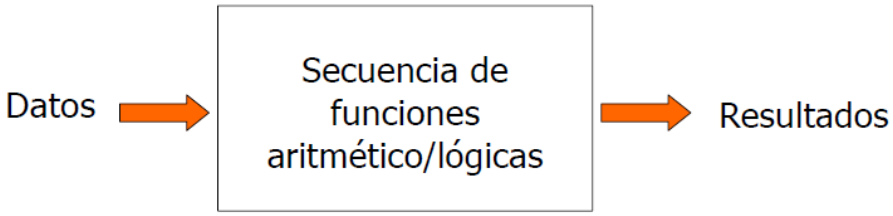
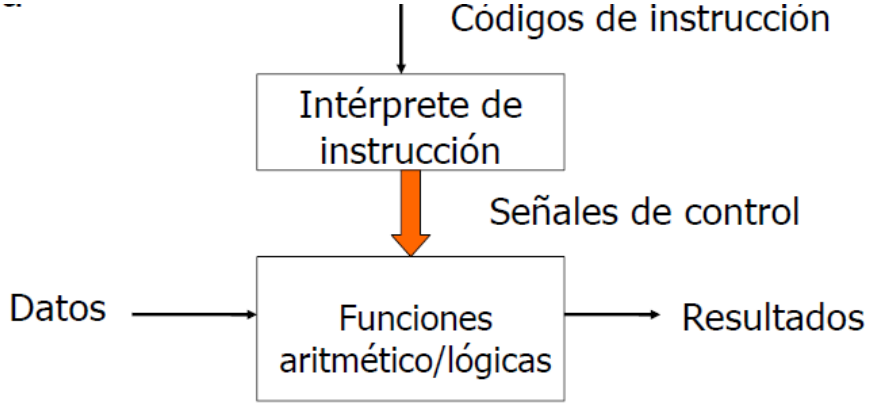
CLASE 1- Repaso

CONCEPTO DE PROGRAMA



Programación en hardware, al cambiar las tareas, cambiaba el hardware.



Programación en software: en cada caso, se efectuará alguna operación sobre datos.

Para cada paso se necesita un nuevo conjunto de señales de control. Las instrucciones proporcionan esas señales. Surge un nuevo concepto de programación

ARQUITECTURA VON NEUMANN

La CPU está formada por la Unidad de Control y la ALU. Los datos e instrucciones deben introducirse en el sistema. Los resultados se proporcionarán entre componentes de E/S. La memoria principal es necesaria para almacenar temporalmente datos e instrucciones.

PC: controlador del programa IR: registro de instrucciones

MAR: registro dirección de memoria MBR: registro buffer de memoria

E/S AR: registro dirección de E/S E/S BR: registro buffer de E/S

REPERTORIO DE INSTRUCCIONES

Conjunto completo de instrucciones que se realizan en una CPU. Representado por un conjunto de código de ensamblaje.

ELEMENTOS DE UNA INSTRUCCION

Código de operación, referencia a operandos fuentes, referencia a resultado, referencia a la siguiente instrucción

ALMACENAMIENTO DE OPERANDOS

En memoria principal, memoria virtual o memoria cache, registros del CPU o en dispositivo E/S

TIPO DE INSTRUCCIONES

- Procesamiento de datos: instrucciones aritméticas, lógicas
- Almacenamiento de datos: instrucciones de memoria
- Transferencia de datos: instrucciones de E/S
- Control: instrucciones de testeo y flujo

CANTIDAD DE DIRECCIONES

- Mas direcciones por instrucciones: instrucciones más complejas, mas registros y menos instrucciones en el programa
- Menos direcciones por instrucciones: instrucciones menos complejas, mas instrucciones en el programa, ejecución más rápida

DESICIONES EN EL DISENIO DEL CONJUNTO DE INSTRUCCIONES

- Tipo de operando
- Repertorio de operaciones: cuantos, cuales, complejidad
- Modos de direccionamiento
- Formato: longitud, numero de direcciones, tamaño de campos
- Registros: número de registros diferenciables

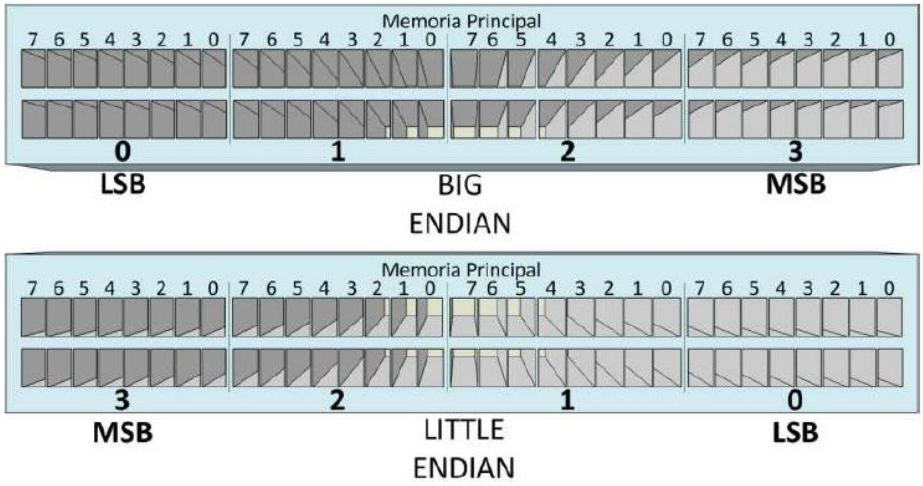
RISC esta contrapuesto a CISC, ambos son dos tipos de arquitecturas: \

RISC: computadora de conjunto reducido de instrucciones

CISC: computadora de conjunto complejo de instrucciones

TIPOS DE OPERANDOS

Direcciones, números (pto. Flotante, entero), caracteres (ASCII,EBDIC,etc), datos lógicos (bits como FLAGS).



BIG ENDIAN: el byte más significativo en la dirección con valor número más bajo

LITTLE ENDIAN: el byte menos significativo en la dirección con valor numérico más bajo

TIPOS DE OPERACIONES

- Transferencia de datos: debo especificar ubicación del operando fuente y del de destino, el tamaño del dato y el modo de direccionamiento
- Aritméticas: operaciones básicas de numero entero, punto flotante, con signo o sin signo y otras operaciones como inc, dec, neg, absolute, shift left/right
- Lógica: operaciones que manipulan bits (booleanas, rotate left/right) u operaciones para cambiar formados como de ASCII a EBCDIC
- Conversión: IN/OUT- MOV. Se puede hacer mediante un controlador a parte
- E/S: modifican el valor contenido en PC como un salto condicional/ incondicional o salto con retorno o llamada a subrutina
- Control del sistema y del flujo

MODOS DE DIRECCIONAMIENTO

Inmediato, directo de memoria/ absoluto, directo de registro, indirecto de memoria (en desuso), indirecto por registro, indirecto por desplazamiento, indirecto basado, indexado o relativo al PC, pila.

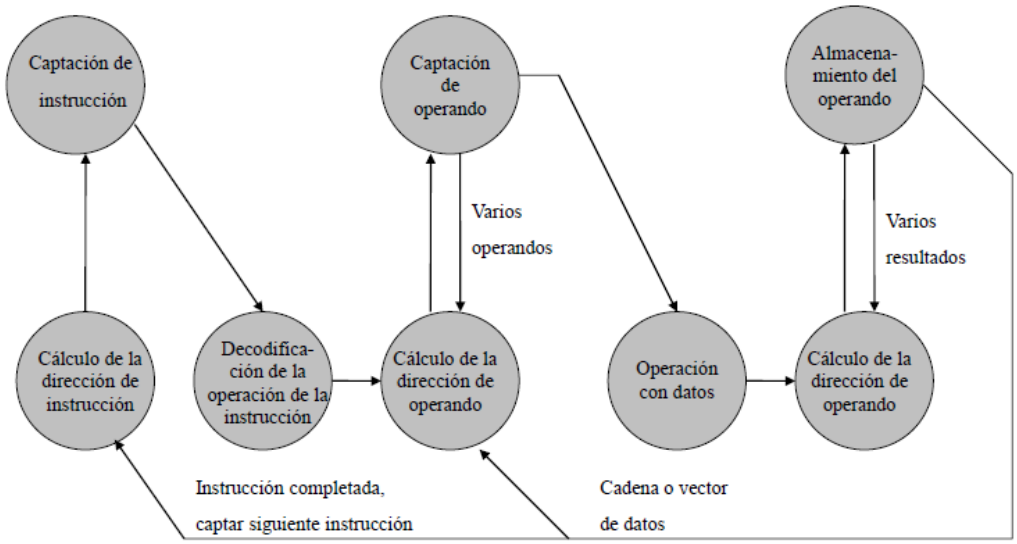
RISC usa modo de direccionamiento inmediato e indirecto por desplazamiento

CICLO DE INSTRUCCIÓN BASICA

Hay 2 ciclos:

- Ciclo de captación: la dirección de instrucción que se debe captar se encuentra en PC. La UC capta la instrucción desde memoria y va al IR. PC incrementa, UC incrementa instrucción y la lleva a cabo
- Ciclo de ejecución: acciones posibles
    - Procesador- memoria: transferencia de datos de CPU a memoria
    - Procesador E/S: transfiere datos entre CPU y modulo E/S
    - Procesamiento de datos: operación aritmética

- Control: alteración de secuencia
- O puede ocurrir una combinación de todos



1 Diagrama de estados

SUBRUTINAS

Innovación en lenguajes de programación. Pueden invocarse desde cualquier punto del programa. Brinda economía y modularidad. Requiere pasaje de parámetros

Pasaje de parámetros:

- Vía registros: el número de registros es la principal limitación
- Vía memoria: usa área definida de memoria, es difícil de estandarizar
- Vía pila: método más utilizado. Independiente a memoria y registros

Funcionamiento de la pila: el operando se encuentra en la cabeza de la pila. Requiere un registro puntero de pila que contiene la dirección de la cabeza de la pila. Las operaciones permitidas son *push* y *pop*

Operación de apilar y desapilar: secuencia de 2 acciones: movimiento de datos. Registro-memoria o memoria- registro. Y modificación del puntero antes/ después de la anterior.

POSIBLES PASOS EN UN PROCEDIMIENTO

1. Salvar SP
2. Reservar espacio para datos locales
3. Salvar el valor de otros registros
4. Acceder a parámetros
5. Escribir sentencias a ejecutar
6. Retornar parámetros
7. Regresar al procedimiento

CLASE 2- Interrupciones

Mecanismo mediante el cual se puede interrumpir el procesamiento normal del CPU que tiene una ejecución secuencial de instrucciones. La interrupción puede ser de origen interno o externo al CPU

MOTIVOS DE INTERRUPCION

- Resultado de una ejecución de una instrucción
- Temporizador interno del procesador
- Operaciones de E/S
- Fallo del hardware

¿QUE HACER SI SE INTERRUMPEN?

En casi todos los casos, implica transferir el control a otro programa que salve el estado del procesador, corrija/responda el motivo que ocasionó la interrupción, restaure el estado original del procesador y retorne la ejecución del programa interrumpido.

JERARQUIA DE INSTRUCCIONES

Si hay múltiples que pueden solicitar interrupción, se establece cuáles son más importantes. Se consideran:

No enmascarables: no pueden ignorarse. Eventos peligrosos o de alta prioridad

Enmascarables: pueden ser ignorados. Con instrucciones podemos inhibir la solicitud

INTERRUPCIONES POR HARDWARE

Generados por dispositivos de E/S. Son “verdaderas” interrupciones. El sistema de cómputo tiene que manejar eventos externos no planeados. No relacionados con el procesamiento en ejecución en ese momento.

EXCEPCIONES

Interrupciones por hardware creadas por el procesador en respuesta a ciertos eventos

- Condiciones excepciones: valores en ALU de punto flotante
- Fallo del programa: tratar de ejecutar instrucciones no definidas
- Fallo del hardware: error de paridad de memoria
- Accesos no lineados o a zonas de memoria protegidos

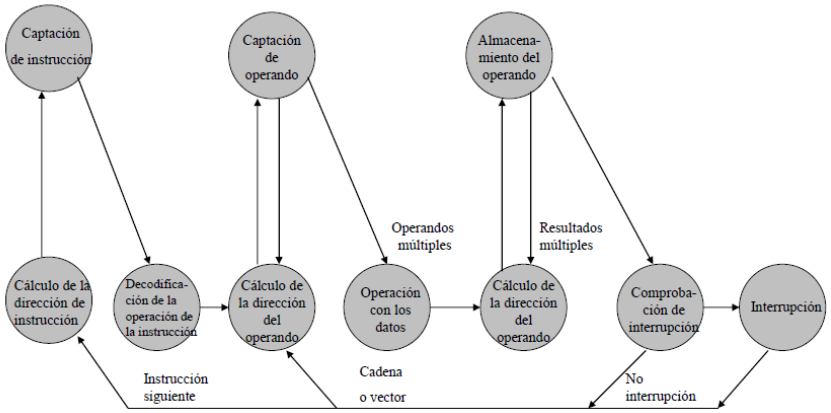
INTERRUPCIONES POR SOFTWARE

Muchos procesadores tienen instrucciones explicitas que afectan al procesador de la misma manera que las interrupciones por hardware. Generalmente usados para hacer llamados a funciones del sistema operativo.

Hay sistemas que no permiten hacer una llamada directa a una dirección de la función del sistema operativo por estar en zonas reservadas. Si no tuviera interrupciones por software,

debería escribir todas las funciones que necesito o al cargar un programa había que mirar función de BIOS y SO y reemplazar el código de dirección de función.

Ciclo de instrucción: captación, ejecución, gestión de interrupciones.



CICLO DE INTERRUPCION

Se comprueba si se ha solicitado alguna interrupción. Si no hay señal, se capta siguiente instrucción. Si hay señal de interrupción, se suspende la ejecución en curso, guarda el contexto (próxima instrucción y el estado del procesador), carga el PC con dirección del comienzo de la rutina de gestión y finalizada, el procesador retorna la ejecución del programa

INTERRUPCIONES MULTIBLES

- Interrupciones inhabilitadas: el procesador puede y debe ignorar la señal repetición de una interrupción si se produce una interrupción en ese momento. Si se hubiera generado una interrupción, se mantiene pendiente y se examinara luego una vez que se hayan habilitado nuevamente. Ocurre una interrupción, se inhabilita, se gestiona la misma y luego se habilitan otra vez. Por lo tanto, las interrupciones se manejan en un orden secuencial estricto.
- Definir prioridades: una interrupción de alta prioridad puede interrumpir a un gestor de interrupciones de prioridad mensos. Cuando se ha gestionado la interrupción de alta prioridad, el procesador vuelve a las interrupciones previas. Terminada las rutinas de gestión, retorna al programa del usuario

RECONOCIMIENTO DE INSTRUCCIONES

- Instrucciones multinivel: cada dispositivo que puede provocar una interrupción, tiene una entrada física de interrupción conectadas al CPU, es sencillo pero caro
- Línea de interrupción única: una sola entrada física de pedido de interrupción a la que están conectados los dispositivos, se debe preguntar a cada dispositivo si ha producido un pedido de interrupción, conocido como polling
- Interrupciones vectorizadas: el dispositivo que quiere interrumpir además de la señal de pedido de interrupción, debe colocar en el bus de datos un identificador. Lo coloca el periférico directamente o el controlador de interrupciones (usado en MSX88)

ESCENARIO DE TRABAJO
Si el procesador tiene una única entrada de pedido de interrupciones y varios productores de interrupciones se soluciona con el PIC “dispositivo controlador programable de interrupciones”

CLASE 3- Problemas de E/S

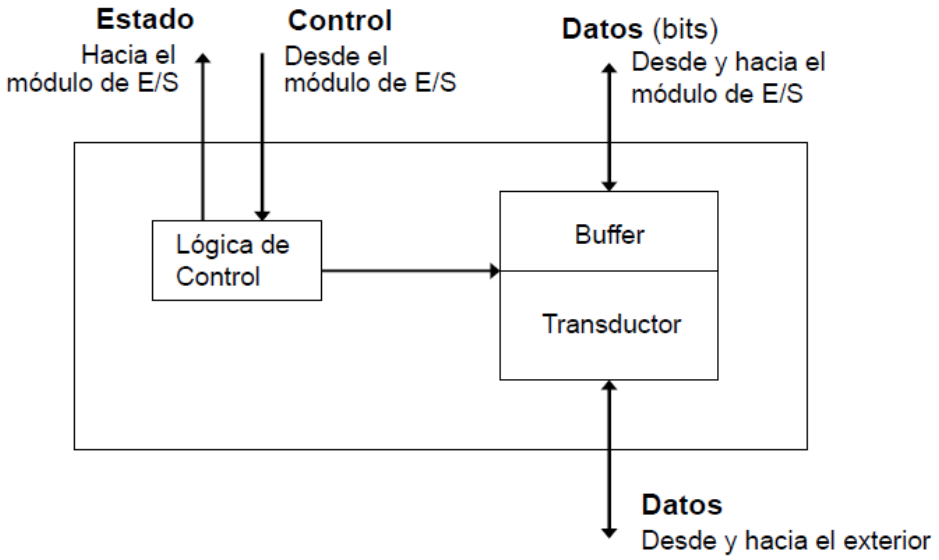
Existe una gran variedad de periféricos con varios métodos de operaciones. Como transmisión de diferentes cantidades de datos en distintas velocidades y usan distintos tamaños y formados de datos (son más lentos que la CPU y RAM, hay una necesidad de módulos de E/S) Los módulos de entrada/salida (E/S) tienen las siguientes funciones básicas:

- Conectar con la CPU y memoria vía bus del sistema.
- Conectar con los periféricos mediante conexiones de datos particularizadas.

Un módulo de E/S realiza una interfaz entre el procesador, la memoria y los periféricos. Pueden manejar 1 o más periféricos

DISPOSITIVOS EXTERNOS

- E/S básicos: monitor, mouse, teclado
- Almacenamiento: disco duro, CD, DVD
- Impresoras
- Comunicación entre dispositivos remotos: modem
- Multimedia: micrófono, parlantes
- Automatización y control: sensores, alarmas, adquisición de datos



Lógica de control: para saber qué hacer con las

señales, manejo de direccionamiento

Buffer: manejo de datos de 1,8,16 bits

Transductor: para convertir datos

Señal de estate: ready/ not ready

Señal de control: para la función a realizar

FUNCIONES DE UN MODULO DE E/S

- Controla y temporiza uno o más dispositivos externos
- Interpreta las ordenes que recibe del CPU y transmite al periférico
- Comunicación con CPU y memoria
- Controla transferencia de datos entre CPU y periférico
- Comunicación con dispositivo
- Informa a CPU entrada del periférico
- Almacenamiento temporal de datos
- Detención de errores

CAPACIDADES DE UN MODULO DE E/S

- Ocultar propiedades del dispositivo a la CPU: temporizador, formatos, electro mecanismos
- Ocuparse de 1 o más dispositivos
- Controlar o no las funciones del dispositivo

OPERACIONES DE E/S

Requieren direccionamiento: E/S mapeada en memoria o aislada, transferencia de información, gestión de la transferencia

DIRECCIONAMIENTO DE E/S

E/S asignada en memoria: lo dispositivos E/S y memoria comparten un único espacio de direcciones, E/S se parece a la memoria de L/E. No hay orden específico para E/S

E/S aislado: espacios de dirección separados, necesidad de líneas especiales de E/S y de memoria

TECNICAS DE GESTION DE E/S

E/S programada con espera de resultados: es un intercambio de datos entre el CPU y el módulo E/S. La CPU tiene control sobre la operación de E/S (comprobación del estado del dispositivo, envió de comandos de L/E, transferencia de datos), la CPU espera que el modulo E/S termine la operación, entonces la CPU permanece ociosa durante un periodo.

La CPU solicita la operación de E/S al módulo, el modulo E/S realiza la operación. El modulo E/S activa los bits de estado del dispositivo direccionado y espera. La CPU

comprueba periódicamente el estado de esos bits, hasta que la operación fue realizada. En caso contrario, la CPU espera y vuelve a comprobarlo más tarde.

La CPU emite una dirección, da una orden y *control* indica al módulo qué hacer, *test* comprueba el estado del módulo y periféricos, *L/E* transfiere datos desde a hacia el dispositivo por el bus de datos.

E/S con interrupciones: la CPU no tiene que esperar la finalización de la tarea de E/S, puede seguir procesando. No se repite la comprobación de los estados de los módulos. El modulo envía un periodo de interrupción a la CPU cuando esta lista nuevamente

La CPU envía una orden de lectura, chequea si hay pedidos de interrupciones pendientes al final de cada ciclo de instrucción. La CPU detecta el pedido, guarda el contexto, interrumpe el proceso y realiza la gestión de la instrucción. La CPU solicita datos. Es más óptimo, solo espera a que lo llamen cuando sea necesario

E/S con acceso directo a memoria (DMA): dispositivo que controla la transferencia de datos entre el periférico y la memoria sin la interrupción de CPU.

Debe actuar como maestro del bus en la transferencia y debe ser capaz de solicitar el uso del bus con esenciales y lógica de arbitraje, especificar la dirección de memoria sobre la que se realiza la transferencia, generar señales de control del bus.

Etapas:

- Inicialización dela transferencia: la CPU debe enviar al interfaz del periférico y al DMAC los parámetros de la transferencia. Inicialización del interfaz y del controlador DMA. Después de la inicialización, la CPU retorna a sus tareas y ya no se ocupa de la evolución de la tarea
- Realización de la transferencia: cuando el periférico esté listo para realizar la transferencia, lo indica al DMAC. Pide el control del bus y realiza a transferencia entre el periférico y la memoria. BUS MASTER: DMAC + periférico. BUS SLAVE: memoria. después de la transferencia de cada palabra, se actualizan los registros del DMAC: número de bytes, dirección de memoria, etc

- Finalización de la transferencia: el DMAC libera el bus y devuelve el control al CPU. DMAC suele activar una señal de interrupción para avisar al CPU la finalización de la operación E/S
  Puede haber problemas como degradar el rendimiento del CPU si el DMAC hace uso intensivo del bus. El problema se reduce con el uso de memoria cache ya que el CPU leería instrucciones de la cache y no necesitaría el bus.

  Hay diferentes tipos de transferencia, si el DMAC solo toma el control del bus durante los intervalos de tiempo en las que el CPU no hace uso del mismo, el rendimiento del sistema no sufrirá degradación. Los tipos de transferencia son *por ráfaga* (burst) o *por robo de ciclo* (cycle-stealing)

  1. DMA modo ráfaga: el DMA 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. Su ventaja es la transferencia rápida, pero durante el tiempo que

dura la transferencia, la CPU no puede usar el bus y degradara el rendimiento del sistema

2. DMA modo robo de ciclo: DMAC solicita control al CPU. Cuando lo concede, se hace la transferencia de una única palabra y el DMAC libera el bus. DMAC solicita las veces que sea necesaria hasta finalizar la transferencia del bloque de datos. Su ventaja es que no hay degradamiento, pero tarda más la transferencia.
   Para la CPU no es una interrupción. Si bien el trabajo es lento, no será tanto como si ella realizara la transferencia

CUESTIONES DE DISENIO

Identificación del módulo que interrumpe:

Diferentes líneas para cada módulo: limita número de dispositivos

Consulta a software: ocurrido un pedido de interrupción, la CPU consulta a cada módulo para quien fue el demandante. Es lento

Conexión en cadena: la línea de reconocimiento de interrupción se conecta encadenando los módulos, la línea de pedido es compartida. Una vez enviada la confirmación por parte de la CPU, el modulo responderá colocando un vector, en el bus que lo identifica. La CPU emplea el vector como un puntero para acceder a la rutina de servicio

Interrupciones múltiples: todas las líneas tienen un orden de prioridad. Las líneas con más prioridad pueden interrumpir a las de menor prioridad. Si hay un maestro de bus, solo él puede interrumpir

CANALES DE E/S

Son módulos controladores de periféricos

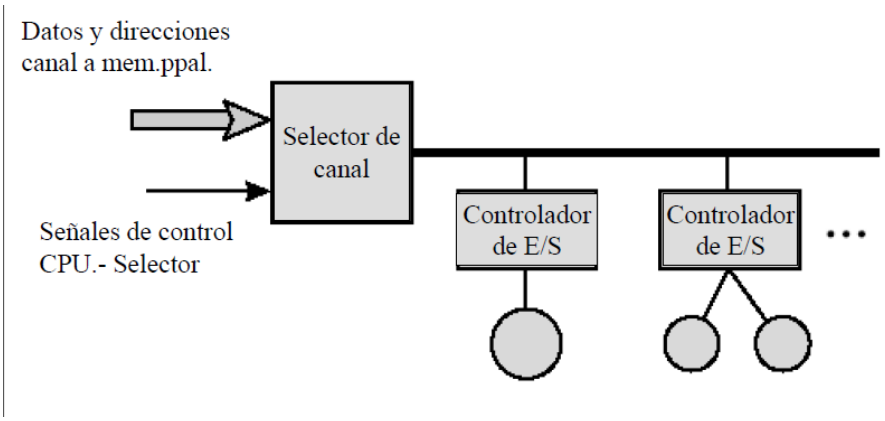
1. CPU controla periféricos
2. Se agrega el modulo E/S o controlador
3. Modulo E/S más los llamados de interrupción
4. El modulo E/S prevé DMA
5. Modulo E/S tiene su propio procesador con su pequeño conjunto de instrucciones
6. El modulo además tiene memoria local, o sea se convierte en una computadora en sí mismo

Características canales E/S: representan una extensión del concepto de DMA, completo control de la transferencia de datos, entonces la CPU no ejecuta instrucciones de E/S

Tipos de canales E/S: selector o multiplexor

Selector: controla varios dispositivos de alta velocidad y uno por vez, entonces el canal se dedica para la transferencia de datos de ese dispositivo. El canal selecciona un dispositivo y

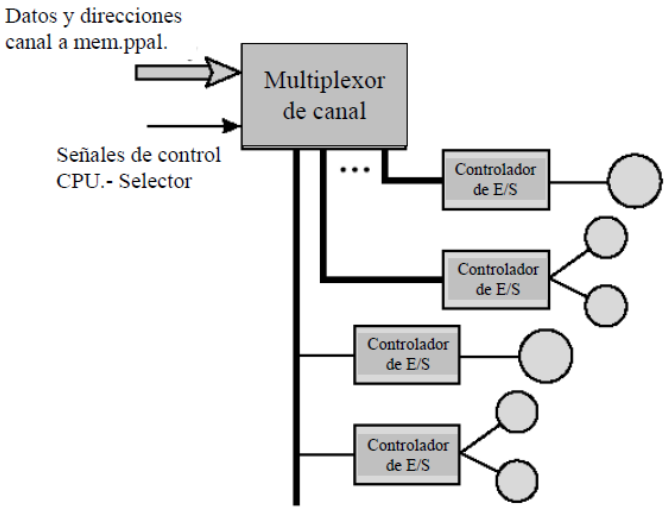
efectúa la trasferencia. Los dispositivos son manejados por un controlador o modulo E/S entonces el canal E/S ocupa el lugar de la CPU en el control de esos controladores



Multiplexor: puede manejar E/S con varios dispositivos a la vez

Multiplexor de bytes: acepta y admite caracteres

Multiplexor de bloques: intercambia bloques de datos



CLASE 4- Segmentación de cauce

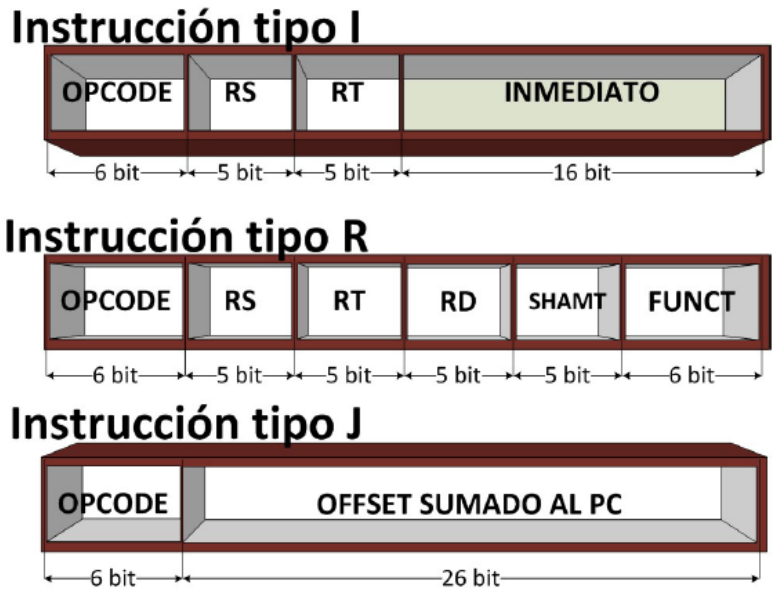
Es una técnica de hardware que busca optimizar el software. La segmentación de cause es una forma efectiva de organizar el hardware de la CPU para realizar más de una operación al mismo tiempo. Consiste en descomponer el proceso de ejecución de las instrucciones en fases/etapa que permiten una ejecución simultánea. Cada etapa usa un recurso distinto, lo cual consigue aprovechar todos los recursos. Explota el paralelismo (forma de computación en la cual varios cálculos pueden realizarse simultáneamente, entre las instrucciones de un flujo secuencial) entre las instrucciones de un flujo secuencial

CARACTERISTICAS

- La segmentación (acción de separar o dividir una cosa en partes) es una técnica de mejora de prestaciones a nivel de diseño de hardware
- La segmentación es invisible al programador
- Necesidad de uniformizar las etapas
- El diseño de procesadores segmentados tiene una gran dependencia del repertorio de instrucciones

TAREAS A REALIZAR POR CICLO

- **Búsqueda** (F, fetch= ir a buscar): se accede a memoria por la instrucción, incrementa PC. Uso *memoria de instrucción*.
- **Decodificación** (D, decode): se decodifica la instrucción, obteniendo la operación a realizar en la ruta de datos, accedo al banco de registros por los operandos (en caso de ser necesario). Se calcula el valor del operando inmediato. Uso *banco de registros*
- **Ejecución** (X, execute): se ejecuta la operación en la ALU. Uso *ALU*
- **Acceso a memoria** (M, memory Access): si se quiere un acceso a memoria, se accede. Usa *memoria de datos*
- **Almacenamiento** (W, writeback): si se quiere volcar un resultado a un registro, se accede. Uso el *banco de registros*.



PRESTACIONES DEL CAUCE SEGMENTADO

teórico: el máximo rendimiento es completar una instrucción con cada ciclo de reloj. Si k es el número de etapas del cauce entonces la velocidad del procesador segmentado = velocidad secuencial x K.

El incremento potencial de la segmentación es proporcional al número de etapas del cauce. Incrementa la productividad (throughtput) pero no reduce el tiempo de ejecución de cada instrucción. O sea, no incremento velocidad de instrucciones si no la corrida del programa

ANALISIS DE LA SEGMENTACION

Suposiciones: todas las tareas duran igual tiempo. Las instrucciones siempre pasan por todas las etapas y todas las etapas pueden ser manejadas en paralelo.

Problemas: no todas las instrucciones necesitan todas las etapas. No todas las etapas pueden ser manejadas en paralelo.

ATASCOS DE CAUCE (STALL)

Cause: cadena de procesos conectados de forma tal que la salida de cada elemento de la cadena es entrada del próximo

Son situaciones que impiden que la siguiente instrucción que se ejecute en el ciclo que le corresponde. Hay distintos motivos de atasco:

Estructurales: provocados por conflictos por los recursos. 2 o más recursos necesitan usar el mismo recurso de hardware en el mismo ciclo

Por dependencia de datos: ocurren cuando 2 instrucciones se comunican por medio de un dato. Condición en que los operando fuente o destino no están disponibles en el momento en que se necesitan en una etapa determinada. Varios tipos

- RAW (Lectura después de escribir): trata de leer un valor antes de que este se escribió
- WAW (Escritura después de escritura): sobrescribe, sucede solo si se adelantan operaciones
- WAR (Escritura después de lectura): instrucción modifica valor antes de que otra anterior que lo tiene que leer lo lea.

Por dependencia de control: ocurren cuando la ejecución de una instrucción depende de cómo se ejecuta otra.

CLASE 5- Posibles soluciones a atascos

Los atascos de un cause son situaciones que impiden a la siguiente instrucción que se ejecute en el ciclo que le corresponde. Están los atascos estructurales (conflictos de recursos de hardware), dependencia de datos (2 instrucciones comunicadas por un dato), dependencia de control (la ejecución de una instrucción depende de cómo se ejecute otra, como un salto condicional)

Penalización: tiempo perdido. Software= compilador

SOLUCIONES A RIESGOS ESTRUCTURALES

Simple: replicar, segmentar o realizar turnos para el acceso a las unidades funcionales en conflicto. Implica duplicar recursos de hardware como sumadores o restadores a parte de la ALU, separación en memorias de instrucción y datos o turnar el acceso al bando de registros.

SOLUCIONES A RIESGOS DE DATOS

Para los riesgos RAW se debe determinar cómo y cuándo aparecen. Será necesario una unidad de detención de riesgos y/o compilador más complejo. Hay dos soluciones por hardware *fowarding* o software *instrucción NOP* o *reordenación de código.*

- Técnica por hardware fowarding o adelantamiento: consiste en pasar directamente el resultado obtenido en una instrucción a las instrucciones que lo necesitan como operando. El dato está disponible en la ALU, se lleva a la entrada de la etapa correspondiente sin esperar a la etapa de escritura. O sea, que la instrucción que lo necesita, no debe esperar a que se escriba en el registro de destino.
  Es fácil de implementar si se identifican todos los adelantamientos y se comunican a los registros de segmentación correspondientes. Debo implementar un nuevo hardware. No las elimina al 100% los atascos RAW si no que los minimiza
- Técnica por software *instrucción NOP* o *reordenación de código:* es responsabilidad del compilador, y en realidad no es u n a técnica q u e 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 (para ello a veces es necesario el renombrado de registros), 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

SOLUCIONES A RIESGOS DE CONTROL

Existe una penalización por salto. Una instrucción de salto o bifurcación (división de algo en dos cosas) condicional puede ser incondicional o condicional. Cuando es incondicional, la dirección de destino se debe determinar lo más pronto posible, dentro del cauce, para reducir la penalización. En los saltos condicionales, introduce riesgo adicional por la dependencia entre la condición de salto y el resultado de una instrucción previa. Hay distintas técnicas como flujos múltiples, buffer de bucles, predicción de salto, salto retardado o pre captar el destino de salto

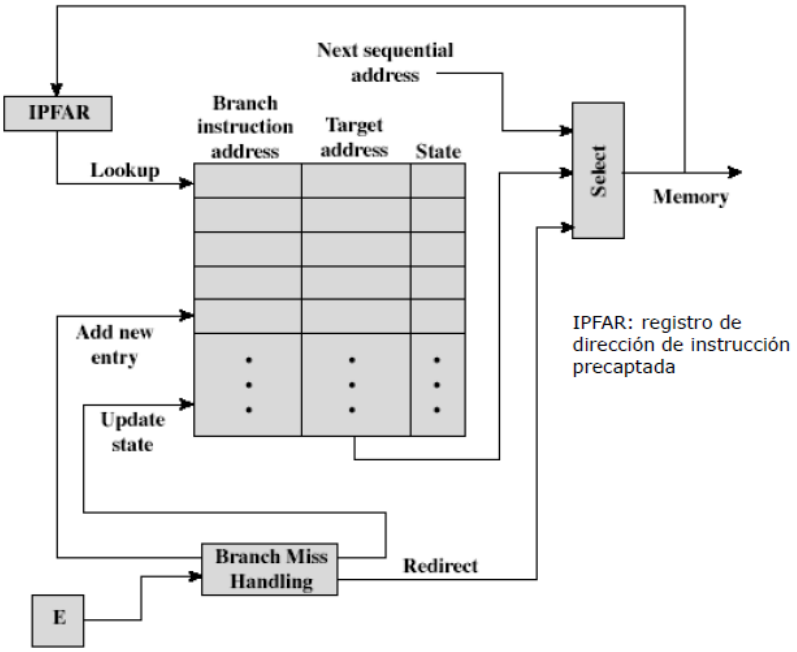
Una modificación sencilla sería adelantar la resolución de los saltos a la etapa, en ella se decodificará y se sabe que es un salto. Se puede evaluar la condición de salto, calcular la dirección de salto. Basado en predicciones

Hay 2 formas de tratamiento: técnica de hardware *predicción de saltos* para evitar parada o técnica de software llamada *salto retardado*.

- Predicción de saltos: puede ser estática o dinámica
  - Técnica estática: no dependen de la historia de ejecución, puede predecir que nunca salta (asume que el salto no se producirá y siempre capta la siguiente instrucción y la penalización solo sucede en caso de que el salto se tome) o predecir que siempre salta (asume que el salto se producirá y siempre capta la instrucción destino del salgo). Si el salto no se toma y la predicción se acertó,

se continua normalmente la ejecución. La predicción falla cuando se toma el falso. O sea, siempre predice que el salto no se toma.

- Técnica dinámica: técnica que dependen de la historia de ejecución. Puede ser un *conmutador saltar/no saltar* o *una tabla de historia de saltos* (Branch target buffer BTB).
  Conmutador saltar/no saltar: basado en la historia de las instrucciones, eficaz para los *bucles*. Por ejemplo, a cada instrucción de bifurcación pueden asociarse uno o más bits que reflejen su historia reciente. Estos bits son referenciados como un conmutador saltar/no saltar, que dirige al procesador a tomar una determinada decisión la próxima vez que se encuentre la instrucción. Estos bits de historia se guardan temporalmente en un almacenamiento de alta velocidad. Va analizando lo que pasa en el programa
  Branch target buffer: memoria cache asociada a la etapa de búsqueda. Va guardando todos los saltos tomados sobrescribiéndose, guarda la dirección de instrucción de salto, dirección de salto y un indicador de estado= veces tomado el salto, hace cuánto tiempo. En un salto, accedo a la información del BTB y según la información, tomo el salto o no, por ejemplo: hace cuánto tiempo se tomó el salto, es una predicción, no salta directamente.
  Si no encuentro el salto en el BTB, actualizo la información.



- **Hardware, salto retardado:** la idea es realizar trabajo útil mientras el salto se resuelve. Un hueco o ranura de retardo de salto (delay slot) es el periodo de penalización luego de una instrucción de salto. El compilador reordena las instrucciones de un programa, tratando de situar instrucciones útiles, o sea que no dependen del salto, en los huecos del retardo. Si no es posible, utiliza la instrucción NOP

Las instrucciones en los huecos de retardo de salto se captan siempre. Requiere reordenar las instrucciones.

Otras soluciones hardware puede ser:

Predecir según código de operación: hay instrucciones con más probabilidades de saltar, su acierto puede llegar a un 75%

Flujos múltiples: un cauce siempre sufre penalización por las instrucciones por las instrucciones de bifurcación, porque debe escoger una de las dos instrucciones a captar a continuación y puede hacer la elección equivocada. Una solución burda es duplicar las partes iniciales del cauce y dejar que este capte las dos instrucciones utilizando dos caminos, pero tiene dos problemas, con cauces múltiples hay retarde debido a la competencia por el acceso a los registros y memoria y además pueden entrar en cauce instrucciones de bifurcación adicionales antes de que se resuelva la decisión de la bifurcación original

Pre captación el destino del salto: cuando se identifica una instrucción de bifurcación condicional, se pre capta la instrucción destino del salto, además de la siguiente a la bifurcación. Se guarda entonces esta instrucción hasta que se ejecute la instrucción de bifurcación. Si se produce el salto, el destino ya hará sido pre captado

Buffer de bucles: es una memoria pequeña de gran velocidad, gestionada por la etapa de captación de instrucción del cauce, que contiene, las n instrucciones captadas más recientes. 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.

Clase 6- RISC

RISC: computadoras de repertorio reducido de instrucciones. Presta segmentación eficiente porque son instrucciones simples. Tiene un repertorio limitado, o sea instrucciones simples. CISC también implementa cauce, pero no es tan eficiente

CARACTERISTICAS PRINCIPALES

- Gran número de registros de uso general o mejor tecnología de compiladores para optimizar el uso de registros
- Repertorio de instrucciones limitado y sencillo
- Énfasis en la optimización de la segmentación de instrucciones, de las características más importantes
- Tamaño fijo de instrucciones
- No más de 2 modos de direccionamiento
- En RISC, va a haber mayor cantidad de instrucciones para un código de HLL al tener un set reducido de instrucciones más básico.

FINALIDAD DE CISC

- Facilitar el trabajo del escritor del compilador
- Mejorar la eficiencia de ejecución: secuencias complejas de operaciones micro código
- Dar soporte a high level lenguajes HLL

INCONVENIENTES DE CISC

- El software es más caro que el hardware
- Nivel de lenguaje cada vez más complicado
- Salto semántico: diferencia entre operaciones HLL y operaciones de arquitectura

Todo esto conduce a un gran repertorio de instrucciones, mas modos de direccionamiento, varias sentencias de HLL implementadas en el hardware

CARACTERISTICAS DE EJECUCIÓN

Estudios sobre programas escritos en HLL, operaciones realizadas, operando usados, secuenciamiento de la ejecución. Estudios dinámicos: medir durante la ejecución

OPERACIONES

Asignaciones: movimiento de datos

Estamentos condicionales: control secuencial

El procedimiento llamado/retorno consume mucho tiempo. Algunas instrucciones HLL conducen a muchas operaciones de código máquina.

OPERANDOS

Principalmente variables escalares (es valor, variable o campo que sólo puede tener un valor en un cierto momento) locales. La optimización debe concretarse en el acceso a las variables locales. Refleja la importancia de una arquitectura que preste a un rápido acceso a operandos, dado que es una operación que se realiza con mucha frecuencia

LLAMADAS A PROCEDIMIENTOS

Consume mucho tiempo, depende del número de parámetros tratados y del nivel de anidamiento. La mayoría de los programas no tienen una larga secuencia de llamadas seguida por la correspondiente secuencia de retornos. La mayoría de las variables son locales, las referencias a operandos están muy localizadas

CONSECUENCIAS

Se puede ofrecer mejor soporte para los HLL optimizando las prestaciones de las características más usadas y que mas tiempo consumen. Usar un gran número de registros (para optimizar las referencias a operandos), prestar cuidadosa atención al diseño de los cauces de instrucciones debido a la alta proporción de instrucciones de bifurcación condicionales y de llamada a procedimientos, es recomendable un repertorio con instrucciones simples (para convertir de HLL a LLL)

Las instrucciones más usadas son las más simples

AMPLIO BANCO DE REGISTROS

Se concluye que es deseable un rápido acceso a los operandos.

Maneras de trabajar con los registros mediante

Aproximación por software: confía al compilador la maximización del uso de los registros. El compilador tratara de asignar registros a las variables que se usen más en un periodo de tiempo dato. el compilador es necesario para asignar registros. Asigna registros a las variables que se usen más en un tiempo dato. Requiere el uso de sofisticados algoritmos de análisis de programas

Aproximación por hardware: utilización de más registros. De esta manera, mas variables pueden mantenerse en registros durante más tiempo

REGISTROS PARA VARIABLES LOCALES

Muchos registros para reducir el acceso a memoria. Almacenar las variables escalares locales en registros. Un problema es en cada llamada de procedimiento o función, cambia la localidad y los parámetros deben ser pasaros, resultados deben ser devueltos, las variables de los programas de llamada tienen que ser restauradas

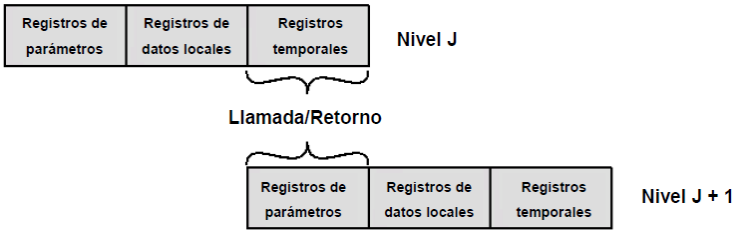
VENTANAS DE REGISTRO

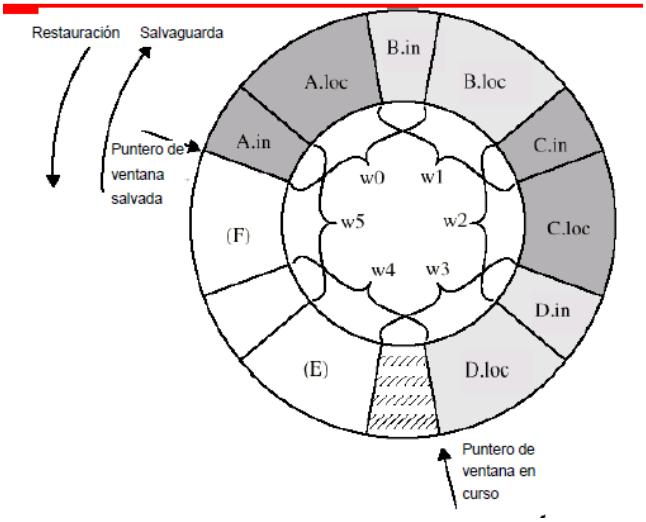
El uso conjunto amplio de registros debería reducir la necesidad de acceder a memoria. La labor del diseño es organizar los registros de tal modo que se alcance esa meta.

La solución evidente es almacenar estos registros, reservando tal vez varios registros para variables locales. El problema es que la definición de local varía en cada llamada y retorno de procedimiento. En cada llamada, las variables locales deben guardarse desde los registros en la memoria, para que los registros puedan ser reutilizados por el programa llamada. Además, deben pasarse parámetros. En el retorno, las variables del programa padre tienen que ser restablecidas.

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 registro distinta de tamaño fijo, en lugar de salvaguardar los registros en memoria. Las ventanas de procedimientos están parcialmente solapadas para permitir el paso de parámetros.

Hay 3 áreas dentro de un conjunto de registros: registros de parámetros, registros de datos locales y registros temporales. Los registros temporales de un conjunto se solaparán con los registros de paramentos del nivel más alto, esto posibilidad que los parámetros se pasen sin que exista transferencia de datos





Tengo hasta 6 ventanas, si quiero añadir una última, se agregará en la v0 y la v0 anterior se transferirá a memoria. Terminada la 7ma, en v0 se transferirá lo que fue puesto en memoria

VARIABLES LOCALES

La ventana de registros es eficiente para variables locales, pero no tiene en cuenta a las globales. Hay dos opciones: una es que el compilador asigne posiciones de memoria a las variables declaradas como globales en el HLL y que todas las instrucciones máquina que referencien esas variables usen operandos referenciados en memoria, pero es ineficiente

Otra opción es incorporar al procesador un conjunto de registros globales. Los cuales serán fijos en cuanto a su número, y accedidos a todos los procedimientos.

El compilador asigna posiciones de memoria a las variables, pero es ineficiente para variables a as que se accede frecuentemente. Incorpora al procesador un conjunto de registros para variables globales, divide registros en la ventana en curso.

AMPLIO BANCO DE REGISTRO VS CACHE

Banco de registros	Cache
Todos los datos escalares locales	Datos escalares locales recientemente usados
Variables individuales	Bloques de memoria
Variables globales asignadas por compilador	Variables locales y globales usadas recientemente
Salvaguarda/restauración basada en profundidad de anidamiento	Salvaguarda/ restauración basada en el algoritmo de reemplazo
Maneja datos individuales	Maneja bloques de datos
Direccionamiento de registro	Direccionamiento de memoria

OPTIMIZACION DE USO DE REGISTROS BASADAS EN EL COMPILADOR

Supongamos un pequeño número de registros. El uso optimizado es responsabilidad del compilador. Los programas HLL no tienen referencias explicitas a los registros. Cada cantidad

del programa candidata se asigna a un registro simbólico. Asignar el número ilimitado de registros simbólicos a un número fijo de registros reales. Si se agotan los registros reales, algunas de las variables se asignan en posiciones de memoria

¿POR QUE CISC?

Apareció para simplificar el compilador. Los compilados eran más pequeños. El programa ocupaba menos memoria cuando la memoria era cara, pero hoy en día dejo de ser una restricción porque es barata.

Tenía una unidad de control más compleja debido a instrucciones complejas.

Había una propensión de usar las instrucciones más sencillas

No está claro que la tendencia hacia CISC fuera apropiada. Cuando se hizo el estudio entre RISC y CISC, ya no había las limitaciones por el cual se usaba CISC

CARACTERISTICAS DEL RISC

- Una instrucción por ciclo de reloj
- Operaciones registro a registro
- Modos de direccionamiento y formatos de instrucción sencillos
- Diseño cableado
- Formado fijo de instrucción
- Mayor tiempo/esfuerzo de compilación
- Térmica de segmentación

RISC FRENTE A CISC

En conclusión, no hay una barrera diferenciadora, muchos diseños incluyen características de ambos. No hay maquina RISC y CISC comparables, las comparaciones son hechas en máquinas juguete.

En RISC el programa será más grande, con mayor velocidad de procesamiento porque es más fácil procesas instrucciones simples que complejas.

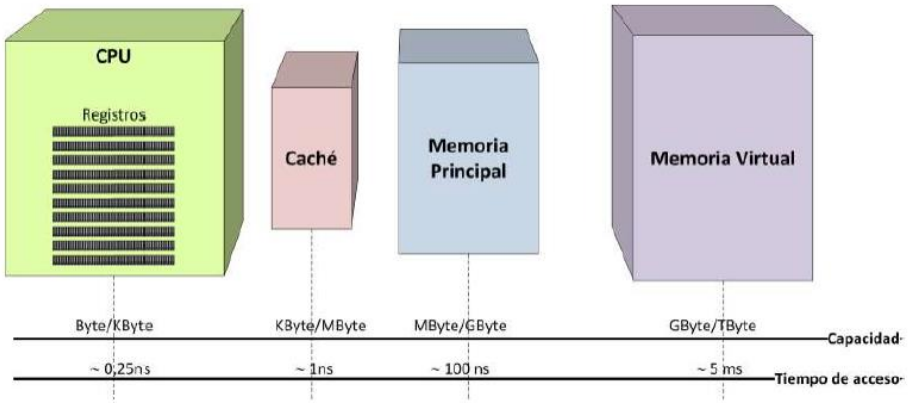
Las diferencias principales son los modos de direccionamiento, el tamaño de instrucciones y la cantidad de registros’

CARACTERISTICAS CISC

- Longitud de instrucciones variables
- Gran cantidad de instrucciones

CLASE 7- Memoria

Sistema de memoria: los programadores desean acceder a cantidades ilimitadas de memoria rápida. La solución es la jerarquía de memoria. Organizada en niveles que son ubicados en distintos lugares físicos con tecnología que se gestiona de manera diferente



Su objetivo es que la velocidad del sistema deberá tener la del nivel más rápido de memoria al costo del nivel más barato. A medida que nos alejamos del CPU, cada nivel inferior es más grande. Lento y barato que el novel precio o superior en la jerarquía. Debe haber correspondencia de direcciones en los distintos niveles.

- Memoria principal (MP): ubicada en un chip diferente al procesador, fabricada con memoria RAM dinámica (D R A M) y controlada por el controlador de memoria principal. Este controlador es muy importante para el rendimiento de la jerarquía de memoria ya que se encarga de la planificación

• Memoria virtual (M V). Ubicada en la actualidad en el disco duro, se fabrica por lo tanto con tecnología magnética y se controla desde el sistema operativo a través del controlador del disco duro.

PROPIEDADES A CUMPLIR

Inclusión: los datos almacenados en un nivel han de estar almacenados en los niveles inferiores a el

Coherencia: las copias de la misma información en los distintos niveles deben contener los mismos valores

¿POR QUE FUNCIONA LA JERARQUIA?

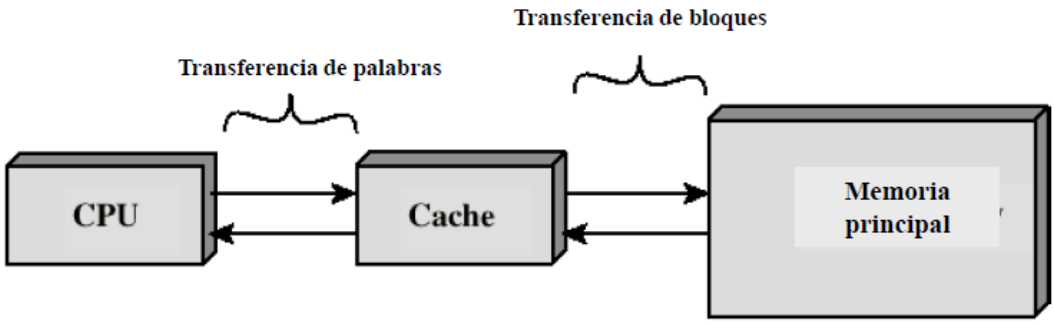
Su objetivo es conseguir una estructura de gran capacidad con bajo coste como el del nivel más bajo de la jerarquía.

Funciona por el principio de localidad de referencias:

- Localidad temporal: los elementos de memoria referenciados recientemente, volverán a serlo en un futuro próximo. En vez de traer palabras, se traen bloques
- Localidad espacial: los elementos de memoria cuyas direcciones están próximas a las ultimas referenciados, serán referenciados

MEMORIA CACHE

Cantidad pequeña de memoria ubicada entre la memoria principal y la CPU. Puede localizarse en un chip o en un módulo CPU. Formada por memoria SRAM, controlada por controladores del cache y puede tener varios niveles. Su objetivo es lograr que la velocidad de memoria sea lo más rápida posible. Contiene copias de la MP.



FUNCIONAMIENTO DE LA CACHE

La CPU solicita contenido de una dirección de memoria. Si la cache tiene ese dato, la CPU la obtiene de la cache. Pero si no está, se lee el bloque que contiene esa dirección desde la memoria principal y la copia en la cache, después la cache entrega el dato requerido a la CPU

La cache incluye etiquetas para identificar que bloque de la memoria principal está trayendo

CONCEPTOS BASICOS

Aciertos: se encuentra en la cache el dato solicitado

Fallo: no se encuentra en cache

Un bloque que contiene la palabra accedida se copia en la memoria principal a una línea de cache

Tiempo para servir el fallo: depende de la latencia y ancho de banda de la memoria principal. O sea, tiempo entre que determina que el bloque no está en la cache y halla el correcto. *Latencia*: tiempo para completar un acceso a memoria, *ancho de banda*: cantidad de información por unidad de tiempo que puede transferirse desde/hacia memoria

Fallos del cache: se gestionan mediante hardware y causan que el procesador se detenga hasta que el dato esté disponible

PRESTACIONES DE JERARQUIA

Tiempo de acceso medio a memoria

$$T_{acceso} = T_{acierto} + T_{fallos\_memoria}$$

$$T_{fallos\_memoria} = \text{Tasa de fallos x Penalización\_fallo}$$

$$T_{acceso} = T_{acierto} + TF \times PF$$

Para mejorar las prestaciones

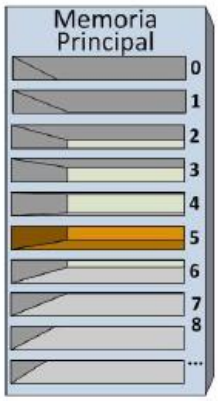
- Reducir el tiempo en caso de acierto ($T_{acierto}$)
- Reducir la tasa de fallos (TF)
- Reducir la penalización por fallo (PF)

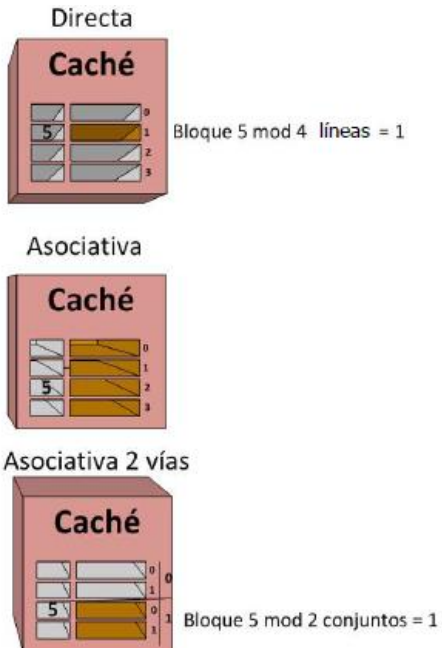
DISENIO DE LA CACHE

Organización: tamaño, costo y niveles. El tamaño de la cache si es pequeño, afecta a la tasa de fallos, pero tampoco debe ser muy grande porque a estar integrado consume más espacio, coste y lento/

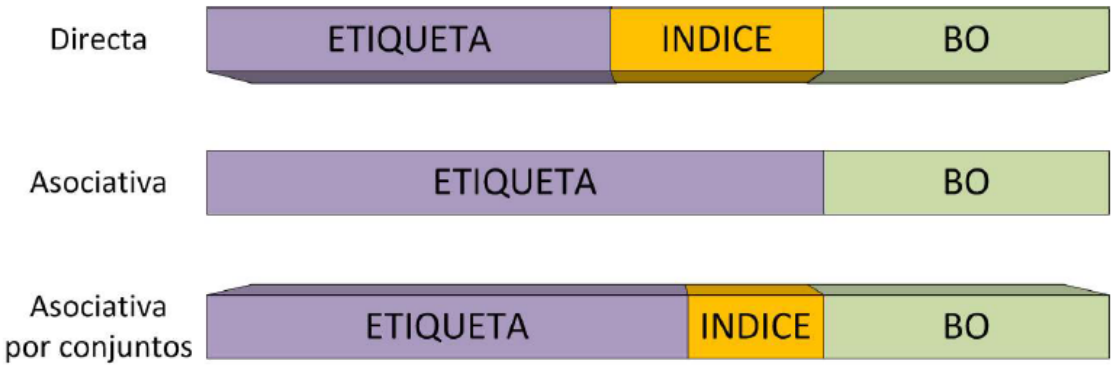
Política de ubicación:

- Correspondencia directa: un bloque solo puede estar almacenada en un lugar de cache. *Nro. línea cache = nro bloque mod nro línea cache*
- Correspondencia total asociativa: un bloque puede almacenarse en cualquier lugar de la cache
- Correspondencia asociativa por conjuntos: un bloque puede almacenarse en un conjunto restringido de lugares en la cache. Un conjunto es un grupo de líneas de la cache. *Nro. de conjunto= nro. bloque referido mod nro. conjuntos de cache*





La interpretación de la dirección física depende del tipo que se use. Índice indicara la línea o el conjunto que le corresponde. BO representa todas las direcciones que pertenecen al bloque



- Correspondencia directa: simple pero costosa. Hay una posición concreta para cada bloque dado. Si un programa accede a 2 bloques que se corresponden a una misma línea de forma repetida, las pérdidas de cache serán grandes
- Correspondencia asociativa: un bloque de memoria principal puede colocarse en cualquier línea de cache. La etiqueta identifica unívocamente un bloque de memoria
- Asociativa de 2 vías: combina lo mejor de otras correspondencias. La cache se divide en un conjunto de grupos, cada conjunto tiene un numero de línea. Un bloque determinado corresponderá a cualquier línea de un conjunto determinado

POLITICA DE REEMPLAZO

Para decidir qué ocurre con un fallo

- Correspondencia asociativa: los algoritmos deben implementarse en hardware
- LRU (menos recientemente usado): requiere controles de tiempo. Uso campo para saber cuántas veces use la línea
- FIFO (primero entrar, primero salir): requiere controles de acceso, se sustituye aquella línea que ha estado más tiempo e la cache. Requiere etiquetas de E/S
- LFU (menos frecuentemente usado): requiere control de uso. Se sustituye aquella línea que ha experimentado menos referencias. Requiere control de tiempo
- Aleatorio: se sustituye cualquier línea al azar

POLITICA DE ESCRITURA

Se debe evitar inconsistencias en el caso de escritura. Tener en cuenta que la CPU escribe sobre una línea cache (el bloque de MP correspondiente debe ser actualizada en algún momento). Un módulo E/S puede tener acceso directo a la MP. En procesamiento paralelo, los múltiples CPU pueden tener caches individuales.

POLITICA DE ESCRITURA EN ACIERTO

Write through (escritura inmediata): se actualizan simultáneamente la posición de la cache y de la MP. Con múltiples CPU, observar el tráfico a memoria principal para mantener actualizada cada cache local. Se genera mucho tráfico y retrasa la escritura. Garantiza que haya distintas copias en toda la jerarquía entonces es coherente

Write back (post escritura): la información solo se actualiza en cache. Cuando se tiene lugar una actualización, se activa un bit ACTUALIZAR asociado a la línea. Después, cuando el bloque es sustituido, en memoria principal si, y solo si, el bit ACTUALIZAR está activo

POLITICA DE ESCRITURA EN FALLO

Ambos incluyen un buffer de escritura para reducir penalización

Write allocate: la información se lleva de la MP a cache. Se sobrescribe en la cache habitual con write back

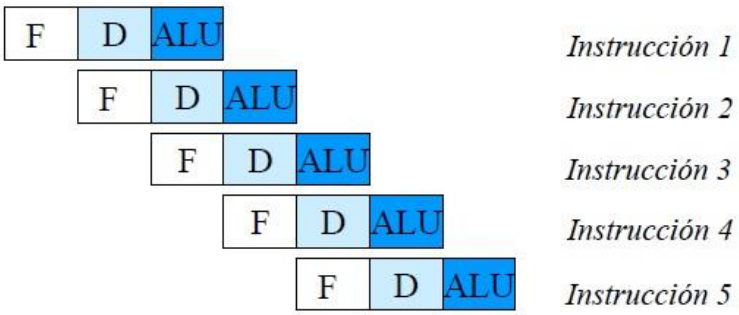
No write allocate: el bloque no se lleva a la memoria cache. Se escribe directamente en MP. Habitual con write through

---

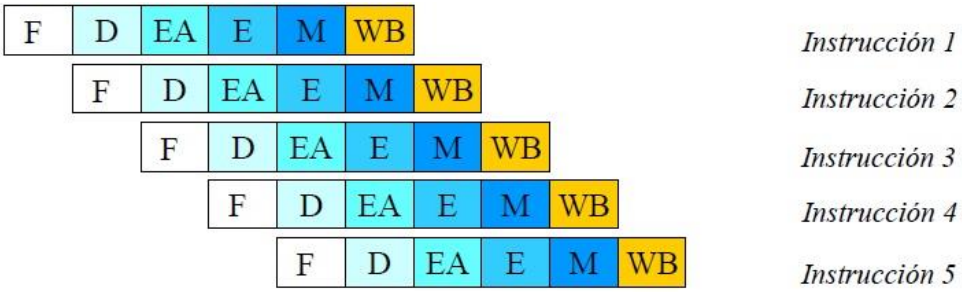
CLASE 8- Procesadores súper escalares

Procesador escalar: ejecuta secuencialmente sus instrucciones

Procesador segmentado de cauce: la unidad sementada es una secuencia de etapas con la propiedad de que nuevas operaciones pueden iniciarse mientras otras están en proceso



Procesador escalar segmentado: técnica orientada al hardware



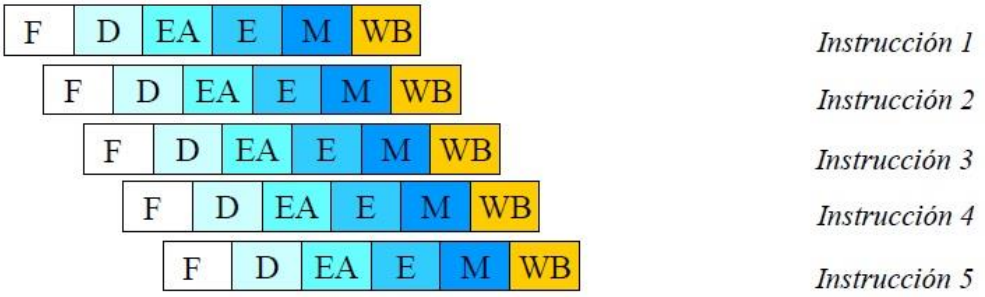
MAYORES PRESTACIONES

La evolución lógica de los diseños de causes segmentados dio lugar a la aparición de dos técnicas de ejecución de mayores prestaciones:

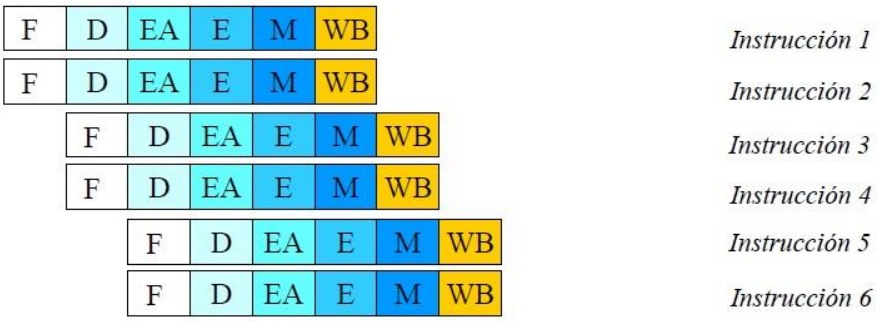
1) Procesadores súper segmentados 2) Procesadores súper escalares

1) Enfoque súper segmentado: muchas operaciones no necesitan un ciclo de reloj por lo tanto obtendremos mayores prestaciones subdividiendo el ciclo de reloj en sub

intervalos. Resulta una mayor frecuencia del ciclo de reloj. Es la división de las etapas del cauce segmentado en sub etapas más pequeñas (y entonces más rápidas) y trasmiten dados a la mayor velocidad del ciclo de reloj. Aumenta el grado de paralelismo. El tiempo para las instrucciones individuales no varía, pero aumenta el grado de paralelismo y mayor la aceleración percibida



2) Enfoque súper escalar: aquel que usa múltiples cauces de instrucciones independientes. Cada cause consta de múltiples etapas, de modo que puede tratar varias instrucciones a la vez. Conlleva la duplicación de algunos o todas las partes del CPU/ALU para captar múltiples instrucciones al mismo tiempo, ejecuta sumas y multiplicaciones simultáneamente, ejecuta carga/almacenamiento, mientras se lleva a cabo una operación ALU
Bastante aceleración, si hay dependencias, no se puede correr simultáneamente, tengo que tener en cuenta cómo son esas instrucciones
El grado de paralelismo y aceleración aumentan ya que se ejecutan más instrucciones en paralelo



PARALELISMO DE INSTRUCCIONES

Grado en que las instrucciones pueden correrse en paralelo. Existe cuando instrucciones de una secuencia son independientes y pueden correrse en paralelo solapándose. Capta varias instrucciones a la vez e intenta encontrar instrucciones que sean independientes entre si

Pero existen limitaciones: dependencias de datos (RAW), *dependencias relativas al procedimiento* (saltos), *conflictos de recursos*, *dependencias de salida* y *anti dependencia*

- Dependencia de datos verdadera: add r1,r2
  move r3, r1
  La segunda instrucción se puede captar y decodificar, pero no ejecutarse hasta que finalice la primera instrucción
- Dependencia relativa al procedimiento: la presencia de bifurcaciones en una secuencia de instrucciones complica el funcionamiento del cauce. Las instrucciones que siguen una bifurcación tienen una dependencia relativa al procedimiento

- Conflictos de recursos: un conflicto de recurso es una lucha de 2 o más instrucciones por el mismo recurso al mismo tiempo. Como memorias, cache, buses o puertos del banco de registro
- Anti dependencia: es una restricción similar a la dependencia verdadera, pero a la inversa; en lugar de que la primera instrucción produzca un valor que usa la segunda, la segunda instrucción destruye un valor que usa la primera

PARALELISMO DE LA MAQUINA

Medida de la capacidad del procesador para sacar partido del paralelismo de instrucciones. Depende del número de instrucciones que pueden captarse y ejecutarse al mismo tiempo, velocidad y sofisticación que usa el procesador para localizar instrucciones independientes

Identificar paralelismo y organizar las etapas de F, D y E en paralelo. Conlleva el renombre de registros y ventana de instrucciones.

Tanto el paralelismo de maquina e instrucciones son factores importantes para mejorar las prestaciones

Sobre las instrucciones: en la localización de instrucciones independientes (orden en que se captan las instrucciones, orden ejecución y orden que se actualizan los registros y posiciones de memoria)

El uso óptimo del cauce es atendiendo dependencias. Un procesador súper escalar capta varias instrucciones a la vez y, a continuación, intenta encontrar cercanas que sean independientes entre sí y puedan, por consiguiente, ejecutarse en paralelo. Si la entrada de una instrucción depende de la salida de otra, la segunda no puede completar su ejecución al mismo tiempo ni antes que la primera.

El procesador puede eliminar algunas dependencias innecesarias mediante uso de registros adicionales y el renombramiento de las referencias a registros.

El procesador debe ser capaz de identificar el paralelismo a nivel de instrucciones y organizar la captación, decodificación y ejecución de las instrucciones en paralelo. El termino **emisión de instrucciones** se refiere al proceso de iniciar la ejecución de instrucciones en las unidades funcionales. Hay distintas políticas de emisión (protocolo para emitir instrucciones) en súper escalar: *emisión y finalización en orden, emisión en orden y finalización desordenada* y *emisión y finalización desordenada*

EMISION Y FINALIZACION EN ORDEN

Es la política más sencilla, pero ingenua. Es útil considerarla como base para el resto de las políticas

Decodificación		Ejecución			Escritura		Ciclo
I1	I2						1
I3	I4	I1	I2				2
I3	I4	I1					3
	I4			I3	I1	I2	4
I5	I6			I4			5
	I6		I5		I3	I4	6
			I6				7
					I5	I6	8

2 Emisión y finalización en orden

EMISION EN ORDEN Y FINALIZACION DESORDENADA

Es usada en los RISC para mejorar la velocidad de las instrucciones que necesitan muchos ciclos. Con la finalización desordenada, puede haber cualquier cantidad de instrucciones en la etapa de ejecución, hasta alcanzar el máximo grado de paralelismo de la máquina, ocupando todas las unidades funcionales. La emisión se para cuándo hay una pugna por un recurso, dependencia de datos o relativa al procedimiento

Decodificación		Ejecución			Escritura		Ciclo
I1	I2						1
I3	I4	I1	I2				2
	I4	I1		I3	I2		3
I5	I6			I4	I1	I3	4
	I6		I5		I4		5
			I6		I5		6
					I6		7

3 Emisión en orden y finalización desordenada

EMISION Y FINALIZACION DESORDENADA

Con la emisión en orden, el reprocesador solo decodifica instrucciones hasta el punto de dependencia o conflicto. No se decodifican más instrucciones hasta que el conflicto se resuelve, por lo tanto, el procesador no podrá buscar más allá del punto del conflicto.

Para permitir la emisión desordenada, es necesario desacoplar las etapas del cauce de decodificación y ejecución. Esto se hace mediante un buffer llamado **ventana de instrucciones.** Cuando un procesador termina de decodificar una instrucción, la coloca en la ventada. Mientras que el buffer no se llene, el procesador continúa captando instrucciones. Cuando una unidad funcional esté disponible, una instrucción pasa del buffer a la unidad. Siempre se ejecutará si no existe conflicto o dependencia y la unidad esté disponible

Decodificación		Ventana	Ejecución			Escritura		Ciclo
I1	I2							1
I3	I4	I1, I2	I1	I2				2
I5	I6	I3, I4	I1		I3	I2		3
		I4, I5, I6		I6	I4	I1	I3	4
		I5		I5		I4	I6	5
						I5		6

4 Emisión y finalización desordenada

RIESGOS

R3: R3 OP R5 {1} 1) RAW. La salida de 1, R3, es uno de los operados en 2

R4: = R3+1 {2} 2) WAW. 1 y 3 intentan escribir en R3

R3: = R5+1 {3} 3) WAR. 3 no puede comenzar hasta que 2 haya obtenido operados

R7: = R3 OP R4 {4}

Renombramiento de registros: para dependencias WAR y WAW, las dependencias de salida y anti dependencias surgen porque los valores de los registros no pueden reflejar la secuencia de valores dictada por el flujo del programa. El hardware del procesador asigna dinámicamente los registros, que están asociados con los valores que necesitan las instrucciones en distintos estantes. Cuando se crea un nuevo valor de registro, se asigna un nuevo registro para ese valor. Las instrucciones posteriores que accedan a ese valor, tienen que sufrir un proceso de renombramiento

Esto puede detener alguna etapa del cauce. Cuando la ejecución de una instrucción guarda el resultado en registro, e almacena en un nuevo registro.

Los registros se asignan automáticamente, o sea, las referencias posteriores son los registros nuevos. El procesador puede eliminar dependencias con registros adicionales y el renombramiento de referencia a registro del código original.

Implementación súper escalar: estrategia de captación simultanea de múltiples instrucciones. Lógica para determinar dependencias verdaderas entre valores de registros y mecanismos para comunicar esos valores. Mecanismos para iniciar o emitir múltiples instrucciones en paralelo. Recursos para la ejecución en paralelo de múltiples instrucciones. Mecanismo para entregar el estado del procesador en un orden correcto.

CONSIDERACIONES DESTACABLES EN EL PROCESADOR SUPER ESCALAR

Excepción: problema. La instrucción que origina la excepción y las que siguen, deben abortarse.

- Influencia de las excepciones: en el momento en que se produce una excepción hay varias instrucciones en ejecución. Para garantizar un estado consistente, las instrucciones anteriores terminan correctamente, la que origina la excepción y las siguientes se abortan y tras la rutina de tratamiento se comienza por a que origino la excepción

Hay un compromiso entre

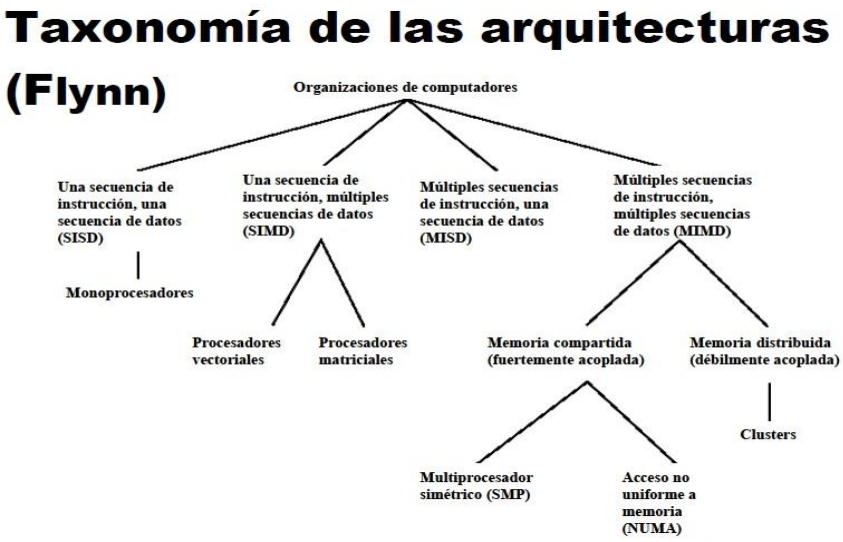
- Ejecución desordenada y liberar las unidades de ejecución
- Completar instrucciones en orden= excepciones precisas, una solución posible es la emisión desordenada y finalización ordenada, con la etapa de escritura temporal y el renombramiento de registros (el registro temporal TWi se usa solo por las instrucciones posteriores a i)

CLASE 9- Procesamiento paralelo

Sea cual sea el nivel de prestaciones, la demanda de máquinas de mayor rendimiento siempre existirá.

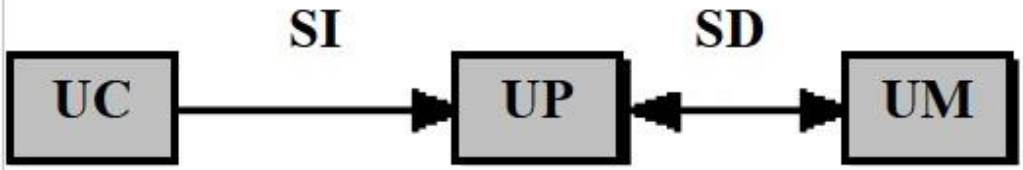
Mejorar el rendimiento de una maquina con un solo procesador= paralelismo a nivel instrucciones

Arquitectura de sistema con varios procesadores= paralelismo a nivel proceso.



CATEGORIAS DE COMPUTADORAS

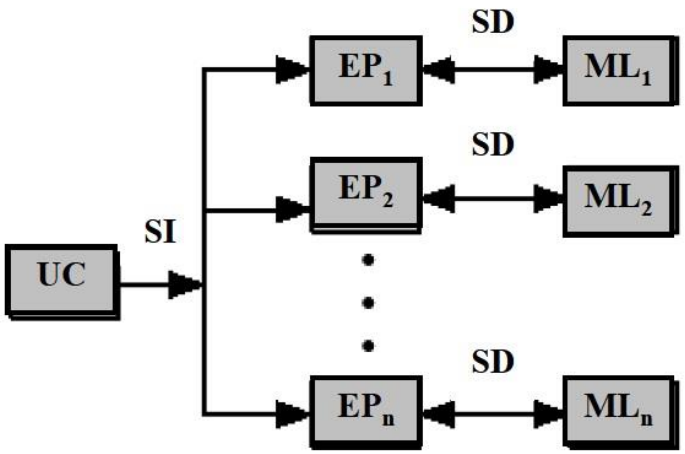
- SISD: una secuencia de instrucciones y una secuencia de datos. Un único procesador interpreta una única secuencia de instrucciones (SI) y datos almacenados en una única



memoria (UM). Son computadoras monoprocesador

- SIMD: una secuencia de instrucciones y múltiple secuencia de datos. Una única instrucción maquina controla paso a paso la ejecución simultanea de un cierto número de elementos en proceso. Cada elemento de proceso tiene una memoria dedicada.

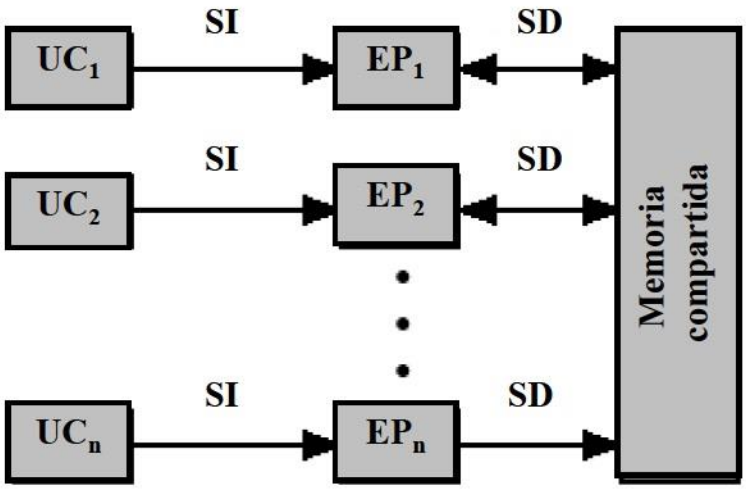
Cada instrucción es ejecutada por cada procesador, con un conjunto de datos diferentes. Son procesadores vectoriales y matriciales (estructura de datos)

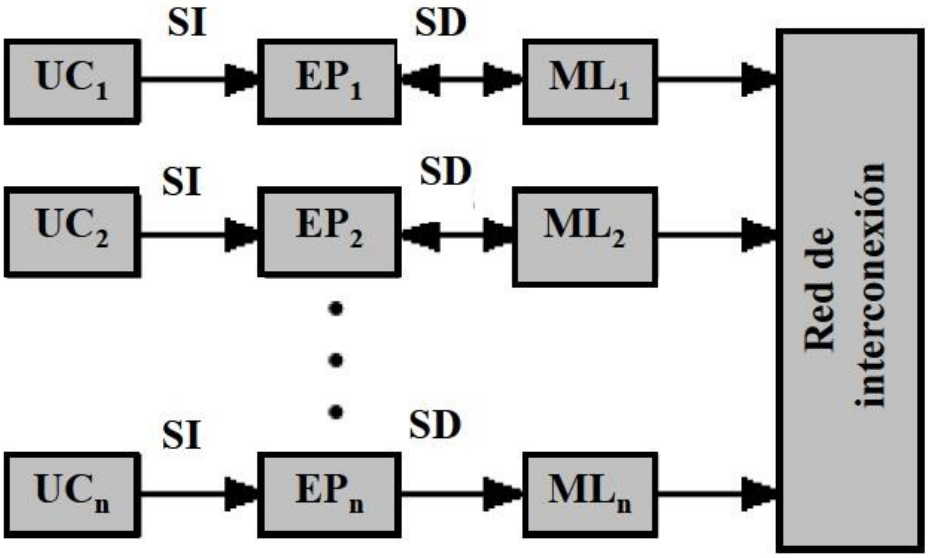


- MISD: múltiples secuencias instrucciones y múltiples secuencias de datos. Se transmite una secuencia de datos a un conjunto de procesadores. Cada procesador ejecuta una secuencia de instrucciones diferentes. Nunca fue implementado

- MIMD: múltiples secuencias de instrucciones y múltiples secuencias de datos. Un conjunto de procesadores ejecuta secuencias de instrucciones diferentes en simultaneo. Se puede dividir según la forma de comunicarse:

  Memoria compartida: SMP (multiprocesadores simétricos) y sistemas NUMA
  Memoria distribuida: clusters





¿MATRICIAL? VECTRORIAL?, ¿PARALELO?

Una computadora con una única “unidad de control” y una matriz de elementos “computacionales”.

Tipos de instrucciones de procesador: extensiones de las instrucciones escalares: sumar, almacenar, multiplicar, etc. Se convierten en operaciones vectoriales ejecutadas en todos los procesadores de modo simultaneo.

Debe añadirse la capacidad de transferir al conjunto de instrucciones los datos escalares y vectoriales entre procesadores: atributos de un “lenguaje paralelo”

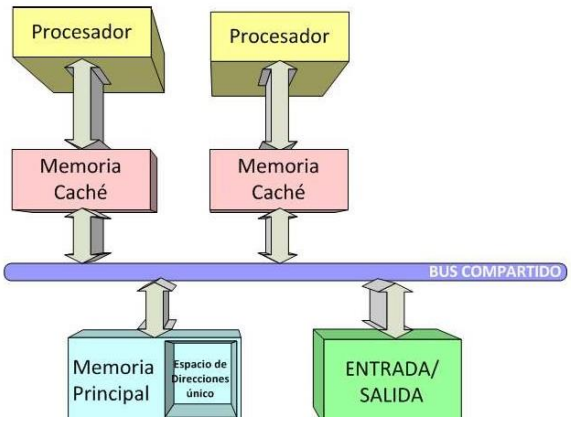
MULTIPROCESADOR SIMETRICO- SMP

Computadora autónoma con las siguientes características:

- Dos o más procesadores similares de capacidades comparables
- Comparten la memoria principal y E/S

- Interconectados mediante un bus u otro tipo de sistema de interconexión
- Tiempo de acceso a memoria similar para todos los procesadores (UMA)
- Todos los procesadores pueden desempeñar las mismas funciones
- Sistema operativo integrado, que proporciona la interacción entre los procesadores y sus programas

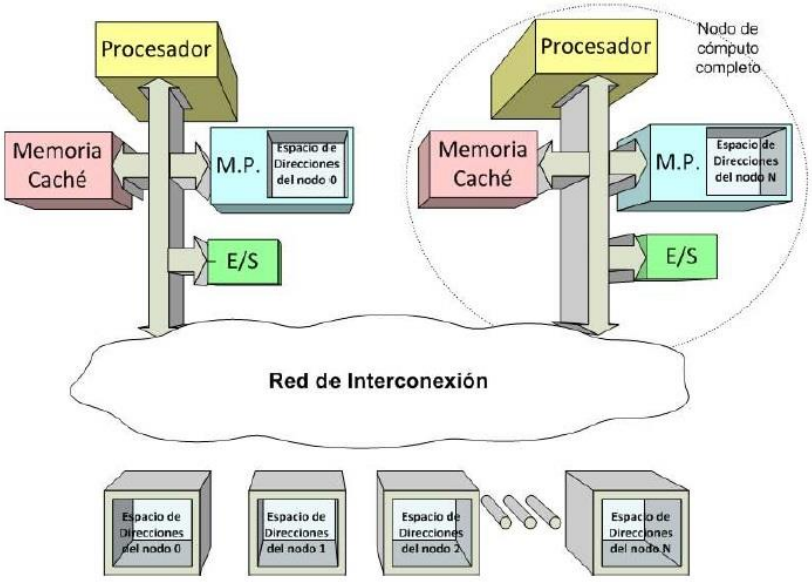
Tiene ciertas ventajas como



- Mayores prestaciones si el trabajo a realizar puede organizarse en paralelo
- Buena disponibilidad, un fallo en un procesador no detendrá la computadora
- Crecimiento incremental, se pueden añadir mas procesadores
- Escalado, en función de la cantidad de procesadores
- Requiere cuidado debido al bus compartido, lo cual tiene sus desventajas.
  La prestación está limitada por el tiempo de ciclo del bus, cada procesador debería estar equipado con una memoria cache para mejorar las prestaciones y reducirá el número de accesos, se pueden producir problemas de coherencia de cache

CLUSTERS

Computadoras completas interconectadas que trabajan conjuntamente como un único recurso, cada computadora se denomina “nodo”



Sus ventajas es la estabilidad absoluta, incremental, alta disponibilidad y mejor relación precio/prestaciones.

- Escalabilidad absoluta: posibilidad de configurar los clústeres grandes
- Escalabilidad incremental: un clúster se configura de forma que sea posible añadir nuevos sistemas de clúster en amplificaciones sucesivas
- Alta disponibilidad: puesto que cada nodo del clúster es un computador autónomo, el fallo de uno no significa la pérdida del servicio
- Mejor relación precio/ prestaciones: al utilizar elementos estandarizados, es posible configurar un clúster con mayor o igual potencia de computo que uno independiente mayor, a menos costo

CLUSTER VS SMP

Ambos dan soporte a aplicaciones de alta demanda de recursos y están disponibles comercialmente

SMP es más fácil de administrar y de configurar. Esta cercano a los sistemas de un solo procesador. La planificación es la diferencia principal. Ocupan menos espacio físico y menor consumo de potencia

CLUSTER: superior escalabilidad incremental y absoluta, superior disponibilidad

TERMINOS UMA, NUMA, CC-NUMA

Multiprocesador de memoria compartida

UMA: Uniform memory Access, igual tiempo de acceso a todas las regiones de memoria e igual tiempo de acceso a memoria para todos los procesadores

NUMA: Nonuniform memory Access, el tiempo de acceso de un procesador difiere dependiendo de la región de memoria que accede, diferentes procesadores acceden a diferentes regiones de memoria a diferentes velocidades

CC-NUMA: cache coherente NUMA, es un NUMA que mantiene coherencia de cache entre las cache de los distintos procesadores.

Operación CC-NUMA: cada procesador tiene cache L1 y L2, cada nodo tiene su propia MP, nodos conectador por algún tipo de red y cada procesador ve un único espacio de direcciones de memoria. Es automático y transparente. El orden de acceso a memoria es

1. Cache L1 (local al procesador)
2. Cache L2 (local al procesador)
3. Memoria principal (local al nodo)
4. Memoria remota

MOTIVACION NUMA

SMP tiene un límite en su número de procesadores y el clusters cada nodo tiene su propia memoria principal (las aplicaciones no “ven” la memoria global, la coherencia de cache se mantiene por software y no por hardware).

NUMA retiene características tipo SMP y brinca múltiple procesamiento a gran escala

El *objetivo* de NUMA es tener una memoria transparente del sistema y permitir nodos, cada uno con su propio bus o sistema de conexión interna

PROCESAMIENTO MULTIHERA

Aumento del paralelismo de instrucciones (sin el aumento de complejidad y consumo de potencia de la segmentación de cause y los súper escalares), la secuencia de instrucciones se

divide en secuencias más pequeñas llamadas hebras que pueden ejecutarse en paralelo, amplia variedad de diseños multihebra

TERMINOS: HEBRA Y PROCESO

Proceso: programa corriendo en una computadora, tiene una propiedad de recursos (despacio de direcciones virtuales) y planificación de ejecución)

Hebra: unidad de trabajo de un proceso que puede asignarse. Incluye un contacto de procesador y área de datos. Ejecutado secuencialmente. Interrumpible

Conmutación de hebra: cambio de control del procesador entre hebras de un mismo proceso

MULTIHEBRA IMPLICITO Y EXPLICITO

Multihera explícito: ejecución concurrente de instrucciones de diferentes hebras explicitas. Mezcla de instrucciones de diferentes hebras en causes compartidos. Todos los procesadores comerciales lo usan

Multihebra implícito: ejecución concurrente de varias hebras extraídas de un único programa secuencial, definidas estáticamente por el compilador o dinámicamente por el hardware

PROCESADOR MULTIHEBRA

PC distinto para cada hebra que pueda ejecutarse concurrentemente. Se trata cada hebra separadamente, aproximaciones con ejecución simultanea real