**CLASE 8:**

**Procesadores supersegmentados y superescalares**

**1) Procesadores supersegmentados**

En la ejecución supersegmentada de instrucciones cada ciclo se divide en fracciones más chicas, en las que se inician nuevas instrucciones.

La ejecución supersegmentada consiste en subdividir cada segmento en partes más pequeñas.

Como muchas operaciones no necesitan todo un ciclo de reloj, se puede hacer más de una tarea en ese ciclo, subdividiendo el ciclo de reloj en sub-intervalos, lo que es equivalente a usar una mayor frecuencia de reloj (menor ciclo de reloj).

El tiempo para las instrucciones individuales no varía, pero aumenta el grado del paralelismo de la máquina temporal.

El pipeline se hace más profundo, pero está limitado por la tecnología (es decir, por la frecuencia máxima).

**2) Procesadores superescalares**

En la ejecución superescalar de instrucciones, en cada ciclo se inician más de 1 instrucción simultáneamente.

Para poder iniciar/ejecutar 2 o más instrucciones simultáneamente, se requiere la duplicación de algunas o todas las partes de la CPU/ALU, por ejemplo, la captación de múltiples instrucciones al mismo tiempo, la ejecución (sumas y multiplicaciones) simultánea y la ejecución de carga/almacenamiento, mientras se lleva a cabo una operación en ALU.

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

Un procesador superescalar dispone, básicamente, de múltiples unidades funcionales, cada una implementada como un cauce segmentado, que admite la ejecución paralela de varias instrucciones.

Superescalar frente a supersegmentado:

Una solución para alcanzar mejores prestaciones es la llamada supersegmentacion. Esta aprovecha el hecho de que muchas etapas del cauce realizan tareas que requieren menos de medio ciclo de reloj. De este modo, doblando la velocidad de reloj interna se permite la realización de dos tareas en un ciclo de reloj externo.

Paralelismo de instrucciones:

La expresión paralelismo de instrucciones se refiere al grado en el que las instrucciones de un programa se pueden ejecutar en paralelo.

Limitaciones fundamentales del paralelismo a las que el sistema tiene que enfrentarse:

* Dependencia de datos verdadera
* Dependencia relativa al procedimiento
* Conflicto en los recursos
* Dependencia de salida
* Antidependencia

Paralelismo de la maquina:

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

Sobre las instrucciones:

En los procesadores superescalares el objetivo fundamental es localizar instrucciones que puedan ser introducidas al pipeline y ejecutadas. En este proceso de detección, es necesario distinguir:

* El orden en que se captan las instrucciones.
* El orden en que se ejecutan las instrucciones.
* El orden en que las instrucciones actualizan los registros y las posiciones de memoria.

Cuanto más sofisticado es el procesador, menos restricciones impone a estos ordenamientos. Incluso puede alterar cualquier ordenamiento, respecto del estrictamente secuencial. La única condición es que el resultado debe ser correcto.

Las políticas de emisión de instrucciones son los protocolos usados para el envío de las instrucciones a las unidades funcionales, es decir, definen la forma en que se captan, ejecutan y finalizan (escriben los resultados) las instrucciones.

En función del orden en que se captan, ejecutan y terminan las instrucciones, existen 3 políticas de emisión de instrucciones:

* Emisión y finalización ordenada
* Emisión ordenada y finalización desordenada
* Emisión y finalización desordenada

Emisión y finalización ordenada: En esta política, las instrucciones se emiten exactamente como están en el programa (como si fuera “secuencial”), y se escriben los resultados en el mismo orden.

Emisión ordenada y finalización desordenada: En esta política, las instrucciones se emiten exactamente como está en el programa, pero los resultados se escriben en “cualquier orden”. Las mismas instrucciones se ejecutan de la siguiente manera.

Emisión ordenada y finalización desordenada – Conclusiones: Las instrucciones entran a las unidades de ejecución a medida que se van decodificando y hay unidades de ejecución disponibles.

Las emisiones de las instrucciones están limitadas por los posibles conflictos de recursos, dependencias de datos y de saltos. Al permitir la finalización desordenada, aparecen nuevas fuentes de conflictos por dependencia de datos.

Emisión ordenada y finalización desordenada – Conflictos: La finalización desordenada requiere detectar estas posibles nuevas dependencias, y evitar que se ejecuten instrucciones en forma desordenada que puedan alterar el resultado del programa. Por tal motivo, la lógica de emisión de instrucciones es más compleja que la empleada en máquinas con política de finalización ordenada. Con la política de emisión ordenada, el procesador solo puede decodificar instrucciones hasta el punto de dependencia o conflicto. Es decir, no analiza si hay nuevas instrucciones que pueden emitirse sin producir conflicto.

Emisión y finalización desordenada: Para permitir que el procesador busque más allá del punto de conflicto por nuevas instrucciones, se requiere usar una política de emisión desordenada. La política de emisión desordenada requiere separar, mediante un buffer, llamado Ventana de instrucciones, las unidades de decodificación y las de ejecución. Cada instrucción decodificada se coloca en un buffer intermedio, desde donde se emiten las que no presentan conflictos. Mientras no se llene el buffer, nuevas instrucciones son decodificadas y colocadas para su emisión a las unidades de ejecución. Las instrucciones emitidas no deben tener conflictos por recursos ni por dependencias de datos.

La Ventana de instrucciones no es una etapa del cauce, es un buffer que retiene la información necesaria para emitirla.

Renombrado de registros:

El hardware del procesador asigna dinámicamente los registros, que están asociados con los valores que necesitan las instrucciones en diversos instantes de tiempo. 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: las referencias a registros de esas instrucciones han de revisarse para referenciar el registro que contiene el valor que se necesita. Las referencias a un mismo registro original en diferentes instrucciones pueden referirse a distintos registros reales.

Ejecución/implementación superescalar:

En resumen, la ejecución superescalar involucra diferentes estrategias de captación simultánea de múltiples instrucciones. Involucra lógica para determinar dependencias verdaderas entre valores de registros y mecanismos para comunicar esos valores. Involucra mecanismos para iniciar o emitir múltiples instrucciones en paralelo, involucra recursos para la ejecución en paralelo de múltiples instrucciones y también involucra mecanismos para entregar el estado del procesador en un orden correcto.

**CLASE 9:**

Categorías de computadoras:

SISD - una secuencia de instrucciones y una secuencia de datos: un único procesador interpreta una única secuencia de instrucciones para operar con los datos almacenados en una única memoria. (Los computadores monoprocesador son ejemplo de esta categoría)

SIMD – una secuencia de instrucciones y múltiples secuencias de datos: una única instrucción maquina controla paso a paso la ejecución simultanea y sincronizada de un cierto numero de elementos de proceso. Cada elemento de proceso tiene una memoria asociada, de forma que cada instrucción es ejecutada por cada procesador con un conjunto de datos diferentes. (Los procesadores vectoriales y los matriciales son ejemplos de esta categoría)

MISD – múltiples secuencias de instrucciones y una secuencia de datos: se transmite una secuencia de datos a un conjunto de procesadores, cada uno de los cuales ejecuta una secuencia de instrucciones diferentes.

MIMD – múltiples secuencias de instrucciones y múltiples secuencias de datos: un conjunto de procesadores ejecuta simultáneamente secuencias de instrucciones diferentes con conjuntos de datos diferentes. (Los SMP, los clusters y los sistemas NUMA son ejemplos de esta categoría).

Multiprocesador simétrico SMP:

Características:

Hay dos o mas procesadores similares de capacidades comparables.

Estos procesadores comparten la memoria principal y las E/S y están interconectados mediante un bus u otro tipo de sistema de interconexión, de forma que el tiempo de acceso a memoria es aproximadamente el mismo para todos los procesadores.

Todos los procesadores comparten los dispositivos de E/S

Todos los procesadores pueden desempeñar las mismas características.

El sistema esta controlado por un sistema operativo integrado que proporciona la interacción entre los procesadores y sus programas a los niveles de trabajo.

Ventajas potenciales de un SMP:

Mayores prestaciones: en general tienen buenos resultados, si las tareas pueden organizarse en paralelo.

Buena disponibilidad: un fallo en un procesador no detiene la operación del sistema, dado que todos los procesadores pueden hacer las mismas tareas.

Crecimiento: pueden aumentarse las prestaciones añadiendo más procesadores, pero hay restricciones a este mecanismo.

Escalado: normalmente limitado, en función de la cantidad de procesadores.

Desventajas potenciales de un SMP: (Bus de tiempo compartido):

Los principales problemas se originan en los conflictos por el bus compartido, la coherencia y consistencia de los datos, y la sincronización de tareas entre procesadores.

Las prestaciones están limitadas por el tiempo de ciclo del bus. Cada procesador está equipado con una memoria cache, que reduce los accesos a memoria y mejora las prestaciones. Pero

eso trae un problema. Al disponer una caché en cada procesador, se pueden producir

problemas de coherencia de cache (datos que pueden estar en más de una caché). Por razones de velocidad, este problema debe ser resuelto por el hardware. Se requieren usar protocolos para la administración de los datos en la caché (protocolos de coherencia tipo sondeo y protocolos de directorio).

Clusters:

Se lo puede definir como un grupo de computadoras completos interconectados que trabajan conjuntamente como un único recurso de cómputo, creandose la ilusión de que se trata de una sola máquina. (El termino computador completo hace referencia a un sistema que puede funcionar por si solo, independiente del cluster).

Beneficios del Cluster:

Escalabilidad absoluta: es posible configurar a los clusters grandes que incluso superan las prestaciones de los computadores independiente mas potentes. Un cluster puede tener decenas de máquinas, cada una de las cuales puede ser un multiprocesador.

Escalabilidad incremental: un cluster se configura de forma que sea posible añadir nuevos sistemas al cluster en ampliaciones sucesivas.

Alta disponibilidad: el fallo de unos de los nodos no significa la perdida del servicio. En muchos casos, es el software el que proporciona automáticamente la tolerancia a fallos.

Mejor relación precio-prestaciones: al utilizar elementos estandarizados, es posible configurar un cluster con mayor o igual potencia de computo que un computador independiente mayor, a mucho menos costo.

Clusters frente a SMP:

AMBOS: dan soporte a aplicaciones de alta demanda de recursos. Disponibles comercialmente.

La principal ventaja de un SMP es que resulta más fácil de gestionar y configurar que un cluster. El SMP esta mucho mas cerca del modelo de computador de un solo procesador para el que están disponibles casi todas las aplicaciones. Otra ventaja del SMP es que necesita menos espacio físico y consume menos energía que un cluster. Y por último es **que los SMP son plataformas estables y bien establecidas.**

Los clusters son superiores a los SMP en términos de escalabilidad absoluta e incremental, y además, también son superiores en términos de disponibilidad, puesto que todos los componentes del sistema pueden hacerse altamente redundantes.

Términos UMA, NUMA, CC-NUMA:

Todos los procesadores tienen acceso a toda la memoria, y se pueden clasificar en:

UMA: tienen igual tiempo de acceso a todas las regiones de memoria e igual tiempo de acceso a memoria para los diferentes procesadores. Un ejemplo de estos es el SMP.

NUMA: el tiempo de acceso de un procesador difiere a diferentes regiones de memoria a diferentes velocidades. Un ejemplo de estos son los Clusters.

SMP tiene limite practico en su numero de procesadores (entre 16 y 64 por degradación de prestaciones). En clusters cada nodo tiene su propia memoria principal, así las aplicaciones no “ven” la memoria global. La coherencia de cache es mantenida por software no por hardware. NUMA retiene las características tipo SMP y brinda multiprocesamiento a gran escala.

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

CC-NUMA: es un NUMA que mantiene coherencia de cache entre las cache de los distintos procesadores. Un ejemplo de estos son los Sistemas con memoria compartida distribuida.

Los sistemas CC-NUMA presentan las siguientes características: Cada nodo tiene 2 o mas procesadores, cierta cantidad de memoria y E/S. Cada procesador tiene su cache. Los nodos están interconectados por algún tipo de red. Cada procesador “ve” un único espacio de direcciones de memoria.

Orden de acceso a memoria:

1. Cache L1 (local al procesador)
2. Cache L2 (local al procesador)
3. Memoria principal (Local al nodo)
4. Memoria remota: petición por red.

La coherencia se mantiene en forma automática y transparente.

Procesamiento Multihebra (multithreading):

Aumento de paralelismo de instrucciones sin el aumento de complejidad y consumo de potencia de la segmentación de cauce y los superescalares.

La secuencia de instrucciones se divide en secuencias mas pequeñas llamadas hebras (threads) que pueden ejecutarse en paralelo.

Tiene una amplia variedad de diseños Multihebra.

Términos: Hebra y proceso:

Proceso: Un proceso es un programa ejecutándose (“corriendo”) en un sistema que:

* Es “propietario” de recursos propios
* Maneja un espacio de direcciones virtuales para almacenar la imagen de un proceso (code, data, stack, etc) con:
  + información propia del estado del proceso
  + Planificación/ejecución por el sistema operativo.
* La comunicación entre procesos es a través de mecanismos específicos.
* La conmutación entre procesos (“Process switch”) requiere un cambio de los recursos asignados (mediante mecanismos precisos y complejos).

Hebra (thread) o hilo:

Es una unidad de trabajo que tiene su propio contexto de procesador (incluido PC y SP) y área de datos para su pila (stack). Comparte con otras hebras, código, variables globales, archivos abiertos por el proceso al que pertenecen, etc. Se ejecuta secuencialmente. Es relativamente fácil de interrumpir, el procesador cambia a otra hebra rápidamente si lo necesita. La conmutación de hebra es un cambio de control del procesador entre hebras de un mismo proceso. Un proceso esta compuesto de muchas hebras o hilos.

Multihebra: Implícito y explicito:

La ejecución de hebras puede ser de 2 tipos:

Multihebra explicito: cuando son definidas en el programa, tanto a nivel de usuario (aplicaciones) como a nivel de sistema operativo (núcleo). (resumen)

Es la ejecución concurrente de instrucciones de diferentes hebras explicitas. Mezcla de instrucciones de diferentes hebras en cauces compartidos o por ejecución paralela de cauces paralelos. (filmina)

Multihebra implícitas: cuando son definidas por el compilador (estático) o por hardware (dinámico). (resumen)

Es la ejecución concurrente de varias hebras extraídas de un único programa secuencial. Están definidas estáticamente por el compilador o dinámicamente por el hardware. (filmina)

Para poder implementar procesamiento Multihebra es necesario, al menos: Un PC distinto para cada hebra que pueda ejecutarse concurrentemente, HW para ejecución concurrente.

Se trata cada hebra separadamente con predicción de saltos, renombre de registros y etc para optimizar la ejecución. Paralelismo entre hebras.

Aproximaciones con ejecución simultanea real.

* Multihebra simultanea (SMT) – Pentium 4
* Multiprocesador monochip