Cuestiones AC

Jesús Rodríguez Heras

10 de octubre de 2017

Índice

1. Contrastar SISD, SIMD y MIMD

1.1. SISD

Un computador SISD solo tiene un procesador (CPU), el cual unicamente tiene una unidad de control (UC) y una unidad de procesamiento (ALU).

1.2. SIMD

Un computador SIMD solo tiene un procesador (CPU), el cual tiene una unidad de control (UC) y varias unidades de procesamiento (ALU).

SIMD:

Arquitectura escalar (SISD)	Arquitectura vectorial (SIMD)
No tiene registros vectoriales (solo tiene registros escalares (o de propósito general))	Sí tiene registros vectoriales (también tiene registros escalares)

No tienen instrucciones vectoriales
Ejemplo: Y=a·X+Y // X e Y son vectores de 64 números en puto flotante doble precisión.
`l.d $f0,a($sp) :load scalar a`
`addiu $t0,$s0,#512 :upper bound of what to load`
`loop: l.d $f2,0($s0) :load x(i)`
`mul.d $f2,$f2,$f0 :a x x(i)`
`l.d $f4,0($s1) :load y(i)`
`add.d $f4,$f4,$f2 :a x x(i) + y(i)`
`s.d $f4,0($s1) :store into y(i)`
`addiu $s0,$s0,#8 :increment index to x`
`addiu $s1,$s1,#8 :increment index to y`
`subu $t1,$t0,$s0 :compute bound`
`bne $t1,$zero,loop :check if done`
Se ejecutarían cerca de 600 instrucciones
	Sí tienen instrucciones vectoriales
	Ejemplo: Y=a·X+Y // X e Y son vectores de 64 números en puto flotante doble precisión.
	`l.d $f0,a($sp) :load scalar a`
	`lv $v1,0($s0) :load vector x`
	`mulvs.d $v2,$v1,$f0 :vector-scalar multiply`
	`lv $v3,0($s1) :load vector y`
	`addv.d $v4,$v2,$v3 :add y to product`
	`sv $v4,0($s1) :store the result`
	Se ejecutan 6 instrucciones

No hay paralelismo a nivel de datos	Sí hay paralelismo a nivel de datos - Extensiones SIMD (ALU particionada o subword) - Procesadores vectoriales (unidad funcional pipelined (lane))

# C code	# Scalar Code	# Vector Code

`for (i=0; i<64; i++)`
`C[i] = A[i] + B[i];`
	`LI R4, 64`
	`loop:`
	`L.D F0, 0(R1)`
	`L.D F2, 0(R2)`
	`ADD.D F4, F2, F0`
	`S.D F4, 0(R3)`
	`DADDIU R1, 8`
	`DADDIU R2, 8`
	`DADDIU R3, 8`
	`DSUBIU R4, 1`
	`BNEZ R4, loop`
		`LI VLR, 64`
		`LV V1, R1`
		`LV V2, R2`
		`ADDV.D V3, V1, V2`
		`SV V3, R3`

Las extensiones SIMD implementan instrucciones para el manejo de múltiples datos y datos vectoriales o de coma flotante. Se implementa mediante una ALU particionada, lo que reduce el número de las operaciones que deben ser soportadas por el sistema y elimina los indicadores de excepciones.
La clasificación SIMD está compuesta de un grupo determinado de procesadores SISD, estos reciben una misma instrucción pero diferentes datos con los que operar, así se consigue un paralelismo a nivel de datos que permite realizar un único proceso. Esto es lo que ocurre en los procesadores multinúcleo, donde cada núcleo se comporta como un procesador.

EXTENSIONES SIMD

Año/computador	Ext. SIMD	N° inst. añad.	N° reg. Vect.	P. Subword		
				N° op	Bit dato	Bit reg.
1996 Pentium MMX	MMX	57	8 MM0 – MM7	8	8	64
				4	16	64
				2	32	64
1999 Pentium III	SSE	70	8 XMM0 – XMM7	4	32	128
2001 Pentium IV	SSE2	144	8 XMM0 – XMM7	16	8	128
				8	16	128
				4	32	128
				2	64	128
2003 Opteron	AMD64	Base	16 XMM0 – XMM15	4	32	128
				2	64	128
2004 Pentium IV Prescott	EM64T	Base	16 XMM0 – XMM15	4	32	128 (1)
				2	64	128 (2)
2011 Sandy Bridge	AVX	128	16 YMM0 – YMM15	8	64	256 (1)
				4	64	256 (2)

(1). VADDPS: Suma vectorial de simple precisión.

(2). VADDPD: Suma vectorial de doble precisión.

1.3. MIMD

Un computador MIMD tiene varios procesadores (CPU) y cada uno de ellos tiene su unidad de control (UC) y su unidad de procesamiento (ALU).
Dentro de los computadores MIMD podemos diferenciar dos tipos:

1.3.1. Memoria compartida (multiprocesadores)

Existe un único espacio de direcciones que es compartido por todos los procesadores y, normalmente, está formado por una sola computadora.
Un ejemplo de ello son los procesadores multinúcleo.

1.3.2. Memoria distribuida (multicomputadores)

Cada procesador tiene su propio espacio de direcciones. Consta de un conjunto de computadoras conectadas entre sí por una red de comunicaciones que es percibido por el usuario como un solo sistema.
Un ejemplo de ello son los clusters y los grid.

2. Arquitectura escalar vs Arquitectura vectorial

ARQUITECTURA ESCALAR	ARQUITECTURA VECTORIAL
- No tiene registros vectoriales, sólo tiene registros escalares (de propósito general) - No tiene instrucciones vectoriales (ADD, LW, SW, BNE, ADDI⋯) - No hay paralelismo a nivel de datos (No se puede operar con varios vectores a la vez)	- Tiene registros vectoriales (guarda varios elementos de un array) y también tiene registros escalares - Tiene instrucciones vectoriales (ADD, LW, SW, BNE, ADDI⋯) - Hay paralelismo a nivel de datos (varios vectores operan a la vez) - El paralelismo es implementado por las extensiones SIMD (ALU particionada o paralelismo subword) y por los procesadores vectoriales (pipeline)

3. Arquitectura vectorial: Procesador con extensiones SIMD vs Procesador vectorial

La arquitectura vectorial es un esquema de microprocesadores cuya función está orientada a las operaciones con vectores.

3.1. Procesador con extensiones SIMD

Un procesador con extensiones SIMD es un procesador escalar capaz de realizar las funciones de un procesador vectorial gracias a las extensiones SIMD (SSE, etc).

3.2. Procesador vectorial

Consta de una unidad escalar segmentada y una unidad vectorial. La unidad vectorial dispone de M registros vectoriales de N elementos y de unidades funcionales vectoriales que trabajan sobre los registros vectoriales, y un conjunto de registros escalares.

4. Paralelismo subword

El paralelismo subword es la posibilidad de realizar varias operaciones simultaneas con datos de tamaño menor a una palabra.

5. Caché directa

En la caché directa, un bloque procedente de memoria principal solo puede situarse en una posición determinada de la memoria caché. Esta posición está determinada por la dirección que tiene ese bloque en memoria principal. Las posiciones de la memoria caché se identifican con un índice. La etiqueta contiene la parte de la dirección que no aparece en el índice, es decir, la parte más significativa de la dirección.

5.1. Funcionamiento

1.) El procesador está ejecutando la instrucción LW $t0, 1101: cargar en el registro $t0 el dato que está en la dirección de memoria 1101. Por tanto, el procesador tiene que acudir al sistema de memoria para obtener el dato.
2.) Comprueba si el dato está en el primer nivel de la jerarquía de memoria, es decir, en caché.
3.) Va al índice que coincide con la parte menos significativa de la dirección 1101 presente en la instrucción LW.
4.) Verifica si la etiqueta de ese índice coincide con la parte más significativa de la dirección 1101. Como coincide, se puede afirmar que el dato está en caché (ACIERTO). En caso de no coincidir, se puede afirmar que el dato no está en caché (FALLO) y habría que ir a memoria principal a buscarlo.

6. Caché asociativa por conjuntos

La caché está dividida en conjuntos de bloques (vías).
Cada conjunto se identifica por un índice.
Un bloque de memoria principal solo puede ser situado en un conjunto concreto de la memoria caché asociativa por conjuntos, pero dentro de ese conjunto puede emplazarse en cualquier bloque (método aleatorio) o en el bloque que lleva más tiempo sin utilizarse (método LRU).

6.1. Funcionamiento

1.) El procesador está ejecutando la instrucción LW $t0, 1101: cargar en el registro $t0 el dato que está en la dirección de memoria 1101. Comprueba si el dato está en caché.
2.) Va al índice y ve si coincide con la parte menos significativa de la dirección 1101 presente en la instrucción LW.
3.) Mira todas las etiquetas del conjunto en paralelo para verificar si hay alguna que coincida con la parte más significativa de la dirección 1101. Como hay una etiqueta que coincide, la petición de memoria acierta en caché. En caso contrario, se produce un fallo y hay que ir a memoria principal a buscar el dato.

7. Contrastar cachés L1, L2 y L3

CACHÉ L1	CACHÉ L2 Y L3
- Está situada en medio de los componentes principales de la CPU - Cuando la CPU requiere un dato del sistema de memoria siempre lo lee de la L1 - Su objetivo es hacer que la CPU trabaje con un ciclo de reloj lo más corto posible (minimizar el tiempo de acceso) - Para minimizar el tiempo de acceso a la cache L1, se decrementa el grado de asociatividad y se decrementa el tamaño de la caché.	- También residen en la CPU, pero se sitúan entre la L1 y la memoria principal. - Si se produce un fallo en la L1 se busca el dato en la L2 para almacenarlo en la L1 y luego la CPU lo lee de la L1. - Si el dato no estuviese en L1 ni en L2 se buscaría en L3 y si tampoco está en L3 pues se buscaría finalmente en la memoria principal. - Su objetivo es disminuir los fallos en caché debido a los largos tiempos que se requiere acceder a la memoria principal. - Para disminuir la frecuencia de fallos en L2 y L3 se incrementa la asociatividad y se incrementa el tamaño de la caché.

La caché L2 puede estar compartida entre todos los cores o ser exclusiva de cada core.

8. ILP (paralelismo a nivel de instrucción)

Consiste en que la combinación de instrucciones de un mismo proceso que ejecuta un procesador puedan ser ordenadas de forma tal que, al ser procesadas a la vez (paralelamente), no afecten al resultado final del programa. La principal finalidad de dicho paralelismo es aumentar la velocidad y aprovechar al máximo las capacidades hardware del ordenador.

8.1. Procesador segmentado (pipelined)

La técnica pipelined es una técnica en la que se separan por etapas el camino de las instrucciones para que instrucciones diferentes puedan alojarse en etapas diferentes.

8.2. Procesador supersegmentado (profundidad del pipeline incrementada)

Es una técnica en la cual se segmentan las etapas de ejecución de una instrucción para que más instrucciones puedan ser ejecutadas consecutivamente.

8.3. Procesador con ejecución múltiple

Podemos diferenciar dos tipos:

8.3.1. Procesador superescalar (planificación dinámica)

Consiste en el duplicado de todos los componentes de una CPU para ejecutar dos instrucciones simultáneamente sin que estas visiten las mismas zonas del DIE.

8.3.2. Procesador VLIW (planificación estática)

Se basa en las arquitecturas superescalares ya que ambas usan varias unidades funcionales o replican dichas unidades funcionales. Se caracteriza por tener pocas instrucciones pero su tamaño es muy grande (Very Long Instrucction Word). En cada instrucción se especifica el estado de todas las unidades funcionales del sistema, con el objetivo de simplificar el diseño del hardware al dejar todo el trabajo de planificar el código en manos del programador/compilador, en oposición a un procesador superescalar, en el que es el hardware en tiempo de ejecución el que planifica las instrucciones. VLIW=IA-64.
Cualquier mejora de la arquitectura implica un cambio en el juego de instrucciones (retrocompatibilidad nula).
No está orientado al uso de propósito general.

9. ILP vs PLP

ILP (paralelismo a nivel de instrucciones)	**PLP (paralelismo a nivel de procesos)**
- Ejecutar varias instrucciones a la vez - Todas las instrucciones pertenecen a un mismo proceso - Técnicas que implementan el ILP **Procesador segmentado(pipelined):** Una instrucción diferente por etapa, que se ejecutan a la vez. **Procesador supersegmentado:** Se incrementa la profundidad del pipelined, mayor nº de etapas. **Ejecución múltiple:** -VLIW -Superescalar	- Se ejecutan varios procesos o subprocesos a la vez. - Permite el paralelismo al tener 2 o más núcleos - En cada ciclo se ejecutan varias instrucciones a la vez en distintos procesos - Si se tiene un núcleo sólo se puede lograr PLP si se hace Hyperthreading

10. Hyperthreading

Es una técnica cuyo objetivo es conseguir PLP en un solo núcleo del procesador. Se simulan dos núcleos virtuales dentro de un núcleo físico.
Tenemos tres tipos:

10.1. Grano fino

En cada instrucción se cambia de hilo (ejecución entrelazada de varios hilos). En la ejecución se salta los hilos parados en ese momento. Se cambia de hilo en cada ciclo de reloj.
Ventaja: Cuando un hilo se para se pueden seguir ejecutando instrucciones de otros hilos por tanto se evitan que hayan muchas paradas largas o cortas.
Desventaja: Se pierde velocidad en la ejecución de un hilo individual porque un hilo que está listo para ejecutarse sin paradas se verá interrumpido por instrucciones de otros hilos.

10.2. Grano grueso

El entrelazado de hilos sólo se realiza cuando hay una parada larga (Ejemplo: fallo de caché L2).
Ventaja: No se pierde velocidad en la ejecución de un hilo individual porque solamente se verá interrumpido el hilo por instrucciones de otros hilos si se produce una parada larga, en parada cortas no.
Desventaja: Se producen perdidas de prestaciones al no haber entrelazado en paradas cortas, cuando se produce una parada el pipelined se vacía y el nuevo hilo que se va a ejecutar debe volver a llenarlo antes de ser capaz de terminar las instrucciones, debido a este sobrecoste de llenado esta ejecución es más útil donde el tiempo de llenado del pipelined es muy pequeño en comparación al de parada.

10.3. SMT

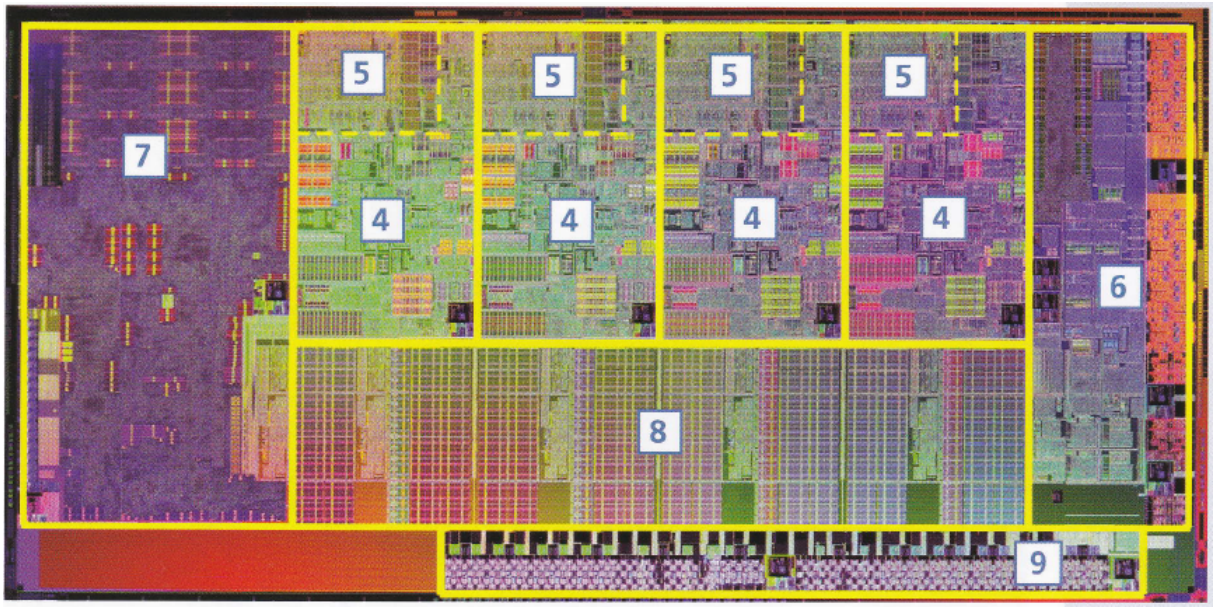
Su objetivo es explotar el paralelismo a nivel de hilos y a nivel de instrucciones a la vez.
No se conmutan sus hilos en cada ciclo, sino que están siempre ejecutando varias instrucciones de varios hilos. (El hardware determina las instrucciones a ejecutar y los registros asociados a cada hilo).
Las dependencias entre instrucciones se resuelven gracias a la planificación dinámica. Resuelve el problema de los huecos en blanco por culpa de las dependencias.

11. Contrastar ILP, PLP e Hyperthreading

ILP	PLP	Hyperthreading
Ejecuta varias instrucciones a la vez	Ejecuta varios procesos a la vez	
Un solo núcleo	Varios núcleos	Un núcleo
Segmentación de la CPU	Multinúcleo	Núcleos lógicos o virtuales

12. Elementos principales de un DIE

Es el chip que está en el procesador y que contiene toda su circuitería:



12.1. Cachés L1 [4], L2 [5] y L3 [6]

Caché L1 está integrada cerca de la zona de los núcleos, cada núcleo tiene su propia caché L1, al igual que la caché L2 que está cerca y es una por núcleo. La caché L3 sólo es una y se encuentra más alejada de los núcleos, es la caché más grande y lenta, es compartida por todos los núcleos del procesador.

12.2. Núcleos [4]

Están justo en el centro del chip y tienen la unidad de control (UC) y la unidad aritmético-lógica (ALU).

12.3. GPU (gráfica) [7]

La gráfica se encuentra integrada en el DIE también lo cual reduce el coste del PC y beneficia la comunicación entre el procesador gráfico y el procesador principal.

12.4. Socket

Cuenta con miles de contactos para dar alimentación y realizar las comunicaciones con el resto del ordenador.

12.5. Puente Norte [8]

Es un chip cuya función principal es la de controlar el funcionamiento del bus del procesador, la memoria y el puerto AGP o PCI-Express. De esa forma, entre la placa madre y los principales componentes de la PC.

12.6. Interfaz de la memoria RAM [9]

Bus que conecta los zócalos de la memoria RAM con el controlador de memoria interno.

13. Tick-Tock y familias de procesadores

Es un modelo adoptado por Intel en el que se asocia cada "tick" con una mejora de la tecnología que representa una evolución en dicha tecnología (tamaño del procesador en nanómetros) y cada "tock" representa una mejora de la arquitectura (juegos de instrucciones).

Año	Gen	Nombre	Tam CPU	Evolución	Mejoras
2008	1ª	Nehalem	45nm	Tock	Incorpora en el DIE: controlador de memoria, PCI-Express x16 (tarjetas gráficas), QPI (QuickPath: bus rápido para comunicar varios procesadores entre sí tal como en los servidores o placas bases con varios procesadores diferentes), DMI SSE4.2: extensión SIMD
2010		Westmere	32nm	Tick	Se implementan algunas mejoras
2011	2ª	Sandy bridge	32nm	Tock	AVX: extensión SIMD Incorpora en el DIE: GPU
2012	3ª	Ivy bridge	22nm	Tick	Se implementan algunas mejoras
2013	4ª	Haswell	22nm	Tock	AVX2(extensiones SIMD), YMM (registros vectoriales)
2014	5ª	Broadwell	14nm	Tick	Se implementan algunas mejoras
2015	6ª	Skylake	14nm	Tock	MPX: Extensiones para prevenir los ataques de stack overflow (desbordamiento de pila)
2016	7ª	Kabylake	14nm	Tock+	Skylake+ (optimización)
2017	8ª	Cannonlake	10nm	Tick	Se implementan algunas mejoras

14. RISC vs CISC

RISC (mayor rendimiento de la máquina)	CISC (mayor facilidad al programar y compilar)
ARM MIPS (instrucciones 32bits)	x86=x86-32=IA-32 x86-64
Número reducido de instrucciones	Amplio número de instrucciones
Instrucciones de tamaño fijo	Instrucciones de tamaño variable lo que complica la decodificación de la instrucción(front-end)
Programas más largos (por tener pocos tipos diferentes de instrucciones para usar)	Programas más cortos (podemos usar muchos tipos de instrucciones lo cual nos facilita el trabajo, ejemplo DIV)
Arquitectura más simple	Arquitectura más compleja
Sólo acceden a memoria las instrucciones LW/SW (carga y almacenamiento)	Pueden acceder a memoria "cualquiera" de las instrucciones. Ejemplo: En ADD los operandos y el resultado pueden estar en memoria de esta forma nos ahorramos instrucciones previas como LW/SW
Facilidad para implementar el pipelined(segmentación del procesador)	Dificulta la segmentación del procesador
Todas las instrucciones duran el mismo tiempo al tener el mismo tamaño fijo	Instrucciones duran tiempos distintitos al ser cada una de un tamaño
Acceso al registro completo	Acceso a diferentes partes del registro
Labor del programador y de los compiladores más compleja	Labor del programador y de los compiladores más sencilla