Procesadores Superescalares

Contenido

1. Diseño
2. PowerPC
3. Pentium

Introducción

El término *superescalar* acuñado en 1987, hace referencia a una máquina diseñada para mejorar la velocidad de ejecución de las instrucciones escalares, que es es siguiente paso en la evolución de los procesadores de uso general de altas prestaciones.

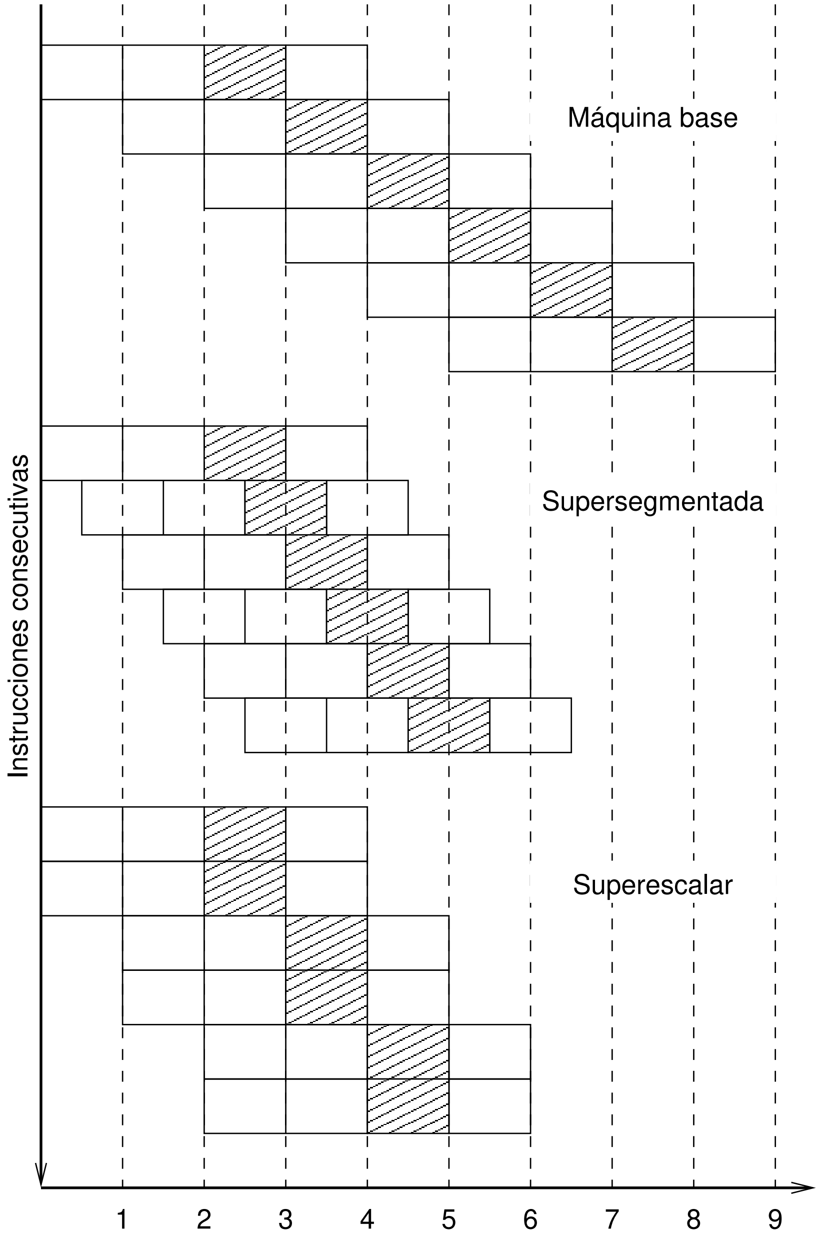
En la implementación superescalar las instrucciones comunes – aritmética entera y en punto flotante, cargas, almacenamientos y bifurcaciones condicionales – pueden iniciar su ejecución simultáneamente y ejecutarse de manera independiente. Lo anterior plantea problemas complejos de diseño relacionados con el cauce de instrucciones.

La arquitectura del RISC se presta para usar técnicas superescalares, pero también se puede hacer en una CISC.

Superescalar Frente a Supersegmentado

Una solución alternativa para alcanzar mayores prestaciones es la supersegmentación, término acuñado en 1988, basada en el hecho de que muchas etapas del cauce realizan tareas que requieren menos de la mitad de un ciclo de reloj, por lo que se dobla la velocidad del reloj interno, para entonces realizar dos tareas en un ciclo de reloj externo.

El procesador supersegmentado toma más tiempo para ejecutar el mismo número de instrucciones respecto al procesador superescalar al comienzo del programa y en cada destino de un salto.



Captación instrucción	Decodificación	Ejecución	Escritura

Comparación de las aproximaciones

Limitaciones

La aproximación superescalar depende de la habilidad para ejecutar múltiples instrucciones en paralelo. El **paralelismo a nivel de instrucciones** se refiere al grado en el que, en promedio, las instrucciones de un programa se pueden ejecutar en paralelo. Para maximizar se puede usar una combinación de optimizaciones realizadas por el compilador y de técnicas hardware. Las limitaciones fundamentales del paralelismo son:

Limitación–Dependencia de datos verdadera

Conocida también como dependencia de flujo o dependencia escritura - lectura. Se presenta cuando hay dependencia de datos entre instrucciones, y se debe retrasar una de las instrucciones tantos ciclos de reloj como sea necesario para eliminar la dependencia, es decir, hasta que los valores de entrada estén disponibles. Limitan las prestaciones de cualquier tipo de cauce. Una forma de compensar el retraso es que el compilador reordene las instrucciones de tal modo que una o más instrucciones posteriores que no dependan de la carga desde memoria puedan empezar a fluir.

Limitación–Dependencia relativa al procedimiento

Se debe a la presencia de bifurcaciones en una secuencia de instrucciones, ya que se puede saltar o no, y no se puede ejecutar hasta la ejecución de la bifurcación. Afecta a un cauce escalar y en un cauce superescalar es más severo. Otra causa pueden ser las instrucciones de longitud variable, ya que no se conoce la longitud.

Otras Limitaciones

- *Conflictos con los recursos*. Se presenta por la competencia de dos o más instrucciones por el mismo recurso (memoria, cache, buses, puertos del fichero de registros y unidades funcionales) al mismo tiempo. Los conflictos pueden superarse duplicando éstos o segmentando la unidad funcional.
- *Dependencia de salida (dependencia lectura-escritura)*.
- *Antidependencia (dependencia lectura-escritura)*.

Diseño

- **Paralelismo a nivel de instrucciones**. Se presenta cuando las instrucciones de una secuencia son independientes y por tanto pueden ejecutarse en paralelo solapándose. Es función de la frecuencia de dependencia de datos verdadera y relativas al procedimiento, los cuales a su vez dependen de la esperaa de una operación (tiempo que transcurre hasta que el resultado está disponible).
- **Paralelismo de la máquina**. Es una medida de la capacidad del procesador para aprovechar el paralelismo a nivel de instrucciones, que depende del número de instrucciones que pueden captarse y ejecutarse al mismo tiempo (número de cauces paralelos) y de la velocidad y sofisticación del mecanismo usado por el procesador para localizar instrucciones independientes.

Políticas de emisión de instrucciones en procs superescalares

El término **emisión de instrucciones** se usa para referirse al proceso de iniciar la ejecución de instrucciones en las unidades funcionales. La **política de emisión de instrucciones** se refiere al protocolo para emitir instrucciones.

El procesador intenta localizar instrucciones que puedan instroducirse en el cauce y ejecutarse. Existen tres ordenamientos importantes para lo anterior:

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

A una mayor sofisticación del procesador, menos limitado estará por los ordenamientos. Para conseguir la máxima utilización el procesador tendrá que alterar uno o más de los ordenamientos, siendo la única restricción que el resultado sea correcto.

Categorías de Políticas de emisión de instrucciones

- *Emisión en orden y finalización en orden*. Es la más sencilla, es equivalente a una ejecución secuencial, no tiene aplicación y se ocupa solamente para comparación.
- *Emisión en orden y finalización desordenada*. Puede haber cualquier número de instrucciones en etapa de ejecución en un momento dado, hasta alcanzar el máximo grado de paralelismo. La emisión se para cuando hay una pugna por un recurso, una dependencia de datos, o una dependencia relativa al procedimiento. La finalización desordenada necesita una lógica de emisión más compleja que la finalización en orden. Es más difícil ocuparse de las interrupciones y las excepciones.

Se tiene una nueva **dependencia de salida** que se presenta cuando se necesita un resultado para una instrucción, pero existen una o más instrucciones que modifican el resultado, por lo que la emisión debe detenerse si el resultado puede ser sobrescrito más tarde por una instrucción anterior que tarda más en finalizar. El siguiente código ilustra esta dependencia, donde I4 debe esperar a que termine I3 para captar el valor correcto.

$$\text{R3} := \text{R3 op R5} \qquad (\text{I1})$$

$$\text{R4} := \text{R3} + 1 \qquad (\text{I2})$$

$$\text{R3} := \text{R5} + 1 \qquad (\text{I3})$$

$$\text{R7} := \text{R3 op R4} \qquad (\text{I4})$$

- *Emisión desordenada y finalización desordenada*. Se desacoplan las etapas del cauce de decodificación y ejecución usando un buffer llamado **ventana de instrucciones**, en donde el procesador coloca las instrucciones decodificadas, mientras el buffer no se llene. Cuando una unidad funcional de la etapa de ejecución queda disponible, se puede emitir una instrucción desde la ventana de instrucciones. Para que una instrucción sea emitida se necesita que la unidad funcional particular este disponible y que ningún conflicto ni dependencia la bloqueen. Con esta organización el procesador tiene una capacidad de anticipación que le permite identificar instrucciones independientes que pueden introducirse en la etapa de ejecución. La **antidependencia** es una nueva dependencia similar a la de la dependencia verdadera, pero a la inversa: en lugar de que la primera instrucción produzca un valor que usa la segunda instrucción, la segunda instrucción destruye un valor que usa la primera instrucción. El fragmento de código ilustra la dependencia citada, donde la instrucción I3 no puede finalizar antes de que la instrucción I2 comience a ejecutarse.

$$\begin{array}{ll} \text{R3} := \text{R3 op R5} & \text{(I1)} \\ \text{R4} := \text{R3} + 1 & \text{(I2)} \\ \text{R3} := \text{R5} + 1 & \text{(I3)} \\ \text{R7} := \text{R3 op R4} & \text{(I4)} \end{array}$$

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

Emision en orden y finalización en orden

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

Emision en orden y finalizacion desordenada

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

Emision desordenada y finalizacion desordenada

Restricciones en el ejemplo de seis instrucciones

- I1 necesita dos ciclos para ejecutarse
- I3 e I4 compiten por la misma unidad funcional.
- I5 depende de un valor producido por I4
- I5 e I6 compiten por una unidad funcional

Políticas de emisión y finalización de instrucciones en un cauce superescalar

Renombramiento de Registros

Para hacer frente a los conflictos de almacenamiento (antidependencias y dependencias de salida) donde varias instrucciones compiten por el uso de los mismos registros, hay un método que se basa en una solución tradicional: la duplicación de recursos, la técnica se conoce como **renombramiento 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 como operando fuente en ese registro tienen que sufrir un proceso de renombramiento, por lo que, las referencias a un mismo registro original en diferentes instrucciones pueden referirse a distintos registros reales.

$$\text{R3}_b := \text{R3}_a \text{ op } \text{R5}_a \qquad \text{(I1)}$$

$$\text{R4}_b := \text{R3}_b + 1 \qquad \text{(I2)}$$

$$\text{R3}_c := \text{R5}_a + 1 \qquad \text{(I3)}$$

$$\text{R7}_b := \text{R3}_c \text{ op } \text{R4}_b \qquad \text{(I4)}$$

Paralelismo

Las tres técnicas de hardware (duplicación de recursos, emisión desordenada y renombramiento) usadas en un procesador superescalar para aumentar las prestaciones fueron estudiadas mediante una simulación aumentando algunas características superescalares, en donde se concluye que probablemente no merece la pena añadir unidades funcionales sin renombramiento de registros, aunque hay algunas mejoras de poca importancia en las prestaciones, a costa de una complejidad mayor del hardware, además si la ventana de instrucciones es demasiado pequeña, las dependencias de datos impiden la utilización efectiva de las unidades funcionales adicionales.

Predicción de Saltos

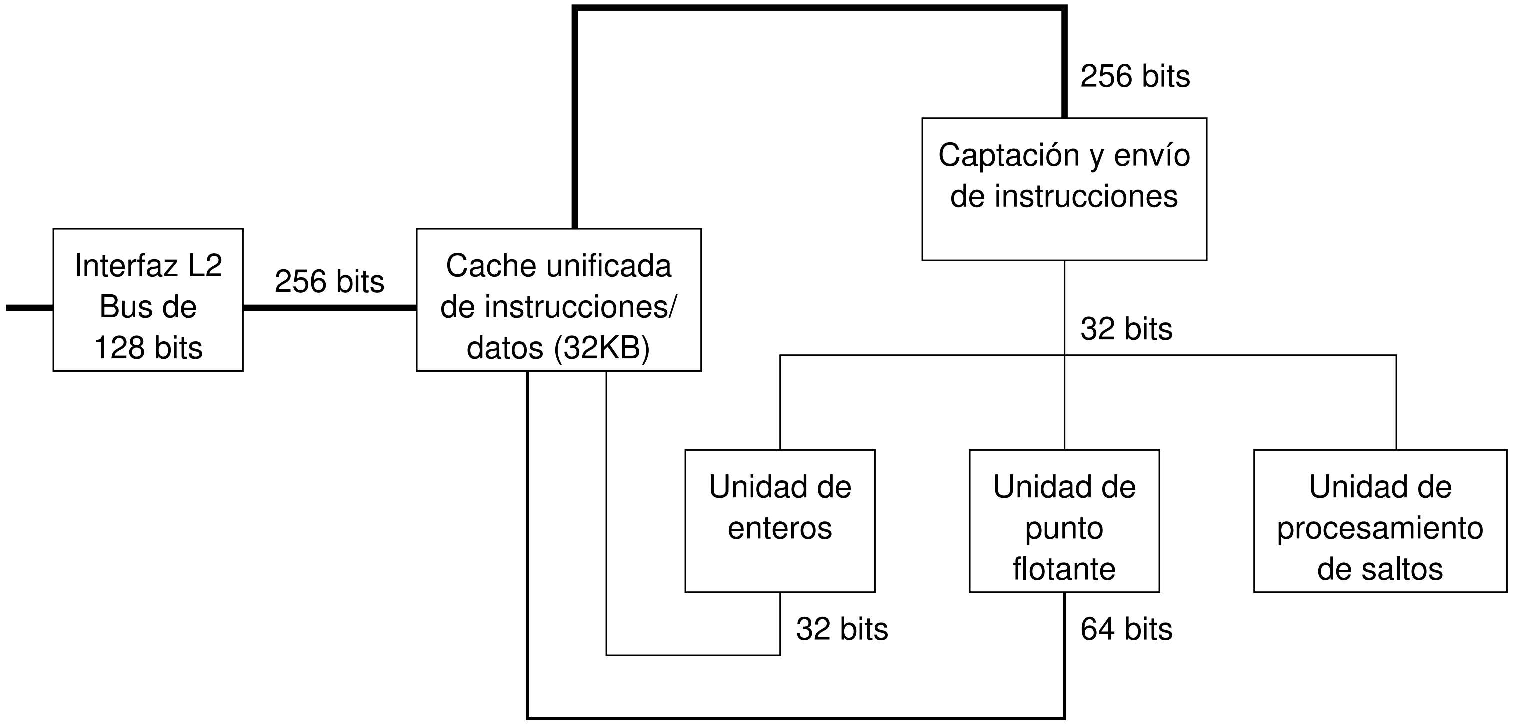
Una máquina segmentada de altas prestaciones debe considerar el tratamiento de las bifurcaciones. El Intel 80486 soluciona el problema captando tanto la siguiente instrucción secuencial a la de bifurcación como la instrucción destino del salto. En las máquinas RISC se uso la estrategia de salto retardado.

Con el desarrollo de las máquinas superescalares, la estrategia de salto retardado ha perdido interés, por la razón de que hay que ejecutar múltiples instrucciones en el espacio de retardo, lo que origina dependencia entre instrucciones, por lo que, las máquinas superescalares han regresado a las técnicas pre-RISC, por ejemplo el PowerPC 601 usa una técnica sencilla de predicción de saltos estática, en cambio el Power620 y el Pentium, usan predicción dinámica de saltos basada en el análisis de la historia de las bifurcaciones.

PowerPC 601

Al igual que otras maquinas superescalares, esta dividido en unidades funcionales independientes segmentadas para enteros, punto flotante y procesamiento de saltos, con lo que se puede realizar la ejecución solapada. La unidad de captación puede precaptar de la cache hasta ocho instrucciones al mismo tiempo. La unidad de cache soporta una cache de datos/instrucciones y suministra instrucciones a las otras unidades y datos a los registros. La lógica de arbitraje de la cache envía a la unidad de captación la dirección del acceso de mayor prioridad.

Diagrama de bloques del PowerPC 601



Unidad de envío de instrucciones

Toma instrucciones de la cache y las carga en una cola de envío, que puede contener ocho instrucciones. La mitad superior de la cola actúa como un buffer de instrucciones de la mitad inferior para asegurar que la unidad de envío no se retrasa esperando instrucciones provenientes de la cache. En la mitad inferior se envían según el siguiente esquema:

- *Unidad de procesamiento de saltos*. Instrucciones de salto si el cauce de la unidad no esta lleno
- *Unidad de punto flotante*. Instrucciones de punto flotante si el cauce de la unidad no esta lleno.
- *Unidad de enteros*. Instrucciones de enteros, las de carga/almacenamiento entre el fichero de registros y la cache, y las instrucciones de comparación de enteros.

Con la posibilidad de emitir instrucciones de salto y de punto flotante desordenadamente se mantienen los cauces de las unidades respectivas llenos. Tiene lógica para calcular la dirección de precaptación, donde se captan instrucciones secuencialmente hasta que una instrucción de salto llega a la mitad inferior de la cola, la cual cuando es procesada, se actualiza la dirección de precaptación para que entren en la cola de envío las instrucciones de la nueva dirección.

Cauces de instrucciones

Hay un ciclo de captación común a todas las instrucciones y tiene lugar antes de que una instrucción se envíe a una unidad concreta. El segundo ciclo comienza con el envío de una instrucción a una unidad particular, la cual se solapa con otras actividades dentro de la unidad. Durante cada ciclo de reloj, la unidad de envío examina los cuatro elementos inferiores de la cola de instrucciones y envía hasta tres instrucciones.

- Para las instrucciones de salto en el segundo ciclo se hace la decodificación y ejecución de instrucciones, además de la predicción de saltos.

- Las instrucciones de carga/almacenamiento de/en memoria (incluyendo punto flotante) son hechas por la unidad de enteros. Hay un ciclo de generación de dirección seguido del envío de la dirección resultante a la cache y, si es necesario un ciclo de escritura.
- La unidad de enteros hace también la transferencia registro-registro y operación de la ALU. Como la cache no está involucrada, hay un ciclo de ejecución seguido de una escritura en un registro.
- Las instrucciones de punto flotante tienen un comportamiento parecido, pero con dos ciclos de ejecución, por la complejidad de las operaciones.

Cauce del PowerPC 601

Instrucciones de salto	Captación	Envio Decodific. Ejecución Predicción				
Instrucciones de enteros	Captación	Envio Decodif.	Ejecución	Escritura		
Instrucciones de carga/almac.	Captación	Envio Decodif.	Generación de dirección	Cache	Escritura	
Instrucciones de punto flotante	Captación	Envio	Decodific.	Ejecución 1	Ejecución 2	Escritura

Procesamiento de Saltos

Al optimizar el cauce en una máquina RISC o superescalar se logran altas prestaciones, el elemento más crítico del diseño es como manejar los saltos. En el PowerPc son manejados por la unidad de saltos, la cual está diseñada para que los saltos no tengan efecto en el ritmo de ejecución de las otras unidades (saltos de cero ciclos) y se emplean las siguientes estrategias:

- Se examina el buffer de envío en busca de saltos. Se generan direcciones destino de salto cuando aparece un salto en la mitad inferior de la cola y no hay saltos anteriores pendientes de ejecución.
- Se intenta determinar el resultado de las bifurcaciones condicionales, si el código de condición se ha ajustado por adelantado lo suficientemente pronto, o la lógica determina si el salto:

- Se producirá (saltos incondicionales o condicionales cuyo código de condición se conoce e indica saltar)
- No se producirá (bifurcación condicional contraria a la anterior).
- No se puede determinar todavía. Se supone que se producirá el salto en las bifurcaciones hacia atrás (bucles).