Introducción

martes, 16 de junio de 2015 11:47

Tipos de paralelismo:

Paralelismo de datos: una misma función, instrucción, etc. Se ejecuta en paralelo sobre un conjunto de datos distinto en cada una de las ejecuciones.

Está implícito en operaciones con estructuras de datos tipo vector o matriz.

Grandes volúmenes de datos independientes entre sí. (Superescalares y máquina segmentada).

Paralelismo funcional: Varias funciones, tareas, instrucciones, etc. Iguales o diferentes se ejecutan en paralelo.

Nivel de instrucción (ILP): las instrucciones de un programa se ejecutan en paralelo. Granularidad fina.

Nivel de blucle o hebra (Thread): Se ejecutan en paralelo distintas iteraciones de un bucle. Granularidad fina/media.

Nivel de procedimiento (Proceso): Distintos procedimientos de un programa se ejecutan en paralelo. Granularidad media.

Nivel de programa: Se ejecutan en paralelo diferentes programas que pueden pertenecer o no a la misma aplicación. Granularidad gruesa.

El paralelismo puede estar de forma implícita por la naturaleza de las estructuras de datos (aunque no puede estar ejecutado de forma paralela) debido a la propia estructura de los datos, o expresarse de forma explícita.

Segmentación

- Identificación de fases en el procesamiento de una tarea.
- Rediseño para implementar cada fase de forma independiente al resto.
- Paralelismo por etapas (el sistema procesa varias tareas al mismo tiempo aunque sea en etapas distintas).
- Se aumenta el número de tareas que se completan por unidad de tiempo.

Riesgos

- Riesgos de datos. Se producen por dependencias entre operandos y resultados de instrucciones distintas.
    - Reorganización de código (intercambio de instrucciones e inserción de NOP)
    - Forwardings
- Riesgos de control. Se originan a partir de instrucciones de salto condicional que, según su resultado, determinan la secuencia de instrucciones que hay que procesar tras ellas.
    - Abortar operaciones
    - Bloqueos o uso de NOP
    - Delayed branch
- Riesgos estructurales o colisiones. Se producen cuando instrucciones diferentes necesitan el mismo recurso al mismo tiempo

Arquitecturas vectoriales

El procesamiento de instrucciones esta segmentado y se utilizan múltiples unidades funcionales.

Utiliza el paralelismo de datos, de forma que cada instrucción vectorial se aplica paralelamente sobre todas las componentes del vector.

Existe un repertorio de instrucciones especializado para dichas operaciones.

Características:

Se calcula cada componente de forma independiente

Cada operación vectorial codifica gran cantidad de cálculos (se reduce el número de instrucciones y se evitan riesgos de control

Se optimiza el uso de memoria

Acceso a memoria:

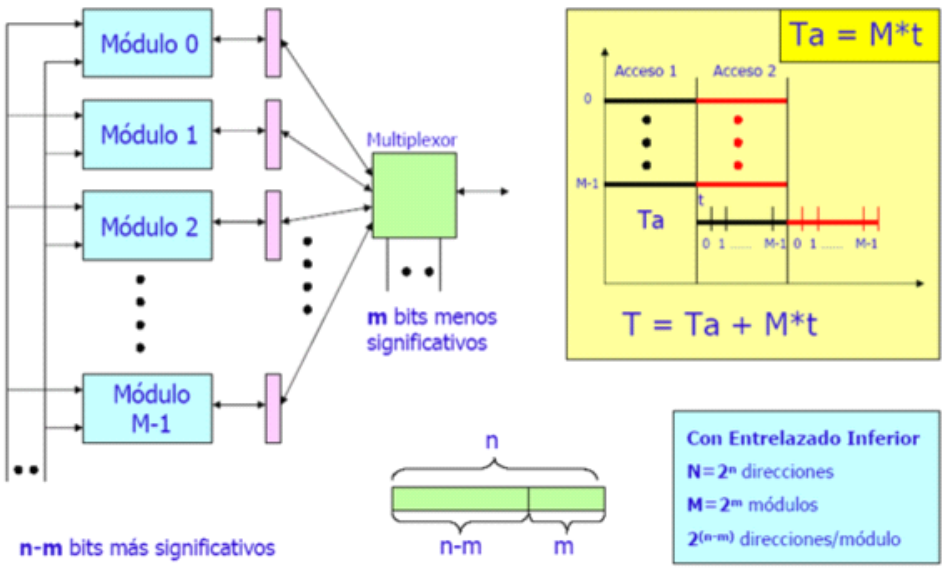
Secuencial o tipo S:

Simultaneamos los accesos en todos los módulos.

El acceso basa su función en superponer, en la medida de lo posible, ambos ciclos (el tiempo de acceso dentro de cada módulo y el tiempo de transferencia entre buffer de memoria y registro donde se necesita la información).

Los “m” módulos acceden todos simultáneamente y mientras se accede se transfieren secuencialmente los datos a través del multiplexor. Funciona muy bien cuando los datos a los que accede son consecutivos, de otra manera pierde rendimiento.

Es síncrono.

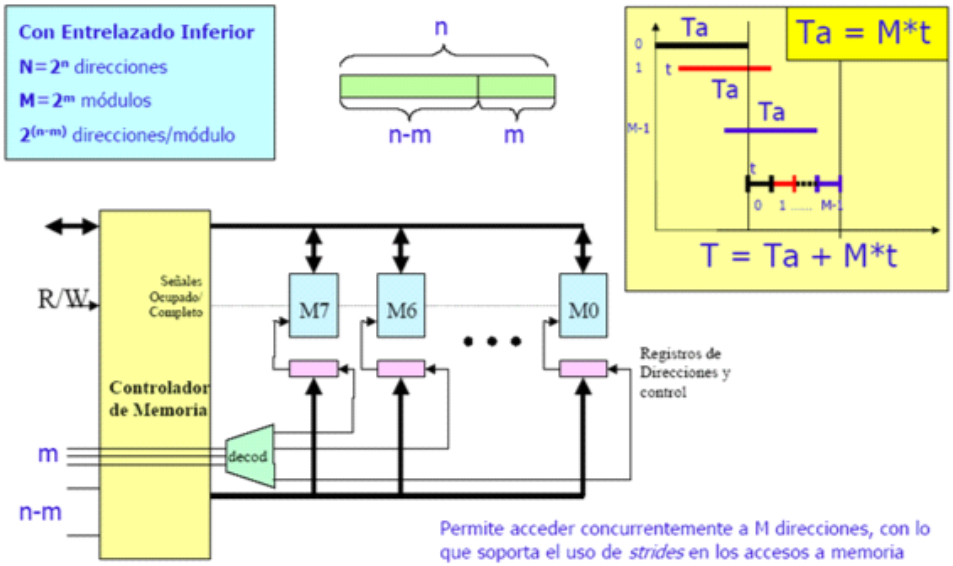


Concurrente o tipo C:

Cada módulo puede trabajar por separado de forma asíncrona leyendo direcciones diferentes. Cada módulo tiene su registro y puede trabajar paralelamente a los demás.

El hardware necesario para controlar peticiones es más complejo.

Actúa bien para vectores contiguos o vectores cuya separación entre elementos sea primo respecto al número de módulos, de otra manera pueden existir conflictos en algún modulo y deben realizarse esperas para solucionarlos.


Con Entrelazado Inferior
N = 2^n direcciones
M = 2^m módulos
2^(n-m) direcciones/módulo
n
n-m
m
Ta
Ta = M*t
0
1
t
Ta
M-1
Ta
t
0 1 ..... M-1
T = Ta + M*t
R/W
Señales Ocupado/ Completo
M7
M6
M0
Registros de Direcciones y control
Controlador de Memoria
m
decod
n-m
Permite acceder concurrentemente a M direcciones, con lo que soporta el uso de strides en los accesos a memoria

Superescalares

martes, 16 de junio de 2015 11:50

Etapas:

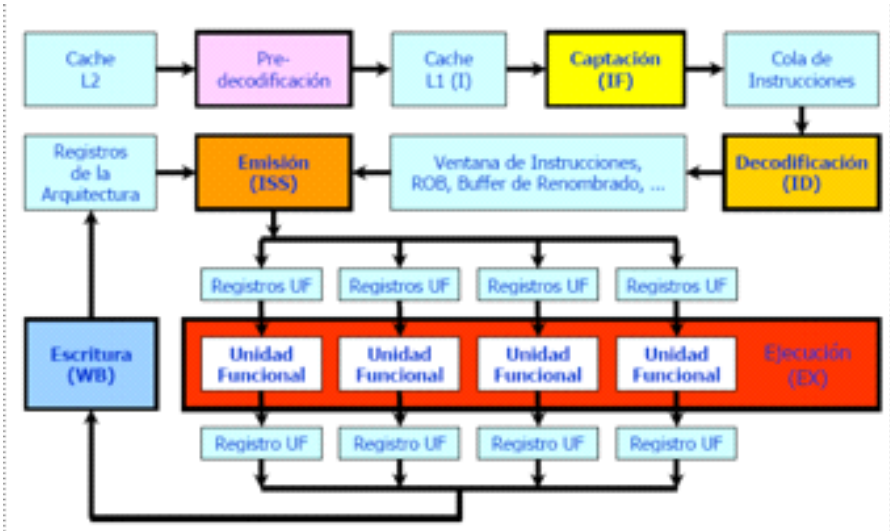
Captación de instrucciones (IF)

Decodificación (ID): buscar operandos, comprobar dependencias, tipo instrucción

Emisión de instrucciones (ISS)

Ejecución (EX)

Escritura (WB)



*Bits de pre-codificación indican:

Si es instrucción de salto

La unidad funcional que necesita

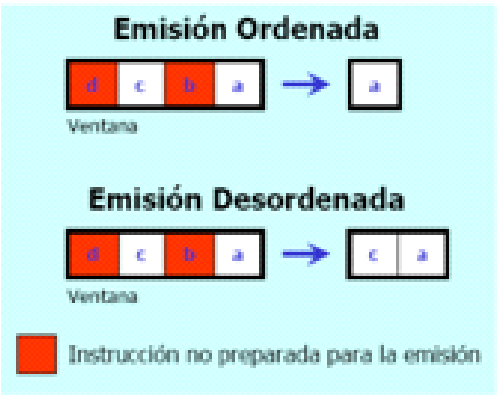
Si hace referencia a memoria o no

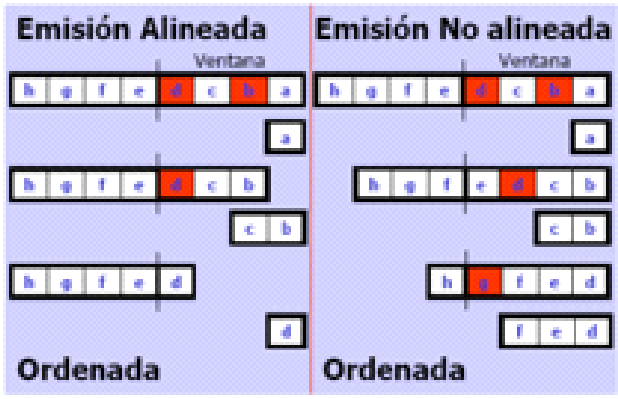
Ventana de instrucciones

La ventana de instrucciones almacena las instrucciones pendientes (centralizada – todas ; distribuida – algunas) ya decodificadas y se utiliza un bit para indicar si esa instrucción está disponible. Por su naturaleza, la emisión de las instrucciones depende del alineamiento y del orden:

Emisión es alineada si no pueden introducirse nuevas instrucciones en la ventana de instrucciones hasta que ésta no esté totalmente vacía o no alineada, mientras que exista espacio en la ventana, se pueden ir introduciendo instrucciones para ser emitidas.

Emisión ordenada se respeta el orden en que las instrucciones se han ido introduciendo en la ventana de instrucciones; si una instrucción incluida en la ventana de instrucciones no puede emitirse, las instrucciones que la siguen tampoco podrán emitirse, aunque tengan sus operandos y la unidad que necesitan esté disponible o desordenada no existe este bloqueo, ya que pueden emitirse todas las instrucciones que dispongan de sus operandos y de la correspondiente unidad funcional.





Estación de reserva

Las estaciones de reserva son similares a la ventana de instrucciones, con la diferencia de que cada estación de reserva corresponde a una unidad funcional.

Es una cola donde están las instrucciones que se van emitiendo para usar la unidad funcional correspondiente, cuando tienen sus operandos disponibles.

Mantiene instrucciones decodificadas hasta que sus operandos sean válidos (instrucción preparada o ready).

La emisión se realiza seleccionando 1 entre los preparados y desactivando el registro ocupado por la instrucción emitida.

Las instrucciones decodificadas deben pasarse a la estación de reserva adecuada (desde la que se pueda acceder a la U.F. correspondiente) y en esta estación donde las instrucciones deben esperar el turno para dicha unidad funcional.

La etapa pasa a llamarse ID/ISS y la segunda parte es el envió, y determina que instrucciones tienen operandos disponibles y pueden pasar a ejecutarse.

Estructura

Buffer de renombrado

En los procesadores con finalización desordenada se producen riesgos por dependencias WAR y WAW y para ello usamos el buffer de renombrado y reorden de los mismos.

Es una técnica para evitar el efecto de las dependencias y antidependencias (WAR y WAW) con el uso de técnicas de desordenación.

Cuando se ejecutan instrucciones que tienen un registro como operando de destino, este no se almacena en el registro directamente sino que se almacena en el buffer de forma auxiliar. Después se escribe el en registro correspondiente el resultado de la última instrucción.

Existen dos mecanismos de acceso a los buffers:

Buffer de Renombrado de Acceso Asociativo

Permite varias escrituras pendientes a un mismo registro

Se utiliza el último bit para marcar cual ha sido la más reciente

Buffer de Renombrado de Acceso Indexado

Sólo permite una escritura pendiente a un mismo registro

Se mantiene la escritura más reciente.

Utiliza los campos: entrada valida, Reg. destino, valor, valor valido, ultimo

Buffer de reorden (ROB)

Campos: nº instrucción (orden), reg. destino, unidad, resultado, ok, marca, ready (ciclo de finalización).

Las instrucciones se introducen en el ROB en orden de programa estricto y pueden estar marcadas como emitidas, en ejecución, o finalizada.

Las instrucciones solo se pueden retirar si han finalizado y todas las que le preceden también.

Existe una tendencia hacia la finalización ordenada de instrucciones (consistencia fuerte), ya que la desordenada no aporta mejoras claras y que gracias al ROB, se permite una emisión y ejecución desordenada que permite el aprovechamiento del paralelismo y tener una finalización ordenada que asegura la semántica del problema.

Al tener una consistencia débil se requiere un hardware adicional que resuelva el WAR y WAW, mientras que si se tiene una consistencia fuerte no se ve penalizado el rendimiento y solo requiere una estructura sencilla como el ROB.

El ROB también se utiliza como buffer de renombrado, ya que se guardan los resultados en su campo resultado.

Permite el adelantamiento especulativo.

Riesgos de control

martes, 16 de junio de 2015 11:54

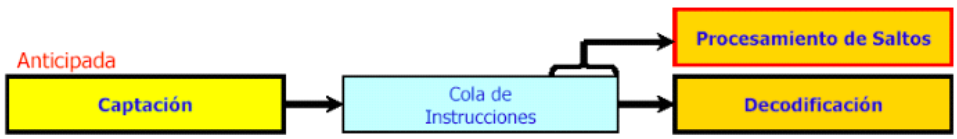
Detección anticipada de instrucciones de salto

Detección en la decodificación

Detección temprana:

- Detección paralela: en paralelo con la decodificación

- Detección anticipada:



- Detección en captación:

	Bloqueo del procesamiento de salto	Se bloquea la instrucción de salto hasta que la condición esté disponible
Gestión de los saltos condicionales no resueltos	Proceso especulativo de los saltos	La ejecución prosigue por el camino más probable (se especula sobre las instrucciones que se ejecutarán). Si se ha errado en la predicción hay que recuperar el camino correcto
	Múltiples caminos	Se ejecutan los dos posibles caminos después de un salto hasta que la condición de salto se evalúa. En ese momento se cancela el camino incorrecto.

Predicción de salto:

Fija : darlo por TOMADO o NOTOMADO

- NOTOMADO: Se calcula la dirección del salto (BTA). Se evalúa la condición y si era buena se borra la BTA y continuamos y sino se captan a través de la BTA.

*más fácil de implementar.

- TOMADO: Se predice como tomado, salva el estado de procesamiento de actividad (PC) y se empieza a partir de la dirección de salto, se evalúa y si es buena se sigue y si no se recupera el estado almacenado.

Verdadera: Estática (Según los atributos de la instrucción) o dinámica (según ejecuciones pasadas)

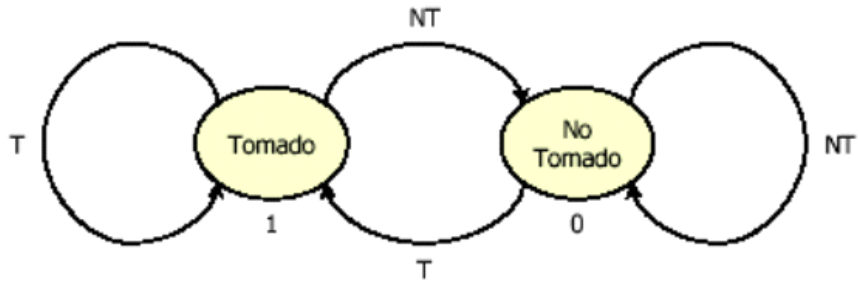
Procesamiento especulativo

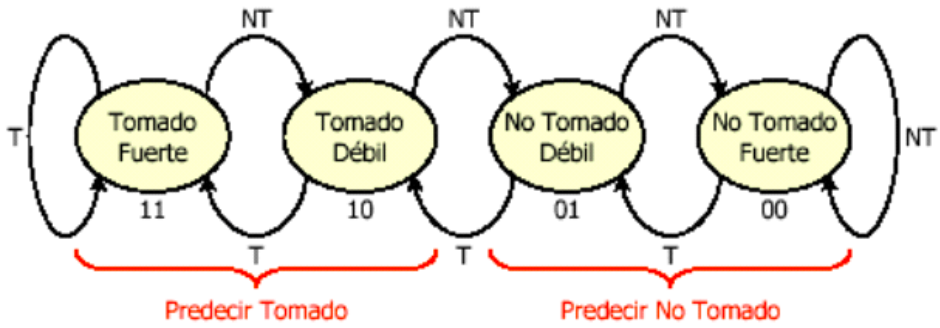
Predicción estática:

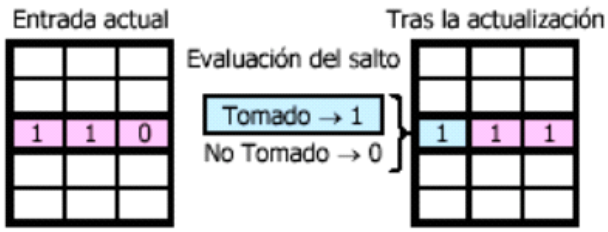
- **Código operación:** ciertas operaciones se predice que salta y otras no.
- **Dirección de salto:** Si es hacia atrás se toma y si es hacia adelante no .
- **Bit de predicción de salto**: el compilador.

Predicción dinámica: Según la historia de si se ha tomado o no , es más probable que sea similar a la anterior. Tiene mejores prestaciones de predicción pero es más costosa.

- **Implícita:** No hay bits de historia, se almacena la dirección de la instrucción que se ejecutó después del salto.
- **Explicita**: Existen bits de historia que codifican información sobre ejecuciones anteriores. Abajo imagen de predicción con 1 bit, dos o tres.







Gestión y estructuras

BTB: contiene los bits de predicción y la dirección destino del salto (BTA)

- Los campos de la BTB se actualizan después de ejecutar el salto
- Para predicción implícita:
  - Tomado: La instrucción se encuentra en la BTB
  - No tomado: La instrucción no se encuentra en la BTB
- Solo se pueden predecir las instrucciones de salto que se encuentran en la BTB
- Bits acoplados
- Campos: tag, dir destino, bits predicción.

BHT: tenemos la BHT con los bits de historia de todas las instrucciones de salto y la BTA con las direcciones destino del salto

- Puede predecir instrucciones que no están en la BTAC
- Aumenta el hardware necesario (dos tablas asociativas)
- Campos: tag, d.destino (BTAC); tag, bits predicción(BTH)

Detección de salto en captación:

- Cuando se capta una instrucción de salto en la cache se accede paralelamente a los bits de predicción

- Utiliza BTB independiente
- Se añade índice sucesor en la cache de instrucciones
- Puede predecir instrucciones que no están en la BTB
- No añade gran cantidad extra de hardware

Paralelismo

martes, 16 de junio de 2015 12:09

Clasificación

Multiprocesadores:

- Comparten el mismo espacio de memoria (el programador no necesita saber dónde están los datos)
- Mayor latencia
- Poco escalable: Cuantos más procesadores, mas latencia
- Comunicación mediante variables compartidas
- Implementa primitivas de sincronización
- No distribuye código ni datos
- Programación más sencilla

Multicomputadores:

- Cada procesador (nodo) tiene su propio espacio de direcciones (el programador necesita saber dónde están los datos)
- Menor latencia
- Mayor escalabilidad
- Comunicación mediante paso de mensajes
- Sincronización mediante mecanismos de comunicación
- Distribución de carga de trabajo entre procesadores
- Programación mas difícil

Tipos de paralelismo

Paralelismo funcional

Se obtiene a través de la reorganización de la estructura lógica de una aplicación. Existen diferentes niveles de paralelismo funcional según las estructuras que se reorganicen.

Paralelismo de datos

Implícito en operaciones con estructuras de datos tipo vector o matriz. Está relacionado con operaciones realizadas sobre grandes volúmenes de datos que sean independientes entre si.

Paralelismo explicito

Paralelismo no presente de forma inherente en las estructuras de programación y que se debe indicar expresamente.

Paralelismo implícito

Paralelismo presente (aunque puede no estar ejecutándose de forma paralela) debido a la propia estructura de los datos (vectores) o de la aplicación (bucle)

Problemas de la programación paralela

- Dividir en unidades de computo independientes (tareas o subproblemas)
- Agrupación de tareas en procesos/threads
- Asignación a procesadores
- Sincronización y comunicación

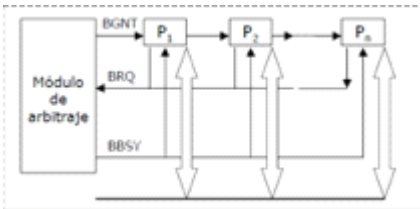
Redes Interconexión

martes, 16 de junio de 2015 13:34

Clasificación

Redes de medio compartido

- Medio de transmisión compartido.
- Arbitraje
- Ancho de banda limitado (escalabilidad limitada)
- Arquitectura UMA
- Clasificación según arbitraje de bus:
  - Señales de control: BRQ (solicitar bus), BGNT (bus garantizado), BBSY (bus ocupado).
  - Prioridad estática:
  - Centralizada-serie

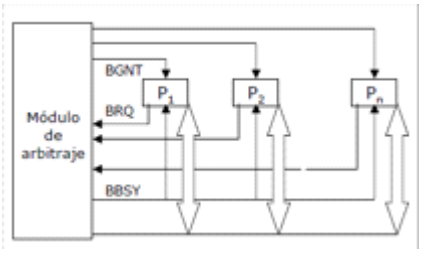


BRQ común

BGNT propagado

BBSY común

  - Centralizada Paralela

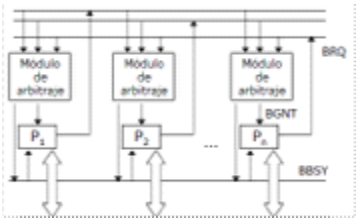


BRQ individual

BGNT individual

BBSY común

  - Distribuida Paralela



BRQ individual

BGNT individual

BBSY común

El criterio más importante para la clasificación de las redes de interconexión se basa en la rigidez de los enlaces. La estática tiene una topología establecida de manera definitiva y estable, y la única manera de modificación es para la ampliación. La dinámica puede variar en la ejecución.

Redes directas

Características

Nodos conectados a subconjuntos de nodos

Escalabilidad baja

Router - comunicación nodos

Canales uni o bidireccionales

Propiedades

Grado

Diámetro

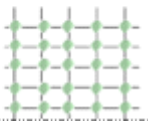
Regularidad(Todos los nodos tienen el mismo grado)

Simetría(se ve semejante desde cualquier nodo)

Topologias

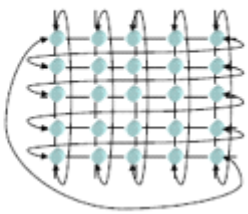
Malla abierta

- F. interconexión:
  - $F_{+1}(i) = (i+1)$ si i mod r <> r-1
  - $F_{-1}(i) = (i-1)$ si i mod r <> 0
  - $F_{+r}(i) = (i+r)$ si i div r <> r-1
  - $F_{-r}(i) = (i-r)$ si i div r<>0
- Grado: 4
- Diámetro: 2(r-1), donde N=r2



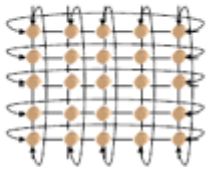
Malla Illiac

- F. interconexión:
  - $F_{+1}(i) = (i+1) \bmod N$
  - $F_{-1}(i) = (i-1) \bmod N$
  - $F_{+r}(i) = (i+r) \bmod N$
  - $F_{-r}(i) = (i-r) \bmod N$
- Grado: 4
- Diámetro: (r-1), donde $N=r^2$



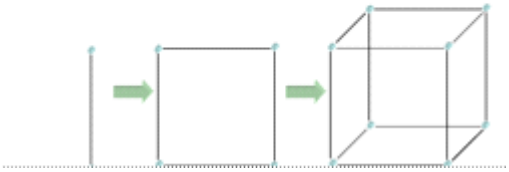
Toro

- n dimensiones, k nodos
- F. interconexión toro 2D:
  - $F_{+1}(i) = (i+1) \bmod r + (i \text{ DIV } r) \cdot r$
  - $F_{-1}(i) = (i-1) \bmod r + (i \text{ DIV } r) \cdot r$
  - $F_{+r}(i) = (i+r) \bmod N$
  - $F_{-r}(i) = (i-r) \bmod N$
- Grado: 4
- Diámetro: $2 \cdot \left\lfloor \frac{r}{2} \right\rfloor$, donde $N=r^2$



Hipercubo

- F. interconexión:
  - $F_i(h_{n-1}, \ldots, h_i, \ldots h_0) = h_{n-1}, \ldots, \bar{h}_i, \ldots h_0$
- Grado: n (n=log N)
- Diámetro: n



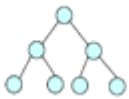
Barril y anillo

- F. interconexión: $F_{+1}(i) = (i+1) \bmod N$
- Grado de entrada/salida: 1/1
- Diámetro: N-1

- ¿Anillo bidireccional?

Árbol

- Balanceado: todas las ramas del árbol tienen la misma longitud
- Cuello de botella → nodo raíz
- N (balanceado)= 2^k-1 (k = niveles del árbol)
- Grado: 3
- Diámetro: 2(k-1)



Redes Indirectas

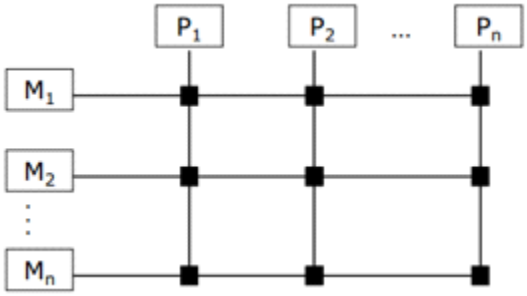
Crossbar:

Conexión directa nodo- nodo, gran ancho de banda y capacidad de interconexión; limitado por los accesos a memoria, coste elevado.

Cada nodo está conectado con todos los demás a través de un conmutador de líneas cruzadas. La red de líneas cruzadas puede interpretarse como una central telefónica que conecta los nodos en función de las necesidades de cada momento.

Ventajas: no son bloqueantes y son fácilmente escalables.

Inconvenientes: coste elevado O(N*M)



Redes MIN:

Conectan dispositivos de entrada con dispositivos de salida mediante un conjunto de etapas de conmutadores, donde cada conmutador es una red Crossbar.

Ventajas: Menor coste Nlogn

Inconvenientes: Son bloqueantes.

- **Omega**: Red de N entradas formado por log2N etapas de N/2 modulos comp 2x2. Tendrá una red (N/2)log2N . Cada uno es independiente. El patrón de interconexión es perfect shuffle.
- **Mariposa**: k^nxk^n formada por n columnas de k ^n-1 conmutador kxk unidos por n-1 permutaciones mariposa de base k
- **Cubo**: baraje perfecto perfect shuffle
- **Delta**: interconexión a^nxb^n con n etapas formado por modulos conmutados. A-shuffle

Memoria

martes, 16 de junio de 2015 14:07

Clasificación de multiprocesadores según la distribución de memoria

UMA (Uniform Memory Access)

Memoria centralizada, el tiempo de acceso a memoria por parte de todos los procesadores es el mismo. El tiempo de acceso de un procesador a cualquier parte de la memoria es el mismo

Fuertemente acoplado: alto grado de compartición de recursos

Cada procesador puede disponer de su cache

Periféricos compartidos entre procesadores

Sincronización y comunicación mediante variables compartidas

Acceso a memoria, periféricos, etc.

Equitativo: Symmetric Multi Processors (SMPs)

No equitativo: Asymmetric Multi Processors (AMP)

NUMA (Non-Uniform Memory Access)

Débilmente acoplado: cada procesador dispone de una memoria local a la que puede acceder mas rápidamente.

El tiempo de acceso a memoria de cada procesador depende de la región a la que acceda

Sistemas de acceso a memoria: local, global, local de otros procesadores

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

Ventajas:

Escalado de memoria con + coste/rendimiento

Reducción de latencia de acceso a memorias locales

COMA (Cache-Only Memory Architecture)

Caso especial de los sistemas NUMA

Las memorias distribuidas se convierten en caches. La capacidad de memoria global es limitada

Las caches forman un mismo espacio global de direcciones

Acceso a las cachés por directorio distribuido

Consistencia y coherencia de memoria caché

En los sistemas multiprocesador contemporáneos es común disponer de uno o dos niveles de caché asociados a cada procesador. Esta organización es esencial para conseguir unas prestaciones razonables (por el tema de la latencia, si todos quieren acceder siempre a la memoria por un mismo bus no sería eficiente).

Esto origina un problema conocido como coherencia de caché: pueden existir varias copias del mismo dato (de la RAM) simultáneamente en caches diferentes, y si los procesadores actualizan sus copias, puede producirse una visión inconsistente de la memoria.

Un sistema de memoria es coherente si cualquier lectura de un dato devuelve el valor más reciente.

Aspectos críticos del sistema de memoria compartida: los datos devueltos por una lectura (coherencia) y cuando un valor escrito será devuelto por una lectura (consistencia)

Estrategias para abordar la coherencia pueden ser:

- Resolución software
- Resolución hardware, la más utilizada

Políticas para mantener la coherencia:

- Invalidación de escritura o coherencia dinámica: siempre que un procesador modifica un dato de un bloque en la caché, invalida todas las demás copias de ese bloque guardadas en las otras caches de los procesadores.
- Actualización en escritura: esta política actualiza la copia en las demás cachés en vez de invalidarla.

Protocolos de coherencia

Protocolos de sondeo o snoopy:

Los protocolos de sondeo distribuyen la responsabilidad de mantener la coherencia de cache entre todos los procesadores.

- Una cache debe reconocer cuando una línea de las que contiene esta compartida con otras caches.
- Cuando se realiza una actualización en una línea de cache compartida debe anunciarse a todas las otras caches mediante broadcast.
- Cada controlador de cache es capaz de sondear o “espiar” (snoop) la red para observar las notificaciones que se difunden y reaccionar adecuadamente.

Los protocolos de sondeo se adaptan bien a los multiprocesadores basados en un bus, puesto que el bus compartido proporciona una forma sencilla para la difusión y el sondeo. No obstante, puesto que uno de los objetivos de utilizar caches locales es evitar los accesos al bus, hay que cuidar el incremento en el tráfico del bus que requiere la difusión y el sondeo no anule los beneficios de las caches locales.

Todas las transacciones aparecen en el bus y son visibles para los procesadores en el mismo orden en el que se producen

Dos enfoques básicos

Invalidación en escritura: Inicialmente, una línea puede compartirse por varias caches con el propósito de lectura. Cuando se quiere hacer una escritura en la línea de una cache, primero envía una notificación que invalida la línea en las otras caches, haciendo que dicha línea sea exclusiva para la cache donde se va a escribir. Una vez que la línea es exclusiva, el procesador puede realizar escrituras locales en la misma hasta que otro solicite la misma línea.

Actualización en escritura: Puede haber varios procesadores que escriben igual que varios procesadores que leen. Cuando un procesador desea escribir en una línea compartida, la palabra a actualizar se distribuye a los demás, y las caches que contienen esa línea lo pueden actualizar.

Menos utilizado que el anterior.

Protocolo MSI

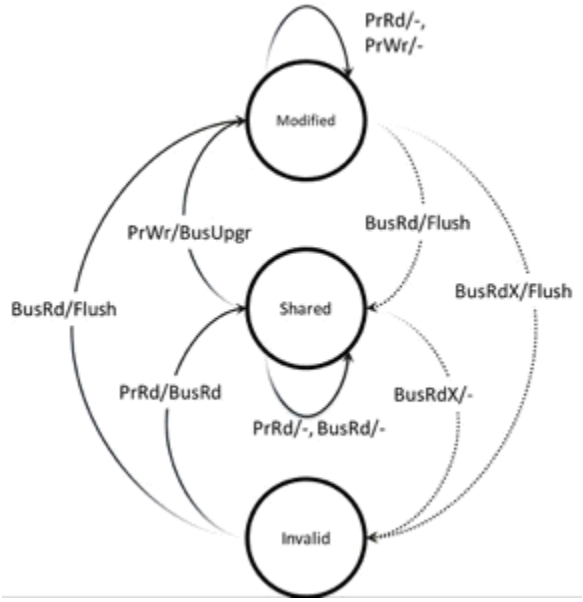
Protocolo de invalidación

Estados:

Invalido: la línea cache no contiene datos validos

Compartido: la línea de cache tiene el mismo contenido que la RAM y puede estar presente en otra caché o no

Modificado: la línea de cache ha sido modificada y está disponible solo en esta cache



Protocolo MESI

Ampliación de MSI. Refinamiento para aplicaciones “secuenciales” que corren en multiprocesadores

En MSI el programa cuando lee y modifica un dato tiene que generar 2 transacciones incluso en el caso de que no exista compartición (solo presente en una cache) del dato.

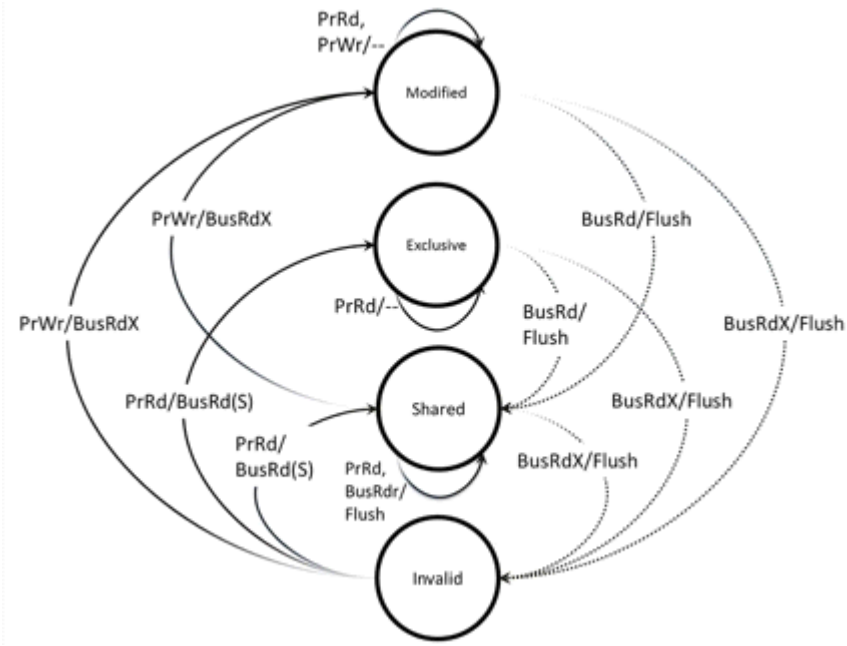
Estados

***Exclusivo** (E): la línea cache tiene el mismo contenido que en memoria principal y no está presente en ninguna otra cache. Al ser exclusivo es posible realizar una escritura o pasar al estado modificado sin ninguna transacción en el bus, al contrario que en el caso de estar en estado compartido

Invalido: la línea cache no contiene datos validos

Compartido: la línea de cache tiene el mismo contenido que la RAM y puede estar presente en otra caché o no

Modificado: la línea de cache ha sido modificada y está disponible solo en esta cache


PrRd,
PrWr/--
Modified
PrWr/BusRdX
Exclusive
BusRd/Flush
PrRd/--
BusRd/
Flush
BusRdX/Flush
PrWr/BusRdX
PrRd/BusRd(S)
Shared
BusRdX/Flush
PrRd/
BusRd(S)
PrRd,
BusRdr/
Flush
BusRdX/Flush
Invalid

Preguntas Examen

martes, 16 de junio de 2015 14:14

ENERO 2013

Explicar la utilidad del buffer de renombramiento y enumera los tipos que hay según su direccionamiento.

Diferencia entre predicción dinámica explicita y predicción dinámica implícita.

La predicción de saltos dinámica se basa en el resultado de la instrucción en ejecuciones pasadas.

Explicita: Existen unos bits que codifican la información sobre la historia (ejecuciones pasadas) de dicha instrucción.

Implícita: No existen bits de historia, se almacena la dirección de la instrucción que se ejecutó después de la instrucción de salto en cuestión.

Justifica la diferencia entre multicomputadores y multiprocesadores en términos de latencia y escalabilidad.

Los multiprocesadores tienen mayor latencia y menor escalabilidad que los multicomputadores.

Latencia: Debido a que los multiprocesadores comparten recursos, se deben implementar métodos de sincronización y concurrencia, lo que provoca un aumento de latencia.

Escalabilidad: La arquitectura de multicomputadores surge en parte con el objetivo de poder aumentar el rango de escalabilidad de los multiprocesadores. Cuantos más procesadores participan en el multiprocesador, mayor es la latencia por lo que son poco escalables. Mediante los supercomputadores podemos conseguir la participación de un mayor número de procesadores o maquinas sin que el rendimiento del sistema caiga.

JULIO 2013

Explica la utilidad de la ventana de instrucciones y explica brevemente cómo puede ser el orden de emisión y el alineamiento de la ventana

La ventana de instrucciones se utiliza para controlar la ejecución de instrucciones de forma paralela. Esta almacena las instrucciones pendientes ya decodificadas y se utiliza un bit para indicar si la instrucción está disponible. Una instrucción estará disponible cuando tenga todos sus operandos disponibles y la unidad funcional donde se procesará.

Alineamiento:

Alineada: No pueden introducirse nuevas instrucciones en la ventana hasta que esta no se vacíe.

No alineada: Se pueden introducir nuevas instrucciones en la ventana en el espacio que vaya quedando libre.

Orden:

Ordenada: Se respeta el orden de las instrucciones y si una instrucción no puede ser emitida por algún motivo, las posteriores no podrán emitirse aunque estén disponibles hasta que la primera se realice.

Desordenada: Si una instrucción no está disponible en un momento determinado, podrá emitirse las siguientes sin que se produzca ningún bloqueo, siempre que estas estén disponibles.

Explica brevemente las estructuras que permiten la gestión de predicción de los saltos dinámica explícita

Bits acoplados

Branch Target Buffer (BTB) es una pequeña memoria asociativa que guarda las direcciones de los últimos saltos ejecutados así como su destino. A su vez guarda información que permite predecir si el salto será tomado o no.

Solo pueden predecirse los saltos que están en la BTB.

Bits desacoplados

Branch History Table (BHT) almacena los bits de predicción de todas las instrucciones de salto condicional.

Branch Target Address Cache (BTAC) almacena las direcciones destino de los últimos saltos ejecutados.

l-caché almacena los bits de predicción de la instrucción de salto en la caché. También se añade en la cache un índice sucesor que indica mediante una tabla BTB independiente la dirección destino del salto.

Explica brevemente qué tipos de paralelismo existen y en qué niveles puede aplicarse

Paralelismo de datos: una misma función, instrucción, etc. Se ejecuta en paralelo sobre un conjunto de datos distinto en cada una de las ejecuciones.

Paralelismo funcional: Varias funciones, tareas, instrucciones, etc. Iguales o diferentes se ejecutan en paralelo.

Nivel de instrucción (ILP): las instrucciones de un programa se ejecutan en paralelo. Granulidad fina.

Nivel de blucle o hebra (Thread): Se ejecutan en paralelo distintas iteraciones de un bucle. Granulidad fina/media.

Nivel de procedimiento (Proceso): Distintos procedimientos de un programa se ejecutan en paralelo. Granulidad media.

Nivel de programa: Se ejecutan en paralelo diferentes programas que pueden pertenecer o no a la misma aplicación. Granulidad gruesa.

Enero 2013

Explica la utilidad del buffer de renombrado y enumera los tipos que hay según su direccionamiento

El buffer de renombrado es una estructura que da soporte para poder realizar el renombrado de registros. Esta técnica sirve para poder utilizar técnicas de desordenación que permitan aprovechar mejor la paralización sin riesgos por dependencias WAR y WAW. Cuando se está ejecutando alguna instrucción que tenga un registro como destino de escritura, dicha escritura no se realizara directamente sobre el registro correspondiente, sino que se almacenara el

resultado en un buffer auxiliar donde podrá ser consultado por otras instrucciones de lectura.

Existen dos tipos según su direccionamiento:

- Acceso asociativo:
  - Permite varias escrituras pendientes a un mismo registro
  - Utiliza un bit para especificar cual es la ultima escritura
- Acceso indexado:
  - Solo permite una escritura pendiente a un mismo registro
  - Solo se conserva el resultado correspondiente a la escritura mas actual

Explica la diferencia entre predicción dinámica explicita y predicción dinámica implícita.

La predicción dinámica significa que la predicción que se va a realizar sobre una instrucción de salto puede variar en cada ejecución.

La diferencia entre la predicción dinámica explicita y la implícita es que en la explicita se utilizan bits de historia que almacenan información sobre las ejecuciones pasadas para las futuras predicciones, mientras que en la implícita, únicamente se almacena la dirección de la instrucción a la que se accedió después de la última ejecución de la instrucción de salto.

Justifica la diferencia que existe entre multicomputadores y multiprocesadores en términos de latencia y escalabilidad.

Los multicomputadores tienen una mayor escalabilidad que los multiprocesadores.

Esto se debe a que los multiprocesadores hacen un uso compartido de los recursos del sistema, por lo que cada procesador debe competir por el uso de estos, aparte de que la comunicación entre ellos se realiza mediante un único bus. Por esta razón, cuantos más procesadores participen en el multiprocesador, mayor latencia existirá.

Los multiprocesadores no tienen este problema ya que cada nodo dispone de sus propios recursos

Cuál es la característica distintiva en las redes de interconexión dinámicas frente a otros tipos de redes de interconexión?

La principal diferencia es que en las redes dinámicas, los enlaces entre nodos (topología) pueden ir variando según la necesidad de cada instante, mientras que en las redes estáticas la topología queda establecida de forma fija cuando se instala el sistema.