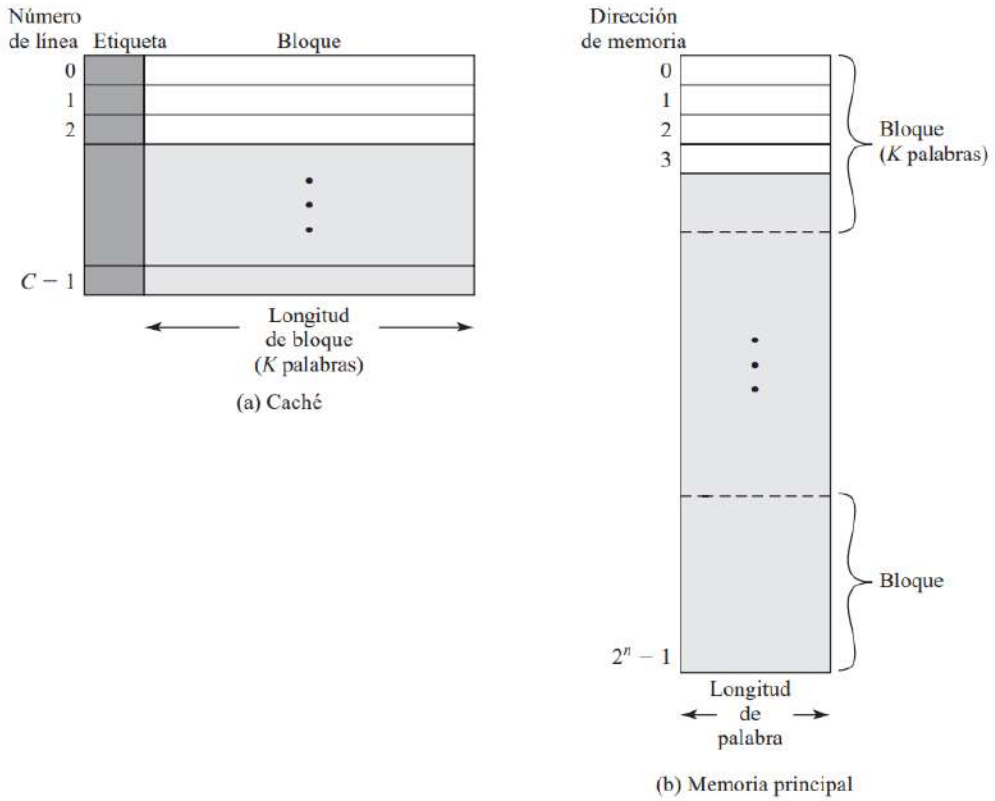


Figura 4.4. Estructura de memoria caché/principal.

▼ ¿Cómo se ubica a un determinado bloque?

Cada línea de memoria cache contiene una etiqueta que identifica que bloque particular almacena. La etiqueta es usualmente una porción de la dirección de memoria principal.

▼ ¿Qué es un acierto y un fallo de cache?

El acierto o hit sucede cuando el dato solicitado se encuentra en la cache. El fallo o miss se produce cuando el dato solicitado no esta en la cache y se tiene que bajar de nivel para buscarlo. Los fallos de cache se gestionan mediante hardware y causan que el procesador se detenga hasta que el dato este disponible.

▼ ¿Cuál es el tiempo para servir a un fallo?

El tiempo para servir a un fallo depende de dos variables, el tiempo de latencia que es el tiempo necesario para acceder a memoria. Y el ancho de banda que es la cantidad de información por unidad de tiempo que se puede transferir desde y hacia la memoria.

▼ ¿Cuánto tiempo le lleva de media al procesador hacer un acceso a memoria?

Se puede calcular el tiempo haciendo la suma del tiempo de acierto mas el tiempo de fallos de memoria. El tiempo de fallos de memoria se calcula multiplicando la tasa de fallos por la penalización por fallo. Para mejorar las prestaciones se debe reducir el tiempo en caso de acceso, reducir la tasa de fallos y reducir la penalización por fallo.

Se pueden ver dos enfoques para analizar esto, el primero supone una jerarquía perfecta. En donde no se producen fallos de memoria y el tiempo de acceso es muy bajo. Este punto de vista demuestra la velocidad que se puede alcanzar bajo las condiciones optimas, pero es irreal. Un enfoque mas real contempla una tasa de fallos del 4% para la cache L1 con una penalización de 100ns y del 6% para la cache L2 con una penalización de 115ns.

▼ ¿Qué cuestiones se deben tener en cuenta para diseñar una cache?

- El tamaño de la cache, nos gustaría que fuera lo suficientemente pequeño para que el costo por bit iguale al de la memoria principal y lo suficientemente grande como para que el tiempo de acceso medio sea cercano al de la cache sola. También existe la motivación de la superficie disponible para hacerlo mas pequeño, también esta demostrado que cuanto mas grande es mas direccionamiento requiere por lo tanto es mas lento incluso que una cache mas pequeña.

  ▼ ¿Qué otros aspectos al tamaño hay que tener en cuenta?

  Tamaño del marco y de bloque que se va a manejar, utilizar bloques de gran tamaño hace que se capture mejor la localidad espacial y se puedan reducir fallos iniciales, pero al tener bloques mas grandes la penalización por fallo aumenta, dado que se tienen que traer bloques mas grandes.

  Otro aspecto al tener en cuenta es la unificación entre datos e instrucciones o su división.

- Política de ubicación, dado que exciten menos líneas de cache que cantidad de bloques se necesita un algoritmo que haga corresponder bloques de memoria principal a líneas de cache.

  ▼ ¿Qué políticas de ubicación existen?

  - Correspondencia directa, en donde a cada bloque le corresponde un único marco de cache y solo puede ubicarse en ese marco. Para calcular la correspondencia se hace el calculo **dirección del bloque modulo del numero de marcos.** Por ejemplo el bloque ubicado en la posición 5 de la memoria principal se debe ubicar en el marco 1 de la cache. considerando que esta cache tiene 4 marcos el calculo es , 5 mod 4 = 1.

    ▼ ¿Qué desventaja presenta?

    Es muy sencilla y poco costosa de implementar, su principal desventaja es que solo un bloque puede ocupar un marco determinado y puede que en determinado momento de ejecución un programa necesita alocar bloques que necesitan el mismo marco de manera repetida, entonces se producen fallos constantemente.

  - Correspondencia asociativa, en donde un bloque puede almacenarse en cualquiera de los marcos existentes en la cache.

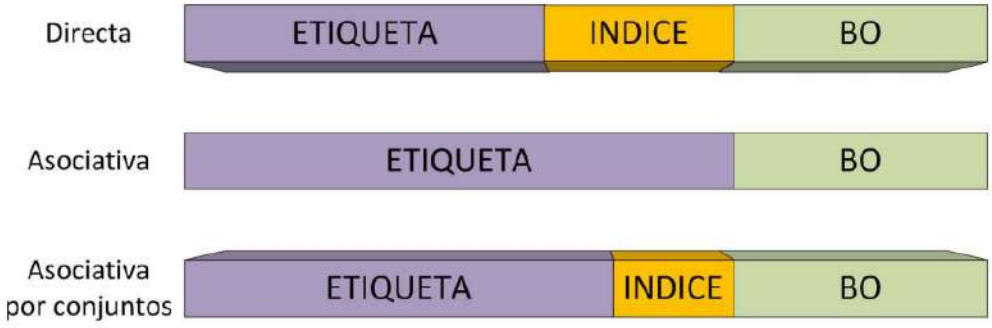
    ▼ ¿Qué desventaja presenta?

    Tiene como ventaja la flexibilidad que provee, pero tiene como desventaja que para poder determinar si un bloque esta en cache, la lógica de control debe ser capaz de examinar simultáneamente todas las etiquetas de lineal para buscar coincidencia.

  - Correspondencia por conjuntos, a un bloque de memoria principal se le asignan un conjunto de posibles marcos para alocarse. Una solución intermedia entre las ultimas dos posibilidades, para calcular en que lugar puede estar un bloque se hace dirección de bloque modulo del numero de conjuntos.

  ▼ ¿Qué se debe contemplar para identificar a una dirección física?

La interpretación de la dirección física depende del tipo de correspondencia que se utilice. En la directa necesitaremos algunos bits mas que sean las índices para localizar en que marco de cache se encuentra el dato(tantos bits como para apuntar a la cantidad de líneas) , en la correspondencia directo no necesitamos nada por lo tanto nos ahorramos unos bits que podemos usar para otra cosa, mientras que en la asociativa por conjuntos también necesitamos bits para índice para apuntar a cada uno de los conjuntos, por lo tanto necesitamos menos bits.



- Política de remplazo, cuando se produce un fallo y se necesita traer un bloque desde memoria principal a la cache se debe sacar un bloque, esta política determina que bloque salta.

  ▼ ¿Qué políticas de remplazo existe?

  - Cuando usamos correspondencia directa no tenemos alternativa mas que sacar al bloque que se encuentra en el marco que le corresponde al entrante.

  Para el resto los algoritmos se implementan sobre el hardware para conseguir mas velocidad.

  - Aleatorio, se genera un numero al azar y se desaloca ese marco.
  - LRU(Last recently used), Se escoge el marco que lleva menos tiempo sin utilizarse. Así se minimizan las posibilidades de sacar un bloque que sea necesitado a futuro.
  - Fifo(First in first out), Se remplaza el marco que lleva mas tiempo en cache.

        - LFU(Last frecuently used), se escoge el marco que experimento menos referencias.

- Política de escritura, se deben evitar las inconsistencias generadas por las escrituras de cache en la información de la memoria principal.

    ▼ ¿Qué políticas de escritura existen?

        - Escritura **inmediata**. utilizando esta técnica, todas las operaciones de escritura se hacen tanto en caché como en memoria principal, asegurando que el contenido de la memoria principal siempre es válido
        - Una técnica alternativa, conocida como **postescritura**, minimiza las escrituras en memoria. Con la postescritura, las actualizaciones se hacen solo en la caché. Cuando tiene lugar una actualización, se activa un bit ACTUALIZAR asociado a la línea. Después, cuando el bloque es sustituido, es (post) escrito en memoria principal si y solo si el bit ACTUALIZAR está activo. El problema de este esquema es que se tienen porciones de memoria principal que no son válidas, y los accesos por parte de los módulos de E/S tendrán que hacerse solo a través de la caché. Esto complica la circuitería y genera un cuello de botella potencial.

        ▼ ¿Qué se puede hacer en el caso de un fallo de escritura?

            - Memoria caché con asignación en escritura. Los fallos de escritura se comportan como los fallos de lectura, se trae el bloque que ha provocado el fallo a la memoria caché y se realiza la escritura en función de la política utilizada.
            - Memoria caché sin asignación en escritura. Cuando se produce un fallo de escritura, el bloque se modifica directamente en el siguiente nivel de la jerarquía de memoria y no se trae a la caché. Sólo se llevan a la caché los bloques que se solicitan para lectura, no para escritura.

▼ ¿Qué es una cache multinivel?

Con el aumento de densidad de integración, ha sido posible tener una caché en el mismo chip del procesador: caché on-chip. Comparada con la accesible a través de un bus externo, la caché on-chip reduce la actividad del bus externo del procesador

y por tanto reduce los tiempos de ejecución e incrementa las prestaciones globales del sistema. Cuando la instrucción o el dato requeridos se encuentran en la caché on-chip, se elimina el acceso al bus. Debido a que los caminos de datos internos al procesador son muy cortos en comparación con la longitud de los buses, los accesos a la caché on-chip se efectúan apreciablemente más rápidos que los ciclos de bus, incluso en ausencia de estados de espera. Además, durante este periodo el bus está libre para realizar otras transferencias.

Tener una segunda cache y una tercera mejoran la tasa de aciertos, pero complican la lógica de asignación, remplazo y escritura.

▼ ¿Cómo funciona la cache del Pentium 4?//Todo: refactoring

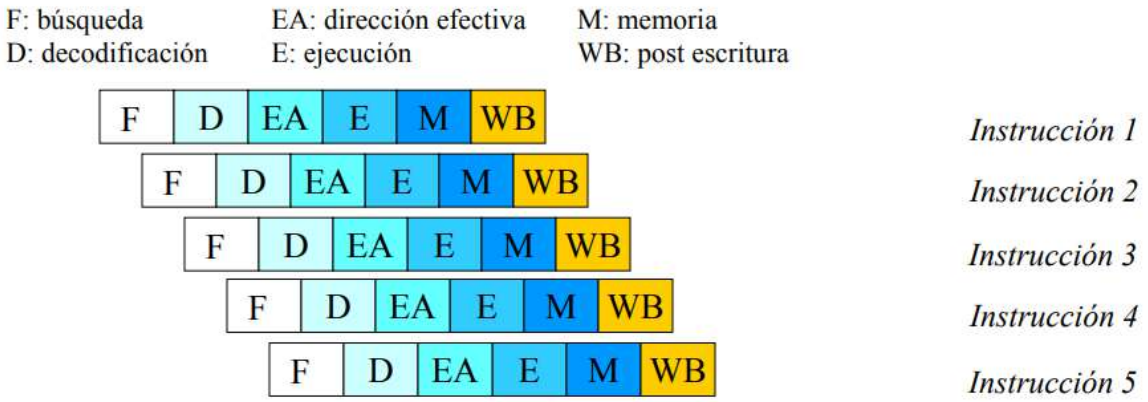
A diferencia de la organización de los modelos Pentium anteriores, así como de la mayoría de los demás procesadores, la caché de instrucciones del Pentium 4 está situada entre la lógica de decodificación de instrucciones y el núcleo de ejecución. El motivo es que el Pentium, como discutiremos con más detalle en el Capítulo 14, decodifica o traduce sus instrucciones máquina a instrucciones más sencillas de tipo RISC, denominadas micro-operaciones. El uso de micro-operaciones sencillas, de longitud fija, posibilita la utilización de segmentación superescalar y de técnicas de planificación que mejoran las prestaciones. Sin embargo, las instrucciones máquina del Pentium son difíciles de decodificar, ya que tienen un número variable de bytes y muchas opciones diferentes. Se consigue mejorar las prestaciones si dicha decodificación se realiza independientemente de la lógica de planificación y de segmentación. Volveremos sobre este tema en el Capítulo 14. La caché de datos emplea una política de postescritura: los datos se escriben en memoria principal solo cuando, habiendo sido actualizados, se eliminan de la caché. El procesador Pentium 4 puede configurarse dinámicamente para utilizar la política de escritura inmediata. La caché L1 de datos se controla por dos bits de uno de los registros de control (véase la Tabla 4.5), rotulados CD (Caché Disable: inhabilitar caché) y NW (Not Write Through: no escritura inmediata). Hay también dos instrucciones del Pentium 4 que pueden utilizarse para controlar la caché: INVD invalida la memoria caché interna e indica que se invalide la caché externa (si la hay); WBINVD postescribe e invalida la caché interna, y entonces postescribe e invalida la caché externa. Las dos cachés, L2 y L3, son asociativas por conjuntos de ocho vías, con un tamaño de línea de 128 bytes.

Unidad 8 - Procesadores superescalares

`Clase 8 filminas. Hojas 527 a 557 del libro Stallings`

▼ ¿Qué es la super-segmentación?

Se parte de la base de que muchas operaciones no necesitan de todo un ciclo de reloj. Tomando este se puede obtener mayores prestaciones si se subdividen el ciclo de reloj en sub-intervalos. Que resulta en una mayor frecuencia de ciclo de reloj. Es básicamente dividir las etapas grandes del cause segmentado en subetapas mas pequeñas y se transmiten los datos a la mayor velocidad del ciclo de reloj.



Ejecución super-segmentada de instrucciones

El tiempo para las instrucciones individuales no varía, pero se aumenta el grado de paralelismo.

▼ ¿Cómo funciona el enfoque superescalar?

El enfoque superescalar plantea que se puede completar mas de una instrucción simultáneamente, duplicando algunas o todas las partes del cpu o la alu.

Podría ser:

- Captar múltiples instrucciones al mismo tiempo.
- Ejecutar sumas y multiplicaciones simultáneamente.
- Ejecutar carga o almacenamiento, mientras se esta haciendo otra operación en la ALU.

Aumenta el grado de paralelismo y la aceleración de la maquina aumenta, ya que se ejecutan mas instrucciones en paralelo.

▼ ¿Qué limitaciones tiene el procesador superescalar?

Existen 5 limitaciones:

- Dependencia de datos verdadera es cuando una instrucción depende de un dato que aun no existe o sobre el cual no se termino de operar. Para evitar esta dependencia se retrasa a la segunda instrucción tantos ciclos de reloj como sea necesario para eliminar la dependencia.
- Dependencia relativa al procedimiento están relacionadas a las bifurcaciones, estas afectaban ya los procesadores de cauce segmentado, en el caso de los de cauce superescalar las consecuencias son aun mas graves ya que se pierden mayor numero de posibilidades de empezar a ejecutar una instrucción en cada retardo.
  Si se utilizan instrucciones de longitud variable, surge otro tipo de dependencia puesto que no se conoce la longitud de una instrucción concreta esta ha de decodificarse al menos parcialmente antes de captar la siguiente instrucción, esto impide la captación simultanea necesaria en un cause superescalar, por esto es mas fácil de implementar el superescalar cuando se cuenta con arquitecturas RISC
- Conflicto en los recursos, es una pugna de dos o mas instrucciones por el mismo recurso al mismo tiempo. Puede ser por la memoria, las caches, los buses los puertos del banco de registros y las unidades funcionales.
- Dependencia de salida, Podría suceder que una instrucción capte un valor equivocado debido a que una instrucción anterior aun no ha guardado el valor que debe ser usado por una instrucción.
- Anti-dependencia, es similar a la dependencia verdadera pero a la inversa, podría suceder que una instrucción futura no pueda empezar a ejecutarse por que necesita de un operando que aun no ha sido escrito o modificado por una instrucción mas antigua.

▼ ¿Qué es el paralelismo entre instrucciones?

Existe paralelismo entre instrucciones cuando las instrucciones de una secuencia son independientes y por lo tanto pueden ejecutarse en paralelo solapándose.

El paralelismo de instrucciones depende de la frecuencia de dependencias de datos verdades y dependencias relativas al procedimiento que haya en el código. Estos factores dependen a su vez de la arquitectura del repertorio de instrucciones y de la aplicación. Otro dependencia del paralelismo de instrucciones es la latencia de una operación, es decir el tiempo que transcurre hasta que el resultado de una instrucción esta disponible para ser usado como operando de una instrucción posterior.

▼ ¿Qué es el paralelismo de la maquina?

Es una medida de la capacidad del procesador para sacar partido al paralelismo en las instrucciones. El paralelismo de la maquina depende del numero de instrucciones que pueden captarse y ejecutarse al mismo tiempo y de la velocidad y las sofisticaciones de los mecanismos que usa el procesador para localizar instrucciones independientes.

Tanto el paralelismo de instrucciones como el de maquina son importantes para mejorar las prestaciones. Un programa podría no tener el suficiente nivel de paralelismo en las instrucciones como para sacarle el máximo partido al paralelismo maquina.

▼ ¿Qué es le emisión de instrucciones?

La emisión de instrucciones se refiere al proceso de iniciar la ejecución de instrucciones en las unidades fundamentales del procesador. Una emisión tiene lugar cuando esta pasa de la etapa de decodificaciones del cause a la primera etapa de ejecución.

La política de emisión de instrucciones esta relacionada al protocolo que se usa al momento de emitir.

▼ ¿Qué políticas de emisión existen?

Existen 3 ordenaciones importantes.

- Emisión en orden y finalización en orden: La política mas sencilla, pero ningún procesador sigue una política tan ingenua
- Emisión en orden y finalización en orden: la finalización desordenada se usa en los procesadores RISC escalares para mejorar la velocidad de las instrucciones que necesitan ciclos. Con la finalización desordenada puede haber cualquier numero de instrucciones en la etapa de ejecución en un

momento dado, hasta alcanzar el máximo grado de paralelismo de la maquina ocupando todas las unidades funcionales. puede darse dependencia de salida. Además es mas difícil ocuparse de una interrupción y las excepciones.

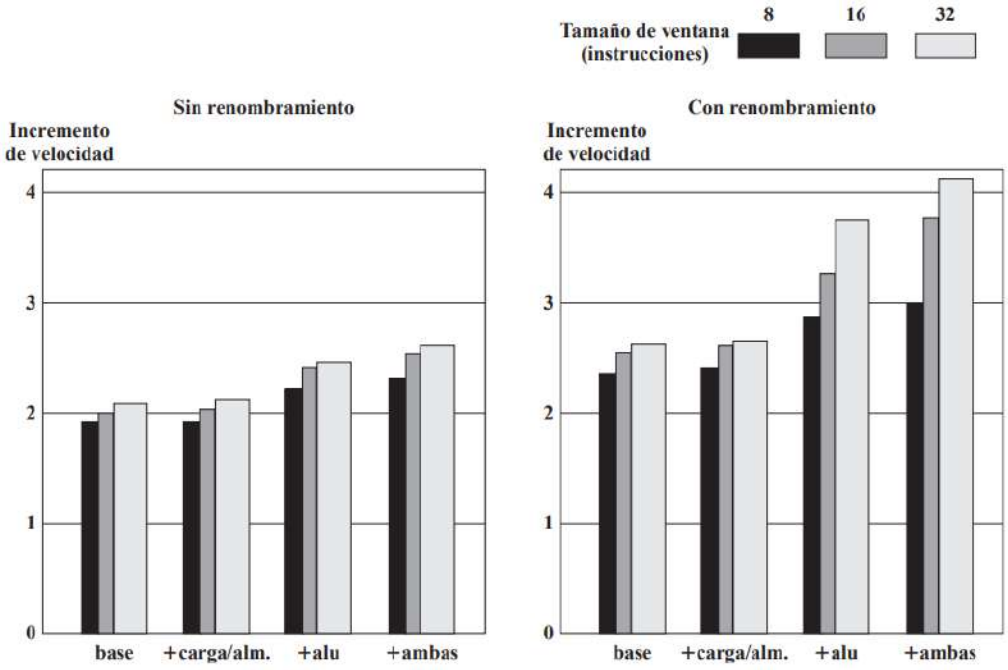
- Emisión desordenada y finalización desordenada: Con la emisión en orden el procesador solo decodificara instrucciones hasta el punto de dependencia y conflicto. No se decodifican mas instrucciones hasta que el conflicto no se resuelve.
  Para la emisión desordenada es necesario desacoplar las etapas del cauce de decodificaciones y ejecución. Esto se hace mediante un buffer llamado ventana de instrucciones. Con esta organización cuando un procesador termina de decodificar una instrucción, la coloca en la ventana de instrucciones. Mientras el buffer no se llene el procesador puede continuar captando y decodificando. Cuando una unidad funcional de la etapa de ejecución esta disponible cualquier instrucción puede emitirse.
  El resultado de esta organización es que el procesador tiene capacidad de anticipación, que le permite identificar instrucciones independientes que pueden introducirse en la etapa de ejecución. Las instrucciones se emiten desde la ventana de instrucciones sin que se tenga muy en cuenta su orden original en el programa. Como antes, la única restricción es que el programa funcione correctamente. Esta política esta sujeta a las mismas restricciones que las anteriores pero se le agrega la antidepndencia. Pero se tiene la ventaja que ahora las instrucciones producen menos atascos en el cause.

▼ ¿Qué es renombramiento de registros?

Con la aparición de nuevos problemas relacionados al desordenamiento de las instrucciones, la dependencia de datos verdaderas y las anti-dependencias. No se puede conocer el valor de un registro cuando es necesario. La técnica para hacer frente a este problema es el renombramiento de registros. Lo que hace es básicamente, el hardware del procesador asigna dinámicamente los registros, cuando se crea un nuevo valor de registro se asigna un nuevo registro para ese valor. Se usan mas registros para poder tener el valor correcto en cada momento.

▼ ¿A que conclusiones se llego cuando se compararon las mejoras?

Se llego a la conclusión de que tener mas hardware solo mejoraba las prestaciones cuando se añadía la estrategia de renombramiento de registros.



El bloqueo por las anti dependencias y dependencias de salidas, causa una merma muy importante al hardware. También es importante recalcar como el tamaño de ventanas empieza a influir cuando combinamos tanto el uso de renombramiento de registros como el la duplicación en los componentes de hardware.

▼ ¿Qué técnica para predecir saltos se usa en las maquinas superescalaes?

Con la llegada de los modelos RISC el salto retardado gano popularidad. Por que podía mantener el cause lleno mientras se predecía la dirección de salto.

Pero con el desarrollo de las maquinas superescalares, la estrategia de salto retardado perdió interés. Por que ejecutar múltiples instrucciones en el ciclo de retardo causaba muchas dependencias entre instrucciones. Por ello las maquinas supersacaleres retornan a la predicción de salto que podía ser dinámica o estática.

▼ ¿Cómo es una ejecución de un programa superescalar?

- Lo primero es el proceso de captación de instrucciones, el cual incluye la predicción de saltos, se usa para dar un flujo dinámico de instrucciones. En

este punto se examinan las dependencias que pueden ocurrir y el procesador elimina las que sean artificiales.

- El procesador envía las instrucciones a una ventana de ejecución, en esta ventana las instrucciones ya no tienen un flujo secuencial sino que están estructuradas de acuerdo a su dependencia de datos verdadera.
- El procesador lleva a cabo la etapa de ejecución de cada instrucción en un orden determinado, determinado por las dependencias y las disponibilidad de los recursos de hardware.
- Las instrucciones se vuelven a poner conceptualmente en el orden secuencial y los resultados se almacenan. a este ultimo paso se lo llama entregar o commit o retirar o retire. Se usa para evitar que instrucciones que fueron ejecutadas especulativamente, como puede ser las predicciones de salto, tengan efecto en el flujo normal del programa y puedan causar errores.

▼ ¿Cuáles son los elementos principales para la implementación superescalar?

- Se necesita una estrategia de captación de instrucciones que capten simultáneamente múltiples instrucciones, que pueda predecir resultados de las instrucciones de salto condicional y captando mas allá de ellas.
- La lógica para determinar dependencias verdaderas entre valores de registros y mecanismo para comunicar los valores a donde sean necesarios.
- Mecanismo mas iniciar o emitir múltiples instrucciones en paralelo.
- Recursos para la ejecución en paralelo de múltiples instrucciones, que incluyan múltiples unidades funcionales segmentadas y jerarquías de memoria capaces de atender múltiples referencias a memoria.
- Mecanismos para entregar el estado del procesador en el orden correcto.

▼ Señale consideraciones destacables del procesador superescalar.

- Influencia de las excepciones, el comportamiento debe ser idéntico al que tendría la misma computadora no segmentada
- Se debe lograr un compromiso entre ejecución desordenada, la liberación de unidades de ejecución, completar las instrucciones en orden y tener excepciones precisas.

- Una posible solución es la emisión desordenada y finalización ordenada.

Unidad 9 - procesamiento paralelo

`Clase 9 filminas, Hojas 667 a 718 del libro Stallings`

▼ ¿Qué es un procesador paralelo?

Cada vez se exigen mayores prestaciones en el mundo de la informática. un procesador paralelo consiste en la ejecución de múltiples programas al mismo tiempo utilizando dos o mas procesadores.

▼ ¿Qué tipos de arquitecturas paralelas existen?

La clásica taxonomía de Flynn, distingue entre cuatro categorías de arquitecturas.

- SISD: una secuencia de instrucciones y una secuencia de datos. Es cuando un único procesador interpreta una única secuencia de instrucciones para operar con los datos almacenados en una única memorias. computadores monoprocesadores.
- SIMD: Una secuencia de instrucciones y múltiples secuencias de datos. Es cuando una única instrucción maquina controla paso a paso la ejecución simultanea y sincronizada de cierto numero de elementos de proceso. Los procesadores vectoriales y los matriciales.
- MISD: múltiples secuencias de instrucción una secuencia de datos. Se transmite una secuencia de datos a un conjunto de procesadores, cada uno de los cuales ejecuta una secuencia de instrucciones diferentes. No se ha implementado esto.
- MIMD: múltiples secuencias de instrucciones y múltiples secuencias de datos. Un conjunto de procesadores ejecuta simultáneamente secuencias de instrucciones de diferentes conjuntos de datos. Los SMP, los clúster y los NUMA.

▼ ¿Qué significa el termino SMP?

Es un tipo de arquitectura de computadoras. Multiprocesamiento simetrico.

Un SMP puede definirse como un computador con las siguientes características:

- Hay dos o mas procesadores 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 pata todos los procesadores.
- Todos los procesadores comparten los dispositivos de E/S, bien a través de los mismos canales o mediante canales distintos que proporcionan caminos de acceso al mismo dispositivo.
- Todos los procesadores pueden desempeñar las mismas funciones.
- El sistema esta controlado por un sistema operativo integrado que proporciona interacción entre los procesadores y sus programas a los niveles de trabajo , tarea ficheros y datos.

▼ ¿Qué ventajas tiene respecto a una arquitectura monoprocesador?

El SMP planifica la distribución de procesos o hilos entre todos sus procesadores. Ventajas:

- Prestaciones: Se puede realizar trabajo en paralelo, entonces proporciona mejores prestaciones.
- Disponibilidad: Un fallo del procesador no hará que todo el equipo se detenga, por que todos los procesadores cuentan con las mismas características.
- Crecimiento incremental: se pueden aumentar las prestaciones del sistema añadiendo mas procesadores.
- Escalado: los fabricantes pueden ofrecer mas gamas de productos con precios y prestaciones diferentes en función al numero de procesadores.

▼ ¿Cómo es la organización del SMP?

- Existen dos o mas procesadores.
- Cada procesador es autónomo, incluye una unidad de control, una ALU registros y muy probablemente una cache.
- Cada procesador tiene acceso a una memoria principal compartida y a los dispositivos de E/S a través de alguna forma de mecanismo de interconexión.
- Los procesadores pueden comunicarse entre sí a través de la memoria.

- Los procesadores también pueden intercambiar señales directamente.
- La memoria debe ser organiza de manera que sea posible los accesos simultáneos a bloques separados.
- En algunas configuraciones cada procesador puede tener también su propia memoria principal privativa y sus canales de E/S además de los recursos compartidos.

▼ ¿Qué es el bus de tiempo compartido?

El bus de tiempo compartido es el mecanismo mas simple para construir un sistema multiprocesador. La estructura y las interfaces son básicamente las mismas que las de un sistema de un único procesador.

El bus consta de líneas de control, dirección y datos

Para facilitar las transferencias de DMA con el procesador de E/S, se proporcionan elementos:

- Direccionamiento.
- Arbitraje.
- Tiempo compartido.

▼ ¿Cuáles son las características atractivas del bus compartido?

- Simplicidad, es la forma mas facil de organizar el multiprocesador. Es muy similar al de un solo procesador.
- Flexibilidad, es generalmente sencillo expandir el sistema conectando mas procesadores al bus.
- Fiabilidad, el bus es esencialmente un medio pasivo y el fallo de cualquiera de los dispositivos no provoca un fallo en todo el sistema.

▼ ¿Cuáles son las principales desventajas de este bus?

- La principal desventaja de este bus son sus prestaciones, todas las referencias a memorias pasan por el bus. En consecuencia la velocidad del sistema esta limitada por el tiempo de ciclo.

        - Tener una cache por procesador reduce ampliamente la cantidad de consultas a memoria.
            - Pero se pueden producir problemas en las coherencias de las caches, por lo tanto hay que considerar esto para el diseño.

▼ ¿Qué significa UMA, NUMA Y CC-NUMA?

- UMA significa, uniform memory acces o acceso uniforme de memoria, lo cual significa que el acceso a cualquier lugar de la memoria tiene el mismo tiempo y que todos los procesadores tienen el mismo tiempo de acceso
- NUMA significa, non uniform memory acces o acceso no uniforme de memoria, lo cual dice que dependiendo de donde se encuentre el dato, el tiempo de acceso podrá variar. Diferentes procesadores tienen diferentes tiempos de accesos a la memoria.
- CC-NUMA: Es como el numa pero mantiene la coherencia de cache entre las caches de distintos procesadores.

▼ ¿Cómo funcionan las arquitecturas de memoria compartida?

En este tipo de arquitecturas cada nodo es un procesador con su propia memoria principal y sus propios mecanismos de E/S. Los nodos trabajan de manera independiente pero están conectados mediante una red de comunicaciones que les permite intercambiar información mediante paso de mensajes.

Desde el punto de vista del programador la memoria es compartida y cualquier procesador puede acceder a cualquier posición de memoria mediante una instrucción de load o store.

Se dice que es de tipo NUMA, por que los accesos a memorias no son siempre iguales, depende de donde este ubicado el dato. Un acceso a memoria puede ser Local o remoto, dependiendo de la localia.

▼ ¿Cómo funcionan las arquitecturas de memoria distribuida?

En estas arquitecturas cada nodo es un procesador con su propia memoria principal y sus propios mecanismos de E/S. Los nodos trabajan independientemente pero están conectados a una red que les permite intercambiar información, es una arquitectura débilmente acoplada.

En este caso la memoria se encuentra distribuida completamente, la memoria principal se encuentra divida física y lógicamente en espacios de direcciones físicas independientes. Cada procesador puede acceder únicamente a su espacio de direcciones local, la comunicación entre procesos debe ser explicita mediante el paso de mensajes

▼ ¿Qué ventajas y desventajas supone esta arquitectura?

Las ventajas son es altamente escalable y es mucho mas fácil de comprender y depurar. Además se elimina la necesidad de resolver el problema de las coherencias de cache y los problemas de sincronización y de consistencia; ya que no se comparte ni la memoria ni su espacio de direcciones.

Las desventajas que hay que tener en cuenta son que aparecen mayor complejidad en la programación y que además introduce una latencia de comunicaciones que puede llegar a ser importante.

▼ ¿Cómo funciona una arquitectura on-chip?

Existen dos tipos de arquitectura on-chip

- on-chip de memoria compartida, se integran varios núcleos de procesador sobre un mismo chip. Depende del fabricante se comparte distintos niveles de memoria. Existen distintas alternativas como tener un núcleo complejo, con varios simples o solo núcleos simples o solo nucleos complejos y que estos también convivan con procesadores específicos
- on-chip de memoria distribuida, tener la memoria compartida limita al procesador en ciertos aspectos. Por eso también existen los on-chip de memoria distribuida, en donde cada núcleo tiene su propia memoria tanto cache como principal.

▼ ¿Qué es un procesador multihebra o multithread?

Se intenta aprovechar el paralelismo entre instrucciones a nivel hilo.

Las secuencias de instrucciones se dividen en secuencias más pequeñas llamadas hebras que pueden ejecutarse en paralelo. Ni la complejidad del circuito ni el consumo de potencia aumentan.

El multitreading consiste en ejecutar al mismo tiempo dos o mas threads de un programa, permitiendo que cada uno de estos threads sea planificado de la manera mas conveniente para el procesador, es decir aprovechando al máximo los recurso disponibles. Es equivalente a tener dos o mas procesadores lógicos o virtuales en lugar de uno sólo.

▼ ¿Qué es una hebra o thread?

Es una unidad de trabajo de un proceso que puede asignarse, incluye un contexto de procesador(sp y pc) y area de datos para su pila. Se ejecuta secuencialmente y es ininterrumpible

▼ ¿Qué es el thread switch?

Es cuando se produce un cambio de control del procesador entre hebras de un mismo proceso

▼ ¿Qué diferencias hay entre el procesamiento multihebra implícito y explicito?

El explicito ejecuta concurrentemente instrucciones de diferentes hebras explicitas, mescla las instrucciones de diferentes hebras en cauces compartidos ó por ejecución paralela en causes paralelos.

Mientras que el implícito ejecuta concurrentemente varias hebras extraídas de un único programa secuencial

▼ ¿Qué cuestiones debe contemplar un procesador multihebra explicito?

- Debe proporcionar un contador de programa distinto para cada una de las hebras que pueden ejecutarse concurrentemente.
- Se trata cada hebra de manera separada, prediciendo saltos, renombrando registros, entre otros para mejorar la ejecución.
- Se aprovecha el paralelismo entre hebras que sumado al paralelismo entre instrucciones puede proporcionar una mejora en las prestaciones considerable.

▼ ¿Cuál es la diferencia entre un SMP y un cluster?

En los clústers, la unidad de interacción física es normalmente un mensaje o un fichero completo. En un SMP, la interacción se puede producir a través de elementos de datos individuales, y puede existir un elevado nivel de cooperación entre procesadores.

Anexo: buses

▼ ¿Qué es un bus?

Un bus es un camino de comunicación entre dos o mas dispositivos. Una característica clave de un bus es que se trata de un medio de transmisión compartido. Al bus se conectan varios dispositivos, y cualquier señal transmitida por uno de esos dispositivos está disponible para que los otros dispositivos conectados al bus puedan acceder a ella. Si dos dispositivos transmiten durante el mismo periodo de tiempo, sus señales pueden solaparse y distorsionarse. Consiguientemente, solo un dispositivo puede transmitir con éxito en un momento dado.

Usualmente, un bus está constituido por varios caminos de comunicación, o líneas. Cada línea es capaz de transmitir señales binarias representadas por 1 y por 0.

▼ ¿Cómo son las estructuras de los buses?

El bus de sistema está constituido, usualmente, por entre cincuenta y cien líneas. A cada línea se le asigna un significado o una función particular. Aunque existen diseños de buses muy diversos, en todos ellos las líneas se pueden clasificar en tres grupos funcionales, líneas de datos, líneas de direcciones y de control., Además pueden existir líneas de alimentación para suministrar energía a los módulos conectados al bus.

▼ ¿Cómo funciona las líneas de datos?

Las líneas de datos proporcionan un camino para transmitir datos entre los módulos del sistema. El conjunto constituido por estas líneas se denomina bus de datos. El bus de datos puede incluir entre 32 y cientos de líneas, cuyo número se conoce como anchura del bus de datos. Puesto que cada línea solo puede transportar un bit cada vez, el número de líneas determina cuántos bits se pueden transferir al mismo tiempo. La anchura del bus es un factor clave a la hora de determinar las prestaciones del conjunto del sistema.

▼ ¿Cómo funciona las líneas direcciones?

Las líneas de dirección se utilizan para designar la fuente o el destino del dato situado en el bus de datos. Por ejemplo, si el procesador desea leer una palabra (8, 16 o 32 bits) de datos de la memoria, sitúa la dirección de la palabra deseada en las líneas de direcciones. Claramente, la anchura del bus de

direcciones determina la máxima capacidad de memoria posible en el sistema. Además, las líneas de direcciones generalmente se utilizan también para direccionar los puertos de E/S.

▼ ¿Cómo funciona las líneas de control?

Las líneas de control se utilizan para controlar el acceso y el uso de las líneas de datos y de direcciones. Puesto que las líneas de datos y de direcciones son compartidas por todos los componentes, debe existir una forma de controlar su uso. Las señales de control transmiten tanto órdenes como información de temporización entre los módulos del sistema. Las señales de temporización indican la validez de los datos y las direcciones.

▼ ¿Qué es la jerarquía de buses?

La jerarquía de bus, es necesario por las dificultades que pueden surgir.

- En general, a más dispositivos conectados al bus, mayor es el retardo de propagación. Este retardo determina el tiempo que necesitan los dispositivos para coordinarse en el uso del bus. Si el control del bus pasa frecuentemente de un dispositivo a otro, los retardos de propagación pueden afectar sensiblemente a las prestaciones.
- El bus puede convertirse en un cuello de botella a medida que las peticiones de transferencia acumuladas se aproximan a la capacidad del bus. Este problema se puede resolver en alguna medida incrementando la velocidad a la que el bus puede transferir los datos y utilizando buses más anchos. Sin embargo, puesto que la velocidad de transferencia que necesitan los dispositivos conectados al bus está incrementándose rápidamente, es un hecho que el bus único está destinado a dejar de utilizarse.

La mayoría de computadores utilizan varios buses, normalmente organizado jerárquicamente. En una estructura típica hay un bus local que conecta el procesador a una memoria cache y al que pueden conectarse también uno o mas dispositivos locales. El controlador de la memoria cache conecta la cache no solo al bus local sino también al bus del sistema, donde se conectan todos los módulos de memoria principal.

Aunque es posible conectar controladores de E/S directamente al bus del sistema. Una solución mas eficiente consiste en utilizar uno o mas buses de expansión. Este

regula las transferencias entre datos entre el bus del sistema y los controladores conectados.

▼ ¿Qué tipos de buses existen?

- Dedicados: Permite asignar una función o a un subconjunto físico de componentes del computador. Un ejemplo es cuando se usan líneas de datos o de direcciones.
- Multiplexados: En el multiplexado se usan menos líneas , y por eso se ahorra espacio y normalmente costes. La desventaja es que se necesita una circuitería mas compleja para cada modulo, además existe una reducción de las prestaciones debido a que los eventos que deben compartir la mismas líneas no pueden producirse en paralelo.

▼ ¿Qué método de arbitraje existen?

Se necesita arbitraje para evitar que la información se mezcle en los buses.

- Centralizada: en donde existe un único dispositivo de hardware. denominado controlador de bus o arbitro, el cual es responsable de asignar tiempos en el bus. Puede ser un dispositivo separado o ser parte del procesador.
- Distribuido: no tiene controlador central, en su lugar cada modulo dispone de la lógica para controlar el acceso y los módulos actúan conjuntamente para compartir el bus.

En ambos métodos el propósito es designar a un dispositivo como maestro de bus. Entonces el maestro podría iniciar una transferencia de datos con otro dispositivo que actúa como esclavo en ese intercambio concreto.

▼ ¿Cómo se coordinan los buses?

Se coordinan usando la temporización. que puede ser sincrónica o asíncrona.

- Cuando es sincrónica, la presencia de un evento en el bus esta determinada por un reloj. El bus incluye una línea de reloj a través de la que se transmite una secuencia en la que se alteran intervalos regulares de igual duración a uno y a cero.
- Con el temporización asíncrona, la presencia de un evento en el bus es consecuencia y depende de que se produzca un evento previo.

La temporización síncrona es más fácil de implementar y comprobar. Sin embargo, es menos flexible que la temporización asíncrona. Debido a que todos los dispositivos en un bus síncrono deben utilizar la misma frecuencia de reloj, el sistema no puede aprovechar las mejoras en las prestaciones de los dispositivos. Con la temporización asíncrona, pueden compartir el bus una mezcla de dispositivos lentos y rápidos, utilizando tanto las tecnologías más antiguas, como las más recientes.

▼ ¿Cómo es la anchura del bus?

El concepto de anchura del bus se ha presentado ya. La anchura del bus de datos afecta a las prestaciones del sistema: cuanto más ancho es el bus de datos, mayor es el número de bits que se transmiten a la vez. La anchura del bus de direcciones afecta a la capacidad del sistema: cuanto más ancho es el bus de direcciones, mayor es el rango de posiciones a las que se puede hacer referencia.

▼ ¿Qué tipos de transferencias existen?

- Lectura
- Escritura
- lectura-modificación-escritura
- lectura-después de- escritura.

▼ ¿Qué es el bus PCI?

El bus PCI es un bus muy popular de ancho de banda elevado, independiente del procesador, que puede ser utilizado como bus de periférico o bus para una arquitectura entreplanta. Da mejores prestaciones en E/S de alta velocidad. El estándar permite el uso de hasta 64 líneas de datos a 66 MHz. no es la velocidad lo que lo hace atractivo sino que se puede implementar con muy pocos circuitos integrados y permite que otros buses se conecten al bus PCI.

Utiliza temporización síncrona y un esquema de arbitraje centralizado.

▼ ¿Qué es un bus SCSI?

La interfaz SCSI (Small computer system interface) en si es un tipo de bus utilizado para conectar controladores de disco y otros periféricos.

Unidad final - Resolución de exámenes finales

Examen 25/03/2010

▼ ¿Qué es un bus? Describa tipos, arbitraje y técnicas de sincronización. Mencione diferencias entre bus PCI y bus SCSI.

Un bus es un camino de comunicación entre dos o mas dispositivos, una característica clave del bus es que se trata de un medio de transmisión compartido. Se tiene que controlar el bus de alguna manera para que los dispositivos no distorsionen los mensajes. El bus esta constituido.

Existen dos tipos, dedicados que solo se usan para una función o dispositivo y los multiplexados que se usan para muchos propósitos, obviamente se ahorra en costos pero es un bus mas lento.

En cuanto al arbitraje existen dos métodos, ambos sirven para que el bus se pueda usar de manera correcta y que cada uno tenga su tiempo.
El primero es el centralizado que tiene una pieza de hardware especial llamada arbitro que controla que dispositivo usara el bus en determinado momento.
Mientras que con el método distribuido son los dispositivos conectados al bus quienes se sincronizan para decidir quien tomara el control del bus.
En ambos casos cuando un dispositivo gana el control se convierte en maestro y el dispositivo con el que se comunica es el esclavo.

Para la sincronización existen dos técnicas, la síncrona y la asíncrona. En la síncrona se tiene un reloj el cual marca cuando se producen los intercambios el cambio de 0 a 1 marca una nueva transferencia. En la asincrónica, el tiempo de intercambio esta marcado por el momento en el cual ocurren los eventos. La sincrónica limita que se puedan aprovechar las mejoras en rendimiento de los dispositivos, aunque es mas fácil de implementar y usar.

El PCI es un Bus especialmente atractivo por que se puede conectar otros buses sobre el, tiene una unidad central que se encarga del arbitraje y es síncrono.

El SCSI es una interfaz que se utiliza para permitir la conexión de distintos tipos de periféricos a un ordenador

▼ ¿Qué es una interrupción?¿Cual es la función del PIC?

Una interrupción es un mecanismo que tienen las computadoras para poder cambiar la ejecución normal de un programa. Puede ser de hardware o de software, tener distintos niveles de prioridades y normalmente son atendidas por el

procesador antes de iniciar un nuevo ciclo de captación, cuando esto sucede se guarda el estado actual del procesador y de la siguiente instrucción a ejecutar y se pasa el control a un gestor de interrupciones. Pueden ocurrir interrupciones cuando se esta ejecutando la subrutina de una interrupción, se debe tomar una estrategia para poder solucionar esto. La mas simple y menos inteligente, es desactivar las interrupciones mientras se esta atendiendo una y mantenerlas pendientes, esto no esta bueno por que pueden ocurrir interrupciones criticas que tengan que ser atendidas inmediatamente. La segunda alternativa es tener distintos niveles de prioridades para las interrupciones y que una interrupción con mayor nivel de prioridad pueda interrumpir a una con menor nivel de prioridad.

Para entender el funcionamiento del PIC, primero tenemos que entender que existen múltiples dispositivos que quieren producir interrupciones en determinados momentos, por lo tanto hay que plantear estrategias para poder captar y gestionar estos dispositivos.

El PIC es un dispositivo que tienen diferentes líneas de conexión en donde se pueden conectar dispositivos que quieran producir interrupciones. Tiene múltiples registros que le permiten gestionar la forma en la cual resuelve las distintas interrupciones.

▼ Estructura de módulo de E/S. Describa técnicas que usa la CPU para realizar operaciones de E/S.

Un modulo de entrada salida es un dispositivo que actúa como interface entre los distintos dispositivos externos y el procesador. Su existencia se justifica en que existen muchos dispositivos de E/S y muchos se manejan de maneras diferentes, el modulo e/s oculta toda lo lógica de manejo al procesador.

Existen tres técnicas de E/S.

- La primera es la e/s programada, en la cual el procesador se comunica con el modulo de es y le pide realizar un intercambio, el modulo de es, se comunica con el dispositivo afectado y durante el tiempo que a este le toma responder a la petición, el procesador le consulta constantemente si su petición a terminado. Esta técnica es bastante pobre por que significa una perdida de velocidad de la CPU.
- La segunda es la e/s mediante interrupciones, es una mejora a la técnica anterior, en la cual el procesador inicia el intercambio con la petición al modulo

de es y continua con su trabajo normal, entonces cuando los datos están listos para ser intercambiados, el dispositivo produce una interrupción; antes de iniciar la próxima captación el procesador atiende esta interrupción y resuelve la es.

- La tercera es la e/s con DMA, esta técnica plantea la incorporación de un nuevo dispositivo de hardware, el dmac, el cual puede comunicarse directamente con la memoria, el procesador solo inicia la transferencia y luego espera a que el dmac la resuelva, este posee algunas instrucciones y un procesador para poder encargarse de esto.

▼ Describa algoritmos de reemplazo de bloque y técnicas de escritura en caché

La cache es mas pequeña que la memoria principal, por lo tanto hay que establecer como serán remplazados los bloques cuando se encuentre llena y necesitamos subir un bloque. Primero hay que plantear que existen tres métodos para asignar un bloque de memoria principal a un marco de cache, asignación directa, asignación asociativa por conjuntos, asignación asociativa, la primera es la que nos interesa explicar puesto que a cada bloque le corresponde un marco que se calcula haciendo numero de bloque mod cantidad de marcos. Volviendo a los métodos de remplazo en el directo no nos queda otra alternativa que intercambiar el marco que le corresponde al bloque que queremos alocar en cache, esto puede causar conflictos si se remplazan constantemente dos bloques que tienen el mismo marco. Después tenemos otras alternativas como el LRU, o menos recientemente usado, que quitara de cache el marco que no se haya referenciado por mayor tiempo. También contamos con el algoritmo LFU, o menos frecuentemente usado, que desaloja el bloque que menos referencias haya tenido. Luego existe el algoritmos FIFO, o primero en entrar primero en salir, que remplaza el bloque que este mas tiempo en cache. Finalmente contamos con un algoritmo que quita marcos de manera aleatorio; a pesar de parecer la mas simple que las anteriores demostró que los resultados eran a penas peores que los otros.

Con respecto a las técnicas de escrituras de cache primero tenemos que entender que la memoria cache tiene que seguir el principio de coherencia, es decir que todos la información que esta tenga tiene que ser igual a la de la memoria cache.

Para respetar esto se plantean dos estrategias. La primera de escritura inmediata que plantea que cuando se modifica un valor que se encuentra en cache inmediatamente se refleja ese cambio en memoria principal, esto tiene como

desventaja que hace un uso intensivo del bus. La segunda es post-escritura que escribe los valores en memoria luego que el marco es remplazado, esta estrategia puede causar problemas si tenemos varios procesadores que comparten la memoria principal, pero que no comparten cache por que podría leer información que no es correcta.

▼ ¿Qué es la segmentación de cauce? Describa tipos de dependencias que afectan el funcionamiento de cauces. ¿Cuánto mejora el rendimiento?

La segmentación de cause es una estrategia que mejora el uso de los recursos de una computadora, consiste en dividir a las instrucciones en diferentes etapas e ir resolviendo estas etapas de manera simultanea.

Una instrucción se puede dividir en **captación y decodificación, búsqueda de operandos, ejecución de la instrucción, acceso a la memoria y almacenamiento.**

Podrían ocurrir dependencias de distintos tipos que pausaria la ejecución del cause; la dependencia de datos que se da cuando una instrucción depende de datos que no están disponibles en ese momento, pueden ser de tres tipos war, waw o raw. La dependencia estructural que señala que dos instrucciones quieren usar la misma pieza de hardware en el mismo momento y la dependencia de control, que se produce cuando no se sabe con que instrucción se continuara la ejecución, esta dada por los saltos condicionales o incondicionales.

Todas las dependencias tienen técnicas que permiten disminuir el impacto causado.

Suponiendo que nuestro cause esta formado por 5 etapas, y no existen dependencias por lo tanto el procesador no se detiene, cuando mas nos acerquemos a infinito mas cerca estaremos del CPI ideal que es 1. Por que en las primeras 5 etapas no vamos a finalizar ninguna instrucción pero luego, finalizaremos una en cada ciclo

---

Examen 11/04/2015

▼ ¿Qué es una pila? Describir el comportamiento con anidamiento de múltiples procedimientos/funciones utilizando pila.

Una pila es una estructura de datos especial que permite almacena información, cuenta con dos operaciones básicas; el push que sirve para almacenar un

elemento al tope de la pila, el pop que sirve para sacar el ultimo elemento que ingreso. Se necesita un dato especial para implementar la pila que guarde la posición del ultimo elemento que ingreso, esta se aumenta o decrementa según la cantidad de push o pop que ocurran.

En cuanto a los procedimientos se debe encolar la información de la subrutina, posición actual de la instrucción, variables del procedimiento.

▼ Justifique el uso dos niveles de caché.

Tener una cache disminuye la frecuencia con la que el procesador hace referencias a memoria principal, por lo tanto mejora la velocidad del sistema en general. Con las mejoras que se lograron en el diseño del procesador y su disminución de tamaños, se pudo meter la cache dentro del procesador y se creo la posibilidad de tener una segunda cache que fuera externa. Con esta segunda cache externa que generalmente es un poco mas grande que la primera se disminuye la tasa de fallos aun mas pero se complejiza la lógica de remplazo, asignación y escritura. Incluso con el avance de la tecnología se empezó a incorporar un tercer nivel de cache.

▼ ¿Cómo funciona un módulo de E/S? Describa las características fundamentales de un DMA

Un modulo de E/S es un dispositivo de hardware encargado de comunicarse con los diferentes tipos de dispositivos externos, justifica su necesidad por que el procesador necesita comunicarse con estos dispositivos, pero sus estándares de transferencias de datos, velocidades y ordenes son diversas; por lo tanto el procesador debería implementar toda esa lógica de control. El Modulo de entrada y salida multiplexa dispositivos de es para abstraer al procesador y que este solo vea un unido dispositivo.

El DMA es una forma de transferencia de datos, que requiere la incorporación al sistema de una pieza de hardware especial conocida como DMAC, este supone una mejora al procesador ya que este solo transfiere la petición al inicio y luego espera que el DMAC disponga los datos en memoria principal.

▼ ¿Qué es la segmentación de cauce? Explique los atascos producidos por saltos

La segmentación de cauce es una estrategia que consiste en dividir las instrucciones en múltiples unidades. Las que se trabajo en la materia son.

- Captación

- Decodificación
- Ejecución de la instrucción
- Acceso a memoria
- Almacenamiento del resultado

Teniendo las componentes de hardware suficientes se puede ejecutar una etapa de cada instrucción de manera simultanea con otras etapas, teniendo todas las etapas del cause ocupadas. Se puede hacer una analogía con una fabrica en donde se divide las tareas de producción en diferentes etapas que se trabajan de manera individual y luego ensamblan este trabajo.

Se pueden producir distintos tipos de atascos. Los de datos, los de control y los estructurales.

Los de control, que están relacionados con los saltos, son los que se forman ante la imposibilidad de saber como el orden de ejecución de un conjunto de instrucciones. Cuando se produce un salto, se debe determinar en que posición de memoria se debe continuar la ejecución, durante ese tiempo el procesador debe pausar el cauce para evitar que se capte una instrucción incorrecta y se produzcan errores. Existen distintas alternativas para evitar la perdida de tiempo.

▼ Funcionamiento de un clúster.

Un clúster esta formado por múltiples nodos, cada uno tiene su propia memoria principal, sus buses, canales de es y un procesador. Entre ellos se comunican, mediante una red de interconexión, para poder hacer diferentes tareas. A diferencia de los SMP, los clúster cuentan con una memoria principal por cada procesador o nodo. Esto hace que sean mas veloces pero complejiza la lógica de implementación.

Examen ??/??/????

▼ Describa el mecanismo de interrupción. Explique características y tratamiento de interrupciones
múltiples

La interrupción es un cambio en la ejecución normal de un programa, puede ocurrir por diferentes motivos entre los cuales se puede encontrar, un fallo por una instrucción que no existe o el resultado de la instrucción es indebido, o puede que

simplemente sea un reloj del procesador que le permite realizar tareas cada determinado tiempo, también pueden ocurrir por algún error relacionado al hardware, como puede ser un error de memoria o falta de energía entre otros.

Pueden ocurrir múltiples interrupciones al mismo tiempo. Se vieron dos posibles soluciones cuando esto ocurría, se podían desactivar las interrupciones mientras se atendía una, pero apareció el problema de que si una interrupción critica llegaba y no era atendida podía causar un fallo en el sistema. Por esto es que la segunda alternativa es tener prioridades para cada interrupción y que una interrupción de prioridad mayor pueda interrumpir a una de menor prioridad.

▼ ¿Qué es segmentación de cauce?¿Qué ventajas proporciona su implementación?

La segmentación de cauce divide las instrucciones en diferentes etapas y lo que hace es ejecutar todas las etapas de manera paralela. Proporciona una aumento del cpi considerable. Puesto que cuando tiende a infinito la cantidad de instrucciones el cpi tiende a 1, el ideal.

▼ Describa las características funcionales del Acceso Directo a Memoria (DMA).

Es una de las mejores formas de intercambio de información entre el procesador y un dispositivo externo, puesto que el procesador solo se involucra al inicio de la transferencia y luego recibe una interrupción cuando esta lista la información en memoria.

Necesita una pieza de hardware extra conocida como DMAC.

▼ Memoria Caché. Describa el mapeo asociativo por conjuntos. Analice las políticas de escritura
desde el punto de vista de la coherencia de datos.

La cache es una memoria rápida ubicada entre el procesador y la memoria principal. Esta esta formada por marcos de n cantidad de palabras y aloca bloques de n cantidad de palabras de la memoria principal.

Para saber en que marco debe alocar determinado bloque utiliza diferentes algoritmos. El método de asociativo por conjuntos. Divide los marcos de la cache en grupos iguales, y a cada bloque le corresponde determinado grupo de marcos, por lo tanto puede ubicarse en cualquiera de los de su conjunto. Para poder calcular que marco va a ocupar se hace el calculo de numero de bloque en RAM modulo de cantidad de grupos = grupo en cache.

Hay que tener en cuenta el factor de la coherencia de dataos al realizar una escritura de cache por que se debe seguir el principio de coherencia en donde todos los cambios realizados en cache deben propagarse a los niveles mas bajos de la jerarquía con el fin de evitar que una lectura de la misma información contenga un valor erróneo, podría suceder en organizaciones que tienen múltiples procesadores y comparten memoria principal pero no cache. Tenemos varios métodos para poder mantener la coherencia. El primero es la vigilancia del bus con escritura inmediata en donde cada controlador de cache monitoriza las líneas de direcciones para detectar cambios en memoria por parte de otros maestros de bus. Si otro maestro escribe en una posición de memoria compartida que también reside en la cache, el controlador invalida el elemento de la cache. Se necesita el uso de escritura inmediata para la cache. Otro metodo es la transparencia de hardware, en donde se utiliza otro hardware especial para asegurarse de que todas las actualizaciones de memoria principal, via cache, se queden reflejadas en toda la cache. Memoria excluida de cache, solo una parte de la memoria principal esta compartida entre los procesadores y esta marcada para que no se pueda subir a cache.

▼ Describa tres características qué usted considere las más importantes de las arquitecturas RISC.

Instrucciones cortas, fáciles de utilizar y de tamaño fijo: Las instrucciones están pensadas para ser fáciles de utilizar y de comprender, a demás que las instrucciones no deben implicar muchos efectos colaterales. y deben ser de tamaño fijo para poder aprovechar las optimizaciones en el momento de la captación.

Gran cantidad de registros: se encontró que el uso de subrutinas era un aspecto muy importante y que poder traer todas las variables necesarias de manera rápida mejoraba la velocidad a la que trabajaba el procesador.

Produce programas mas largos, a diferencia del CISC, los programas son mas largos pero no por eso mas lentos.

Modos de direccionamiento sencillos.

▼ ¿Qué son los procesadores superescalares?

Los procesadores superescales son aquellos que pueden atender mas de una instruccion al mismo tiempo, lo logran implementando mas de un dispositivo de hardware lo cual le permite atender las etapas del cauce de manera simultanea.