

Las principales magnitudes

Sumario

3.1. Frecuencia de reloj	49
3.2. Tecnología de integración	50
3.2.1. Evolución y significado	50
3.2.2. Efectos directos sobre otras variables	54
3.2.3. Efectos laterales entre las variables afectadas	56
3.2.4. Cómo dar empleo a un ejército de transistores	58
3.3. Paralelismo a nivel de instrucción	58
3.3.1. Segmentación (pipelining)	59
3.3.2. Superescalaridad	60
3.3.3. Combinación de segmentación y superescalaridad	62
3.3.4. Supersegmentación	63
3.3.5. Dependencias: Las enemigas del paralelismo	64
3.3.5.1. Ejecución fuera de orden	67
3.3.5.2. Predicción de salto	68
3.4. Memoria caché integrada	69
3.4.1. Breve sinopsis histórica	72
3.4.2. Jerarquía	73
3.4.3. Optimizaciones	77
3.4.3.1. Buses desacoplados	77
3.4.3.2. Caché no bloqueante	77
3.4.3.3. Caché segmentada	78
3.4.3.4. Caché con lectura anticipada	79
3.4.3.5. Caché víctima	79
3.4.3.6. Caché de tercer nivel (L3)	80
3.4.4. Proximidad al núcleo del procesador	81
3.4.5. Ubicación del controlador de caché	85
3.4.6. Velocidad	88
3.4.7. Análisis del rendimiento de caché en relación al procesador	88
3.4.8. Análisis del coste asociado a una caché	93
3.4.8.1. Caché interna	93
3.4.8.2. Caché integrada	94
3.5. Conjunto de instrucciones	94
3.5.1. CISC versus RISC	95
3.5.2. Diseño RISC	97

3.5.2.1. Selección del conjunto de instrucciones	98
3.5.2.2. Soporte software para una arquitectura RISC	100
3.5.3. Diseños VLIW	104
3.5.4. Instrucciones multimedia	106
3.5.4.1. El concepto: SIMD	108
3.5.4.2. El embrión: MMX	109
3.5.4.3. Criterios para la selección de instrucciones	110
3.5.4.4. Compatibilidad	112
3.5.4.5. Ampliaciones al conjunto MMX	114
3.5.4.6. Otras extensiones multimedia	118
Resumen	119
La anécdota: Magnitudes oficiales y oficiosas	121
Cuestionario de evaluación	122

definición

El microprocesador es el cerebro del computador y el centro neurálgico de sus actividades. Se trata de un circuito integrado o chip cuya función consiste en interpretar y ejecutar instrucciones máquina, para lo cual se divide en dos grandes unidades funcionales: La Unidad de Proceso, que ejecuta las instrucciones siguiendo una secuencia de pasos, y la Unidad de Control, encargada de interpretarlas y generar las señales eléctricas que controlan dicha secuencia. El conjunto se denomina también CPU (del inglés, *Central Processing Unit - Unidad Central de Proceso*).

U. Proceso

U. Control

CPU

conjunto de instrucciones

Las instrucciones que entiende un microprocesador conforman su repertorio o conjunto de instrucciones máquina, el alfabeto con el que se articulan todos los programas que en él se ejecutan. Estos programas se encuentran almacenados en la memoria del equipo, junto con los datos con los que operan. El microprocesador accede a la memoria a través de los buses del sistema, al igual que al resto de componentes del equipo, incluidos los periféricos. Todos estos componentes se conectan o vienen ya integrados en un soporte rígido poco mayor que las dimensiones de un folio de papel, que recibe el nombre de placa base. El sistema informático se completa con los periféricos, como los discos, las distintas tarjetas, la impresora, el ratón y el teclado.

sistema

informático

secuencia

de estudio

La memoria, los buses y la placa base serán analizados, por este orden, más adelante, pero antes de eso, comenzaremos nuestro estudio con el microprocesador. En el presente capítulo, conoceremos las principales variables que inciden en su funcionamiento y rendimiento, y cómo éstas se encuentran ligadas entre sí. Los capítulos posteriores nos descubrirán las características de todos sus modelos, clasificados por generaciones.

El microprocesador es un sistema extremadamente complejo, por lo que una de las primeras cosas que debemos aprender en su estudio es acotar los parámetros que realmente influyen en su rendimiento y saberlos distinguir de aquellos que son meros elementos decorativos y/o reclamos publicitarios. Sus cinco magnitudes más importantes son las siguientes:

cinco

magnitudes

- ❶ Frecuencia de reloj.
- ❷ Tecnología de integración.
- ❸ Paralelismo a nivel de instrucción.
- ❹ Memoria caché integrada.
- ❺ Conjunto de instrucciones.

secuencia e

interrelación

El orden elegido para su tratamiento no ha sido caprichoso. Comenzamos por las variables de bajo nivel, más ligadas a su constitución eléctrica, proseguiremos en un nivel intermedio con las que se encuentran más asociadas a su diseño, y finalizaremos a más alto nivel, con la frontera de diálogo con el software. La [figura 3.1](#) muestra la secuencia que seguiremos en nuestro tratamiento, y la interrelación entre las distintas magnitudes atendiendo a criterios de vecindad dentro de la pirámide que adjuntamos.

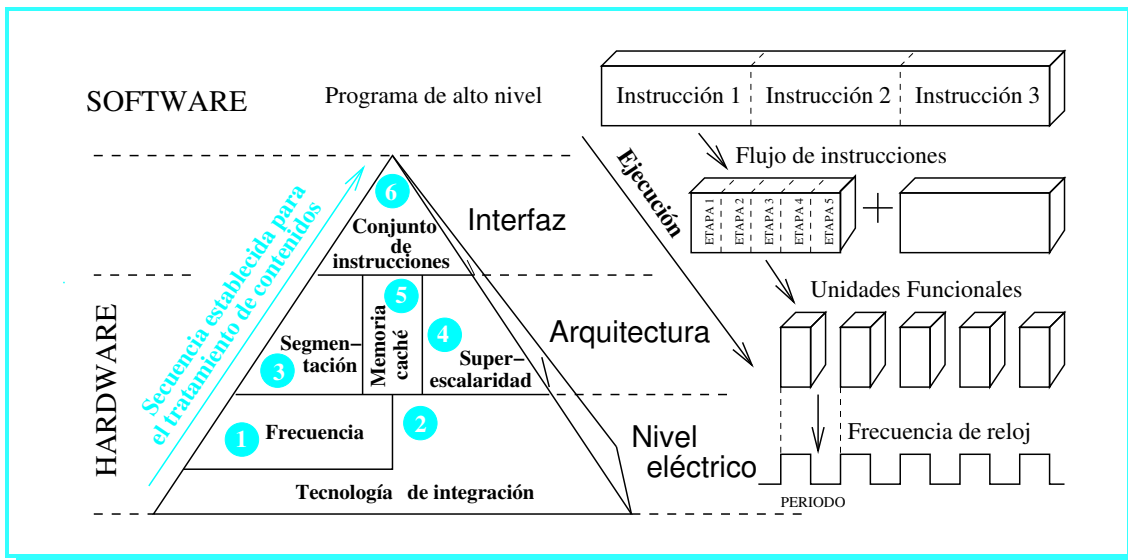


FIGURA 3.1: La secuencia en el tratamiento de las principales magnitudes del procesador y su interrelación y dependencia, atendiendo a donde se encuentra cada una apoyada en nuestra pirámide. Fuera de ella, podemos también observar la relación con la vertiente software del procesador.

SECCIÓN 3.1

Frecuencia de reloj

La frecuencia de reloj es un factor cuantitativo que indica la velocidad del microprocesador. Tiene su origen en un cristal de cuarzo, que ante la aplicación de un voltaje comienza a vibrar (oscilar) a una frecuencia armónica determinada por la forma y el tamaño del cristal.

origen

Las oscilaciones emanan en forma de una corriente que sigue la función senoidal correspondiente a su frecuencia armónica, y que una vez filtrada en un circuito PLL se convertirá en la secuencia de pulsos digitales, cuadrados, periódicos y síncronos cuya cadencia marcará el ritmo de trabajo de los distintos chips del computador. Para mayor información sobre la distribución de esta señal por toda la placa base del PC, consultar la [sección 17.2](#).

forma

➔ [p.8/Vol.3](#)

El oscilador suele integrarse ya dentro del juego de chips de la placa base, por lo que cada vez es menos frecuente advertir su presencia en la circuitería mediante una exploración visual. No obstante, siguen existiendo muchos modelos de tarjetas y placas base en las que se monta como un elemento aparte, en forma de una pequeña pastilla de estaño que reviste su delgada lámina de cuarzo. Las [fotos 17.3](#), [30.1](#) y [30.2](#) muestran el aspecto de varios de estos osciladores y su circuito PLL asociado.

oscilador

➔ [p.9/Vol.3](#)

➔ [Vol.5 en Web](#)

La magnitud inversa de la frecuencia es el **período de reloj**. De esta manera, si la frecuencia se expresa en Megahercios, o millones de pulsos por segundo, el período lo hará en microsegundos, y si la primera lo hace en Gigahercios, el segundo se regirá en nanosegundos. La [figura 3.2](#) muestra el aspecto de esta señal en contraste con otra de tipo analógico. Para tener conciencia de las magnitudes que manejamos, diremos que un reloj de 1 GHz realiza dos millones de ciclos en el mismo tiempo que dura un leve parpadeo de nuestros ojos.

período de reloj

magnitudes

➔ [pág. 50](#)

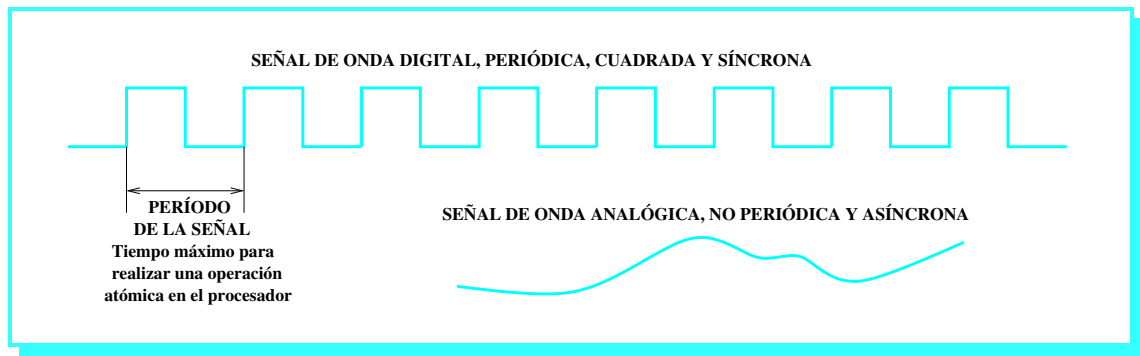


FIGURA 3.2: Aspecto de la señal de reloj que sincroniza la velocidad de un microprocesador y comparativa con otra señal de naturaleza analógica.

Las instrucciones máquina que acepta el microprocesador se descomponen en una sucesión de operaciones atómicas o etapas, cada una de ellas consumiendo un período o ciclo de reloj, tal y como hemos ilustrado en la [figura 3.1](#). Así, cuanto mayor sea la frecuencia del chip, menor será el tiempo que tardará en ejecutar las instrucciones de un programa, con el consiguiente aumento en el rendimiento de nuestro PC.

Mucha gente resume la potencia de un microprocesador en el valor de su frecuencia de trabajo, y los fabricantes y distribuidores de hardware siguen aprovechando esta práctica instaurada entre sus clientes para tratar de encandilarnos con ella. Realizar tal simplificación en un sistema de la sofisticación de un procesador con decenas de millones de transistores es una temeridad. Ni siquiera con el estudio de las cinco magnitudes que aquí proponemos puede uno pretender conocerlo, aunque sí creemos que puede forjarse ya una impresión válida del conjunto.

Remitimos al lector al [capítulo 29](#), dedicado en exclusiva a la frecuencia del procesador, para profundizar en su caracterización física y la incidencia que tiene sobre el resto de variables eléctricas del chip. Si quiere aprender más sobre la farsa que este parámetro representa desde hace ya un tiempo, le recomendamos la lectura del [capítulo 30](#). En los próximos capítulos encontrará ejemplos esclarecedores del peso relativo que tiene la frecuencia en el rendimiento de un microprocesador, siendo el Pentium 4 (ver [sección 6.4](#)) un buen exponente de lo que decimos.

SECCIÓN 3.2

Tecnología de integración

definición La tecnología de integración es un indicador más cualitativo que cuantitativo. En general, puede definirse como *la mínima resolución de la maquinaria responsable de integrar los circuitos mediante técnicas de litografía*.

importancia Estamos frente a la magnitud de la que más dependen las otras cuatro, ya que disminuir esta resolución supone reducir el coste por cada chip integrado y su voltaje de alimentación, y aumentar la frecuencia y el número de transistores disponibles. Profundizaremos más sobre todas estas implicaciones, pero después de acercarnos a su significado intrínseco.

2.1 ► Evolución y significado

transistor MOS

Dentro del microprocesador existen millones de minúsculos conmutadores, los **transistores**, cuyo funcionamiento comporta dos estados que son interpretados como '0' o '1' para componer el sistema lógico binario con el que se procesa toda la información dentro del computador. En su

pág. 49
descomposición

estereotipos

Vol.5 en Web

Vol.5 en Web
información
adicional
pág. 208

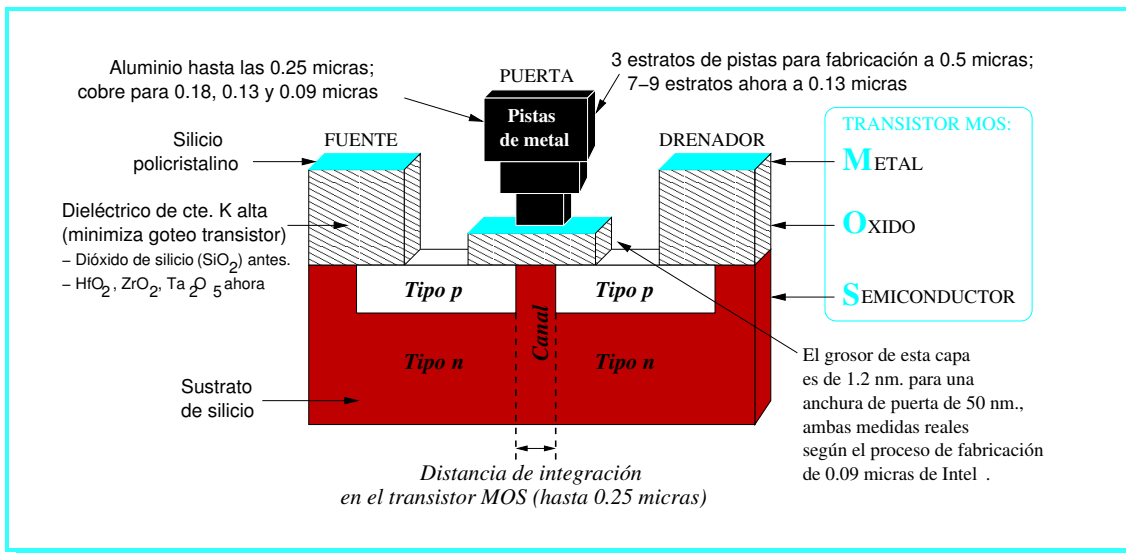


FIGURA 3.3: Un típico transistor MOS, donde indicamos los materiales utilizados para su fabricación y su distancia de integración. El silicio dopado con impurezas actúa como semiconductor cuya dualidad conductor/aislante permite la codificación de un bit al nivel más interno.

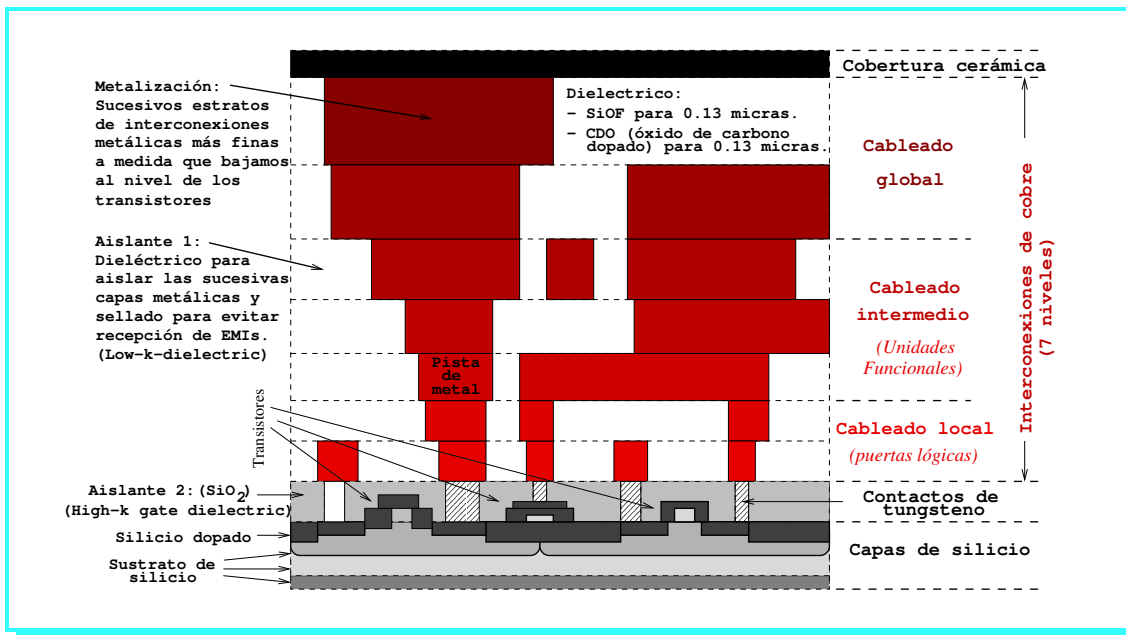


FIGURA 3.4: Sección transversal de un chip, donde se aprecian los niveles de interconexión de sus transistores: Entre 6 u 8 estratos de pistas metálicas de aluminio o cobre, responsables de definir la funcionalidad del chip.

versión MOS (*Metal Oxide Semiconductor*), los transistores se fabrican con un material semiconductor, y se conectan entre sí por medio de un metal (ver figura 3.3).

La integración de estos dos componentes en el área del chip tiene lugar mediante sofisticadas técnicas de encapsulado de materiales, donde los transistores, que no son apilables, se disponen en una estructura bidimensional que ocupa la capa más inferior, y se interconectan mediante un enrejado compuesto por entre seis y ocho capas de aluminio o cobre como metal situado por

	Tecnología de integración (micras)									
	1.00	0.80	0.50	0.35	0.25	0.18	0.13	0.09	0.065	0.045
Anchura (micras) para la puerta del transistor	1.00	0.80	0.50	0.35	0.20	0.13	0.07	0.05	0.035	0.025
Area (micra ²) para la celda de caché (6 transistores)	220	111	44	21	10.6	5.6	2.09	1.00	?	?
Voltaje de alimentación (voltios)	5.0	5.0	3.3	2.5	1.8	1.5	1.3	1.2	?	?
Maquinaria y año de fabricación	P648 1989	P650 1991	P852 1993	P854 1995	P856 1997	P858 1999	P860 2001	P1262 2003	P1264 2005	P1266 2007

TABLA 3.1: Evolución de la tecnología de integración y su relación con la anchura de la puerta de sus transistores y otras variables eléctricas en las plantas de fabricación de Intel. La maquinaria de fabricación se renueva aproximadamente cada dos años.

encima del silicio (ver [figura 3.4](#) - para más información puede consultarse el [capítulo 34](#)).

Vol. 5 en Web

CMOS

La tecnología de integración más ampliamente utilizada durante el proceso de fabricación es la CMOS (Complementary Metal Oxide Semiconductor), responsable del 75 % del volumen total de chips manufacturados a escala mundial según la Semiconductor Industry Association.

bipolar

De todos los microprocesadores que analizaremos, tan sólo el Pentium y el Pentium Pro no utilizan íntegramente esta tecnología: Son encapsulados CMOS en su parte más interna, pero la zona perimetral donde se sitúa su patillaje (287 y 386 pines resp.), están integrados con tecnología bipolar. Esta alternativa se utilizaba para las zonas del chip que necesitaran una mayor intensidad de corriente, pero ha entrado en claro desuso debido a la fuerte disipación de potencia que esto conlleva y los problemas de temperatura que padecen los microprocesadores actuales.

distancia de integración

pág. 51

pág. 55

El parámetro clave de una tecnología de integración concreta es la **distancia de integración**. En tecnología CMOS, por ejemplo, esta distancia coincidió durante las dos últimas décadas con la anchura del canal con que se fabrican sus transistores de silicio (ver [figura 3.3](#)), aunque dicha coincidencia dejó de cumplirse con la llegada de las 0.25 micras (los datos de las [tablas 3.1](#) y [3.3](#) son bastante clarificadores a este respecto); en otras tecnologías de fabricación de transistores, la distancia de integración se asocia con la anchura de la pista de metal que une los transistores.

de la micra al nanómetro

Tradicionalmente, la distancia de integración ha venido midiéndose en **micras**, diminutivo del *micrómetro*, que representa la millonésima parte del metro. Sin embargo, la evolución tan espectacular que ha seguido la miniaturización de los transistores ha dejado grande a esta escala, y cada vez es más usual emplear el nanómetro (abreviado nm. - mil millonésima parte del metro). Por ejemplo, los últimos modelos de K7 y Pentium 4 se fabrican a 0.13 micras o 130 nm., y la próxima mejora en este sentido nos lleva en 2004/2005 a las 0.09 micras o 90 nm.

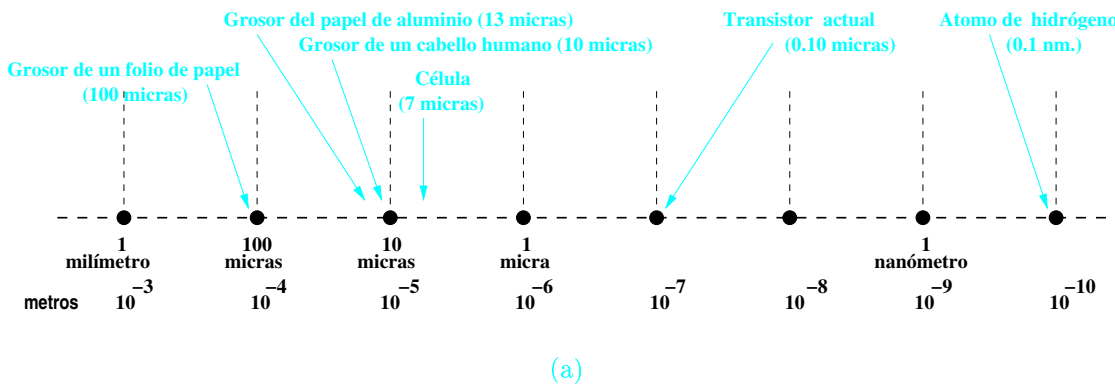
escala

pág. 53

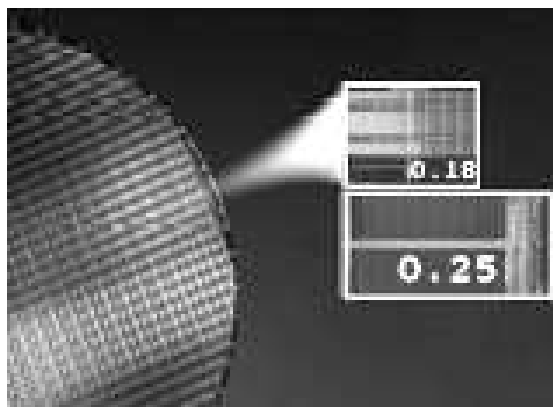
Para tener una referencia visual de lo que todo esto significa, diremos que tendríamos que apilar la anchura de más de 750 transistores de 0.13 micras para alcanzar el grosor de una hoja de este libro. La [foto 3.1](#) muestra una escala comparativa donde ubicamos una serie de elementos de dimensiones microscópicas.

nomenclatura

Puesto que la tecnología de integración va a ser siempre CMOS (salvo en las dos excepciones obsoletas ya reseñadas), tenderemos a omitirla a partir de ahora, y para simplificar las cosas englobaremos también en este término a la distancia de integración en micras. Es decir, diremos que un microprocesador se fabricó, por ejemplo, con *tecnología de integración de 0.13 micras* cuando formalmente tendríamos que haber dicho que se fabricó con *tecnología de integración CMOS a una distancia de integración de 0.25 micras para las puertas de sus transistores*.



(b)



(c)

Fotos cortesía de Intel

FOTO 3.1: Microscopía de la distancia de integración. (a) Escala de distancias (logarítmica) que permite relativizar el tamaño de un transistor actual respecto a otros elementos. Si la distancia de 0.25 micras fuese este libro abierto, las dimensiones del átomo de hidrógeno en que acaba la escala se corresponderían con el punto en que acaba esta frase. (b) Un grano de sal fotografiado junto a la orografía de un circuito integrado. (c) Comparativa espacial (horizontal) entre las 0.18 micras (arriba) y las 0.25 micras (abajo).

De forma más reciente en la que esa correspondencia ya no existe (por ejemplo, para 130 nm., la anchura de la puerta del transistor es de apenas 70 nm. - ver [tabla 3.1](#)), estaríamos ya obligados a utilizar la definición más general que hace referencia a la resolución mínima de la maquinaria de litografía con la que se fabricó el microprocesador, pero seguiremos utilizando la jerga a la que estamos acostumbrados aún reconociendo que científicamente no es lo más purista.

La tecnología de integración no avanza de forma continua, ya que una vez instaurado un proceso de fabricación debe transcurrir un tiempo hasta la amortización de sus plantas de fabricación. Ciertamente es que una empresa como Intel o AMD produce cientos de millones de chips cada año, pero el coste de sus plantas de fabricación es también enorme.

plantas de
fabricación

Los plazos establecidos por la industria del chip para la renovación de sus plantas de fabricación están en torno a los dos años, tal y como se refleja en la [tabla 3.2](#), donde resumimos los valores utilizados por Intel y AMD. Ese período ha demostrado ser lo suficientemente extenso como para amortizar el enorme coste de la planta, y al mismo tiempo, lo suficientemente efímero como para mantener a estas empresas a la vanguardia del tren de la alta tecnología.

renovación
cada 2 años
pág. 54

Distancia de integración (micras)	Año de implantación	Modelo más representativo	
		Intel	AMD
1	1989	80486	80486
0.8	1991	80486DX	-
0.8 (0.5, 0.35)	1993	Pentium	-
0.6 (0.35)	1994	Pentium Pro	-
0.35 (0.25)	1995	Pentium MMX	K5
0.35 (0.25)	1997	Pentium II	K6
0.25 (0.18, 0.13)	1999	Pentium III	K7
0.18 (0.13)	2001	Pentium 4	Athlon XP
0.13	2003	Pentium 4	K8

TABLA 3.2: La evolución en la distancia de integración ha transcurrido siempre a pasos discretos. En esta tabla recogemos los valores que ha tomado esta variable en los últimos diez años. Los números entre paréntesis corresponden a evoluciones posteriores.



Ejemplo 3.1: LA AMORTIZACIÓN DE LAS PLANTAS DE FABRICACIÓN DE CHIPS

La planta de fabricación de microprocesadores más avanzada instalada en suelo europeo se encuentra emplazada en Dresden (Alemania), y pertenece a la empresa AMD.

Inició su actividad en Noviembre de 1998, fabricando desde Enero de 1999 los K6 de 0.25 micras y aluminio, desde Enero de 2000, los K7 de 0.18 micras y cobre, y desde Noviembre de 2002 los Athlon XP de 0.13 micras y 9 niveles de metal. Para mediados de 2003 fabricará los futuros K8, y ya en 2004, integrará éstos a 0.09 micras.

Su ritmo de producción es de más de un millón de chips por semana (5000 obleas de 20 centímetros de diámetro cada una, para ser exactos), y si dicho ritmo disminuye sólo un 10% durante una jornada laboral de 8 horas, las pérdidas para AMD sobrepasan los 10 millones de euros.

A esa velocidad de manufacturación, lo primero que uno piensa es que la amortización de costes es casi inmediata. No tanto: La inversión inicial superó los 2.000 millones de euros.

ASML Por cierto, que la empresa líder en suministrar a los fabricantes la maquinaria de litografía más puntera con la que hacer realidad sus chips es precisamente europea: La holandesa ASML.

2.2 ▶ Efectos directos sobre otras variables

El potencial que esconde una reducción de la distancia de integración es tan grande que consigue hasta cuatro efectos benignos sobre las variables físicas ligadas a la constitución interna de un microprocesador:

número de transistores

- 1 Consigue aumentar de forma cuadrática el **número de transistores** que se pueden integrar en un mismo espacio físico. Es decir, una reducción desde 1 micra hasta las 0.5 micras para un mismo diseño permitiría albergar cuatro veces más transistores, y una reducción hasta las 0.25 micras, dieciséis veces más. Esto hace crecer de forma considerable el patrimonio de que dispone el diseñador del microprocesador para aumentar sus prestaciones: incorporando funcionalidad adicional, aumentando el tamaño de las cachés integradas, y un sinfín de posibilidades más que iremos desvelando un poco más adelante.

	Tecnología de integración (micras)				
	0.13	0.10	0.07	0.05	0.035
Año de su puesta en marcha	2002	2005	2008	2011	2014
Anchura de puerta del transistor	0.085	0.065	0.045	0.030	0.020

Fuente: International Technology Roadmap for Semiconductors. Edición 1999

TABLA 3.3: Estimaciones para los sucesivos valores de la tecnología de integración y la anchura de puerta del transistor, junto a su puesta en funcionamiento en un marco temporal de quince años en adelante. Las predicciones han sido realizadas por la Semiconductor Industry Association. A finales de 2002, IBM y AMD firmaron un acuerdo para la fabricación de chips que confirma la llegada de las 0.065 micras para 2005 y las 0.045 incluso para 2007 en una nueva planta de fabricación con obleas de 30 cm. de diámetro a construir en Singapur.

Generación y modelo de microprocesador	Distancia de integración en micras							
	1	0.8	0.6	0.5	0.35	0.25	0.18	0.13
Cuarta - 80486	33-100MHz							
Quinta - Pentium	66MHz — 233MHz							
Sexta - Pentium Pro	133 — 200MHz							
Quinta - Pentium MMX					133-233MHz			
Sexta - K6					166-333MHz			
Sexta - Pentium II					233-450MHz			
Sexta - K6-2					266-500MHz			
Sexta - Pentium III					0.45 — 1.3 GHz			
Séptima - K7					0.5 — 2.5 GHz			
Séptima - Pentium 4					1.4 — 4 GHz			

TABLA 3.4: La reducción de la distancia de integración en las sucesivas generaciones de microprocesadores es una de las claves sobre las que se sustenta la consecución de frecuencias más elevadas.

- ② Aumenta la velocidad de operación del transistor, y con ello, la **frecuencia** del chip. La agilidad para conmutar entre los estados lógicos 0 y 1 viene dada por el tiempo que debe transcurrir para que la corriente que provoca ese cambio fluya entre la fuente y el drenador del transistor (ver [figura 3.3](#)). Como la distancia de integración es precisamente la que separa estas dos zonas, cuanto menor sea ésta, menor será el tiempo de paso y mayor la velocidad de conmutación. La física del transistor nos dice en este sentido que una reducción de su distancia de integración conlleva un aumento en su frecuencia de similares proporciones, es decir, que como cada nuevo proceso de fabricación contempla una reducción de distancia de factor 0.7x respecto a su predecesor, la frecuencia aumentará en un factor 1.5x.
- ③ Disminuye el **voltaje de alimentación** que requiere el chip. La principal beneficiaria de esta disminución es la menor **potencia disipada** en forma de calor, lo que repercutirá en un descenso de la **temperatura** del chip. En el [capítulo 29](#) descubrimos que la temperatura es uno de los parámetros que más limitan la frecuencia de reloj de un microprocesador, por lo que una reducción del voltaje complementa la consecución de altas frecuencias.
- ④ Permite disminuir el **coste de fabricación** del microprocesador, ya que al acortarse las distancias entre transistores, se reduce el área de silicio que se necesita para la integración de un diseño dado un número de transistores fijo. En definitiva, aumenta la densidad de integración y el número de chips que caben en cada oblea de silicio. Como la distancia de integración es una magnitud lineal y el precio del chip se determina en función del coste por oblea (área circular, y por tanto, bidimensional), resulta que una reducción en la distancia de integración abarata el coste de forma cuadrática.

frecuencia

→ [pág. 51](#)

voltaje
potencia
disipada
temperatura

→ [Vol.5 en Web](#)

coste de
fabricación

 **Ejemplo 3.2:** CUANTIFICANDO EL EFECTO DIRECTO DE LA DISTANCIA DE INTEGRACIÓN SOBRE OTRAS VARIABLES

Supongamos que los últimos Pentium (P55C) y primeros Pentium II (Klamath), todos de 0.35 micras, hubiesen sido fabricados con las distancias de integración de 0.18 micras de los últimos Pentium III (Coppermine) y primeros Pentium 4 (Willamette).

En ese caso, el Pentium 200 MHz hubiese sido un Pentium 400 MHz, y el Pentium II 300 MHz hubiese sido un Pentium II 600 MHz. Asimismo, el Pentium, de 3.1 millones de transistores habría podido tener 12.4 millones, y el Pentium II, de 7.5 millones, hasta 30 millones. Bastantes más de los que realmente dispone el Coppermine, y en un área de silicio muy similar. Con estos transistores hubiera sido posible: (a) Dotar al Pentium de una caché L2 de 128 Kbytes sincronizada a su misma velocidad utilizando la tecnología de aquella época, pues se necesitaban entonces unos 8 millones de transistores para ello, y (b) Incorporar al Pentium II una caché L2 interna de 384 Kbytes a su misma velocidad, algo que ni siquiera tuvo a su alcance el Pentium Pro en sus inicios con un coste superior a los 1.200 €. (El ejemplo cuantifica sólo la incidencia directa entre variables, porque dado que en la práctica existen multitud de efectos laterales, sería extraordinariamente difícil precisar un valor real exacto).

2.3 ▶ Efectos laterales entre las variables afectadas

Desgraciadamente, las cosas no son tan sencillas como acaban de ser expuestas. Ya avisamos que estamos ante un sistema extraordinariamente complejo en el que se producen multitud de efectos laterales, y al menos debemos hacer referencia a los más importantes:

- 1 Si contrastamos la evolución de la distancia de integración con la de la frecuencia, vemos que si los diseños de una micra rondaban los 33 MHz y los de 0.13 micras se encuentran en torno a los 3 GHz, en el mismo espacio temporal en el que un parámetro se ha reducido en un factor de 7.5, el otro ha aumentado en un factor de 100. Esto representa 13 veces más de lo esperado según la incidencia lineal de uno sobre otro que acabamos de postular, pero lo que ha ocurrido aquí es que otros aspectos que también inciden positivamente sobre la frecuencia han sido responsables del rango de mejora restante. Estos otros aspectos se encuentran documentados en el [capítulo 29](#).

velocidad

Vol.5 en Web 

 **Ejemplo 3.3:** CUANTIFICANDO LA MEJORA QUE LA DISTANCIA DE INTEGRACIÓN REVIERTE SOBRE LA FRECUENCIA TENIENDO EN CUENTA EFECTOS LATERALES

Si un viejo Pentium Pro se fabricara ahora utilizando distancias de integración de 0.18 micras, alcanzaría su techo de frecuencia en torno a los 1.2 GHz. Esto significa tres veces más del valor esperado según el [ejemplo 3.2](#), lo que evidentemente pone de manifiesto la incidencia de los efectos laterales comentados.

Parámetro eléctrico	Efecto de una reducción de la distancia de integración	Efecto lateral sobre los demás			
		Núm.	Frec.	Coste	Volt.
Núm transistores	Aumento cuadrático	=	Baja	Sube	Sube
Frecuencia	Aumento lineal	Baja	=	=	Sube
Coste	Reducción cuadrática	Baja	=	=	=
Voltaje	Reducción lineal	Baja	Baja	=	=

TABLA 3.5: Efectos directos que produce una reducción de la distancia de integración y la repercusión que éstos tienen a su vez en forma de efectos laterales sobre los demás parámetros eléctricos de un chip.

- ② Tampoco es correcto considerar que el precio del chip se reduzca de forma cuadrática, pues hemos cuantificado su coste en función de la materia prima utilizada, pero no en la dificultad de integración y testeo, que obviamente son tareas más complejas al realizarse sobre más transistores y más diminutos, lo que exige mayores inversiones en infraestructura de litografía.

coste

Además, no todos los transistores de un chip son iguales en tamaño. Los de la caché son muy pequeños en silicio pero muy complejos en sus capas de metalización (interconexiones), mientras que los que contiene una ALU ocupan un área de integración mayor. A efectos prácticos, y de forma implícita, estamos considerando un tamaño medio común a todas las unidades funcionales con objeto de poder establecer una proporcionalidad entre el número de transistores utilizados y el área de integración del chip.

espacio

- ③ El voltaje y la frecuencia distan mucho de ser independientes entre sí. Si subimos la frecuencia para aprovechar las mejoras que nos brinda la tecnología, ésta tirará a su vez para arriba del voltaje, contrarrestando los niveles de tensión inferiores que la tecnología ponía a nuestro alcance. El resultado puede ser que incluso se haga necesario aumentar el voltaje a distancias de integración más pequeñas.

voltaje

- ④ El número de transistores y la frecuencia tampoco son independientes, porque si nos decidimos a aprovechar el mayor número de éstos, a buen seguro que aumentaremos la cantidad de unidades funcionales del chip y el número de elementos que debe atravesar su **camino crítico** (aquel que atraviesan las señales eléctricas cuando el microprocesador ejecuta su operación atómica más lenta), y la longitud de éste condiciona fuertemente la máxima frecuencia de funcionamiento del conjunto. El resultado contrasta de nuevo con las cuatro premisas establecidas en el apartado anterior, ya que una reducción de la distancia de integración ha beneficiado a la funcionalidad del diseño, pero ha perjudicado a la frecuencia.

camino crítico

En consecuencia, podemos concluir que no existe de antemano una estrategia ganadora en la construcción de un microprocesador. Y el mercado es, una vez más, quien nos proporciona la lección más soberana: En él coexisten diseños que priman descaradamente la frecuencia de reloj (como el Alpha 21264 de Digital que en 1993 superó los 500 MHz), frente a otros orientados claramente a aumentar el número de unidades funcionales (como la familia del Power PC de Motorola, que para ese mismo año disponía de diseños con un factor superescalar de seis). Y entre ambos extremos tenemos toda una gama de soluciones intermedias que ponderan de diferente manera uno y otro aspecto. Un buen ejemplo que recorre todo este territorio intermedio sería la familia de los Pentium, que basculó progresivamente desde el lado de las unidades funcionales con el Pentium Pro hacia el lado de la frecuencia con los Pentium II, III y 4 ¹.

Alpha

Power PC

Pentium

¹ Escribiremos los ordinales de los microprocesadores en números romanos o arábigos según se haya contemplado en la marca registrada de su fabricante.

2.4 ► Cómo dar empleo a un ejército de transistores

Llegado este punto, conocemos que las sucesivas reducciones en la distancia de integración traen consigo la posibilidad de aumentar la frecuencia o el número de transistores del diseño, pero no ambas, dados los conflictos apuntados como efectos laterales.

bajo nivel La opción de aumentar la frecuencia exige ciertos conocimientos sobre el funcionamiento interno del transistor a bajo nivel, de ahí que hayamos decidido abordarla más adelante dentro del contexto microelectrónico que predomina en el volumen 5, disponible en nuestra Web. En concreto, en el [capítulo 29](#) contamos los fundamentos teóricos, mientras que en el [capítulo 30](#) pasamos a la acción para su manipulación.

nivel arquitectural Ahora, lo que toca es centrarse en el nivel arquitectural del chip, o lo que es lo mismo, conocer qué infraestructura habilitar para aprovechar el creciente número de transistores que la tecnología de integración va a ir poniendo a nuestro servicio de forma sucesiva. Mostraremos cómo esa circuitería adicional puede colaborar en el aumento del rendimiento de un microprocesador a pesar de que la frecuencia del chip pueda verse en ocasiones resentida. De esta manera, destruiremos ese falso mito que muchos profanos de la informática tienen en la cabeza: “el mejor procesador es aquel que funciona a más MHz”.

Las vías que se han utilizado para estas mejoras del rendimiento giran en torno a tres ideas principales que ocuparán los tres tramos siguientes de este capítulo:

paralelismo ❶ El paralelismo a nivel de instrucción ([sección 3.3](#)), o la facultad de poder procesar varias instrucciones de forma simultánea, todas ellas procedentes de un único programa en ejecución.

caché ❷ La incorporación de memorias cachés como un elemento más del diagrama de bloques del microprocesador, o la virtud para hacerle llegar grandes volúmenes de datos con la presteza que los necesita ([sección 3.4](#)).
pág. 69 ➡

instrucciones ❸ Las ampliaciones y/o modificaciones del conjunto de instrucciones máquina, o la potestad para especializar al microprocesador en las nuevas operaciones multimedia que van requiriendo los programas, al tiempo que se logra también optimizar el procesamiento de las más tradicionales ([sección 3.5](#)).
pág. 94 ➡

SECCIÓN 3.3

Paralelismo a nivel de instrucción

1a idea En su descomposición funcional más sencilla, un microprocesador se compone de una Unidad de Control y una Unidad de Proceso. En esta última convivían inicialmente la ALU y el banco de registros en solitario, pero enseguida se les unieron nuevas unidades funcionales en la búsqueda de un aumento del rendimiento. La mayoría de estos aditivos llevan a la práctica alguna forma de **paralelismo a nivel de instrucción**, idea que de forma genérica consiste en romper con la ejecución secuencial de instrucciones (una detrás de otra en el tiempo) para simultanear su ejecución (varias a la vez). Distintas formas de explotar este paralelismo son la segmentación, la superescalabilidad y la supersegmentación.

Segmentación (pipelining)

3.1

Un procesador **segmentado** es aquel que divide el proceso de ejecución de una instrucción en N etapas de similar duración, con el objetivo último de procesar N instrucciones simultáneamente, encontrándose cada una en una etapa diferente de su ejecución.

definición

Conseguimos así que N unidades funcionales del procesador estén trabajando a la vez, lo que redundará en un factor N de mejora en el rendimiento del chip en circunstancias ideales.

rendimiento

Las instrucciones fluyen secuencialmente por las distintas unidades funcionales del procesador de igual forma que el agua fluye por el cauce de una tubería, de ahí que en la jerga de la calle se les conozca también como procesadores *pipeline*². A nosotros nos ha parecido más elegante emplear el término *cauce segmentado* o simplemente *cauce*, denominación que seguiremos a partir de ahora.

etimología



Ejemplo 3.4: EL CAUCE DE SEGMENTACIÓN MÁS CLÁSICO

El modelo de segmentación más repetido en la primera mitad de los años 90 es el compuesto por las siguientes cinco etapas: Búsqueda de la instrucción, decodificación, lectura de operandos, ejecución de la operación asociada y escritura de su resultado.

De esta manera, mientras el procesador escribe una instrucción, ejecuta simultáneamente la siguiente, busca los operandos de una tercera, decodifica una cuarta y busca de memoria una quinta instrucción, lográndose en el caso ideal una aceleración de cinco para el rendimiento del chip.

Tanto el MIPS, el procesador modelo de esta técnica desarrollado en Stanford, como el Pentium de Intel, presentan una división en cinco etapas muy similar a la descrita.

Todos los microprocesadores actuales se encuentran segmentados, habiendo aumentado el número de etapas con el paso de las generaciones. Así, lo normal en séptima generación es encontrarnos con cauces de ejecución entera compuestos de hasta 20 etapas de segmentación, como ocurre, por ejemplo, en el procesador que lidera este rasgo en la arquitectura PC de 2003, el Pentium 4 de Intel.

Junto al cauce de ejecución entero se sitúan otros cauces por donde circulan los otros tipos de instrucciones, principalmente multimedia y de punto flotante. Como hasta que no concluye la fase de decodificación de instrucción no se puede realizar la pertinente ramificación, las primeras etapas de segmentación son siempre comunes a todos los cauces del procesador.

variedad de cauces

El empleo tan superlativo del concepto de segmentación es algo que no sorprende desde el momento en que se conoce su gran activo subyacente: El aumento del rendimiento se consigue con sólo reorganizar las unidades funcionales existentes, es decir, no supone incremento de coste para la Unidad de Proceso. El diseño de la Unidad de Control sí se complica un poco a medida que aumenta el número de etapas y el grado de concurrencia en la ejecución de instrucciones, pero en cualquier caso, con carácter marginal frente al espectacular incremento logrado en el lado del rendimiento.

coste

²El término anglosajón que significa precisamente *tubería*.

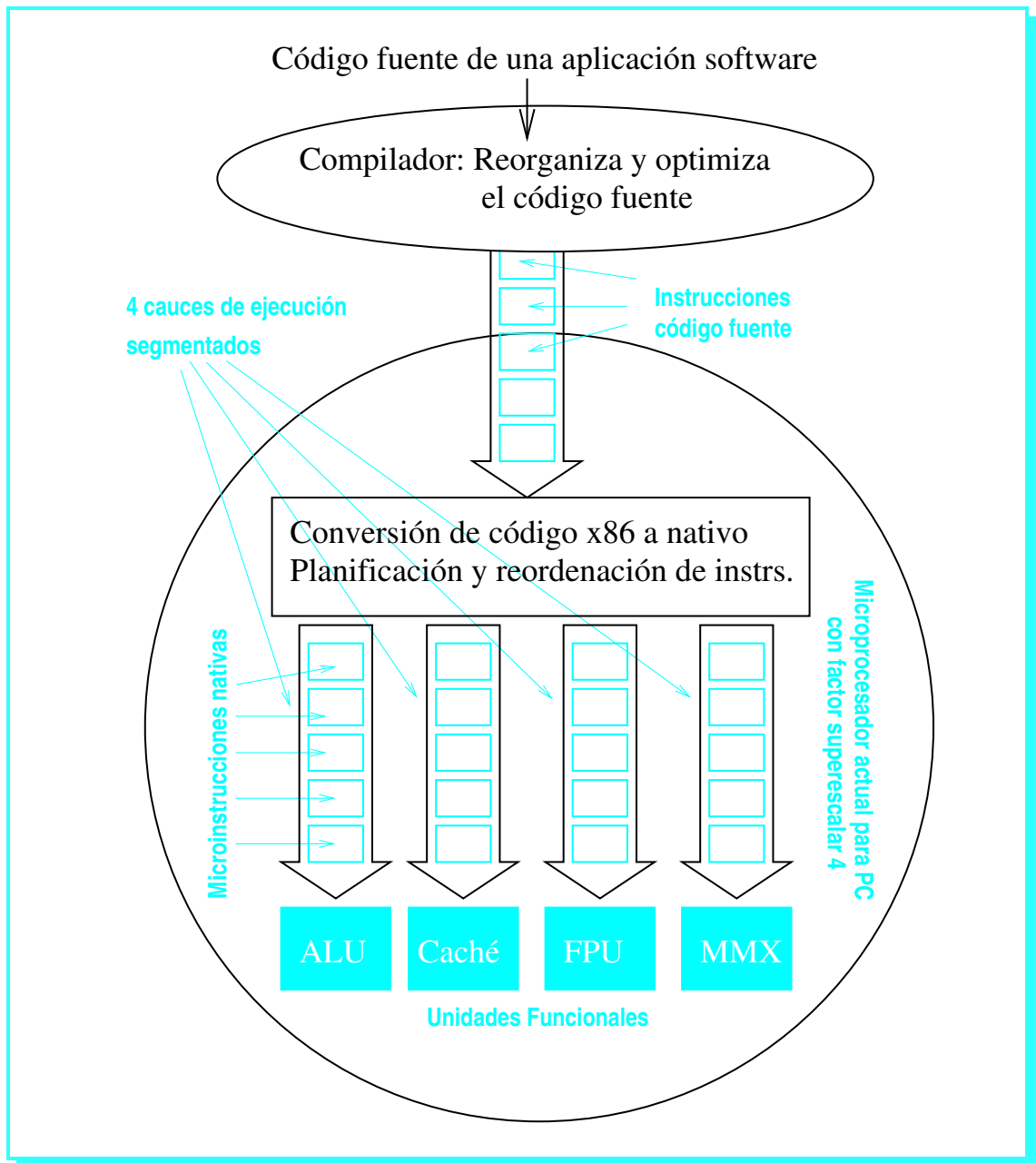


FIGURA 3.5: La forma tradicional de ejecutar instrucciones en un microprocesador de quinta y sexta generación utiliza los conceptos de segmentación y superescalaridad combinados.

3.2 ▶ Superescalaridad

concepto

Un procesador **superescalar** de factor N es aquel que replica N veces la circuitería de alguna de sus unidades funcionales con el fin de poder ejecutar N instrucciones en sus respectivas etapas de computación.

rendimiento

Al igual que la segmentación, a medida que las mejoras en la integración de circuitos han permitido concentrar más y más componentes dentro de un único chip, los diseñadores de microprocesadores han aumentado el factor de superescalaridad, logrando un incremento del rendimiento que al igual que en la segmentación alcanza el factor N en circunstancias ideales.

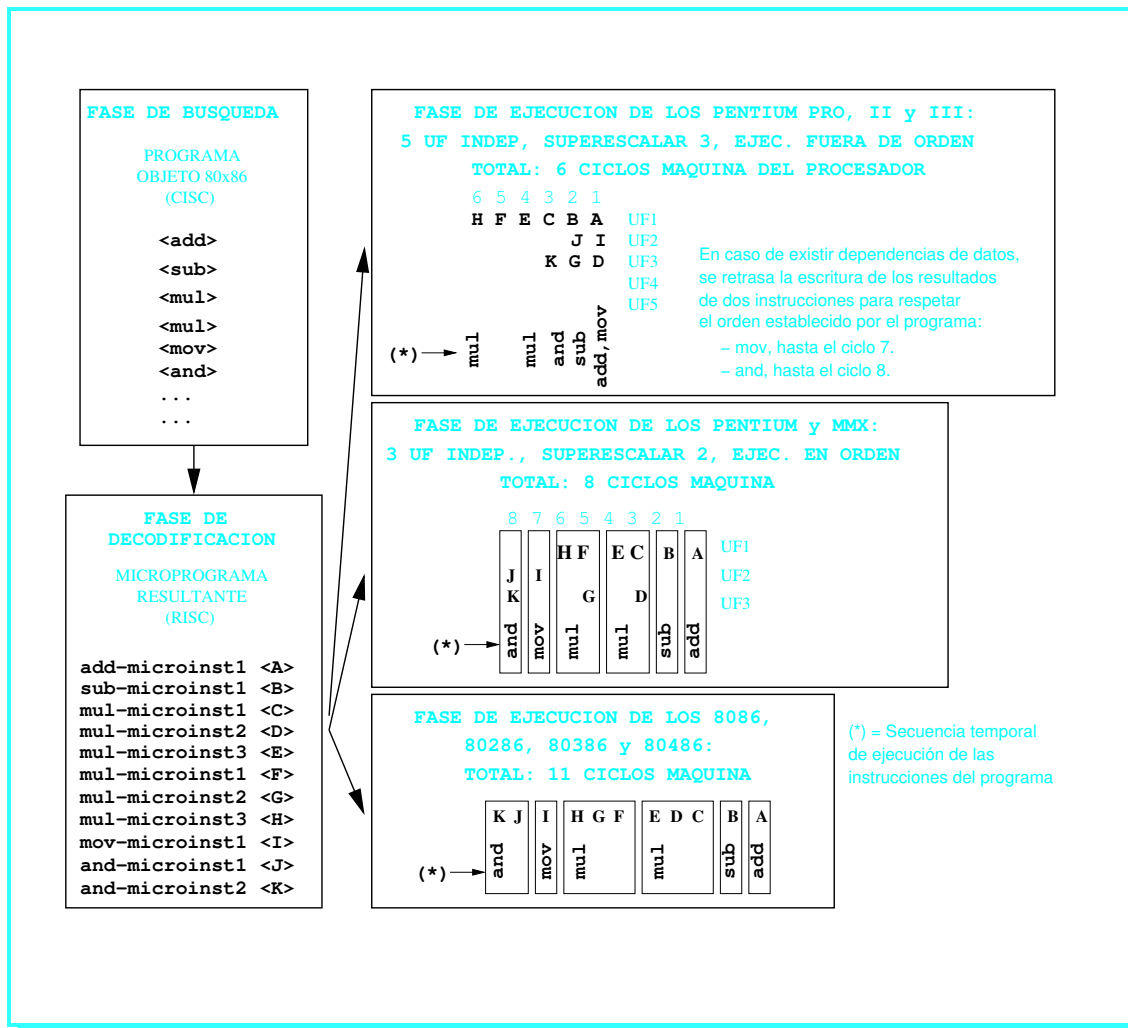


FIGURA 3.6: Evolución de los procesadores de Intel de los últimos veinte años en relación a su ejecución superescalar tomando como ejemplo un breve programa secuencial. El proceso de decodificación que tiene lugar en la parte izquierda del diagrama es una conversión de código x86 al código nativo de cada uno de los procesadores indicados con objeto de mantener la compatibilidad hacia atrás que ha estado presente en todos los modelos de Intel y AMD hasta la fecha. El esquema se repite en diseños tan contemporáneos como el K7 y el Pentium 4.

Ahora bien, el coste de crecer por esta vía es superior al de la segmentación, puesto que aquí tenemos un incremento lineal de complejidad en la unidad de proceso. El hecho de que el coste de integración por transistor haya seguido siempre una evolución descendente explica que estemos ante otro negocio altamente rentable para los diseñadores del procesador.

coste

El primer procesador superescalar, el i960, fue diseñado por Intel en 1989, y podía ejecutar dos instrucciones por ciclo de reloj. Ya en 1995, lo normal era encontrar diseños de 4 instrucciones por ciclo, y aunque aparecieron algunos como el Power PC capaces de ejecutar hasta seis, entre la complejidad y el choque frontal que supone con la forma en que están escritos los programas, el mercado volvió rápidamente sobre sus pasos y se encuentra cómodamente instalado en factores de superescalaridad de entre 3 y 4. La figura 3.5 muestra cómo se complementan la superescalaridad y la segmentación para ejecutar los programas de forma más eficiente.

1989
1995

2003
pág. 60

MAGNITUDES



Ejemplo 3.5: LA SUPERESCALARIDAD EN INTEL

El Pentium de Intel y su versión MMX son procesadores superescalares de factor 2. Disponen de dos ALU para operar con números enteros y una tercera para operar con números reales, permitiendo ejecutar de forma simultánea dos instrucciones aritméticas de tipo entero, o bien una de tipo entero y otra de tipo real (con ciertas limitaciones). Las versiones posteriores de Intel, como el Pentium Pro, II y III, adoptan todos un factor tres de superescalaridad, que puede aplicarse sobre un total de cinco unidades funcionales de ejecución independiente.

Aunque en las secciones dedicadas a cada microprocesador desglosaremos ampliamente todas estas estrategias, la [figura 3.6](#) resume en un sencillo ejemplo las diferencias básicas existentes entre el esquema de ejecución del 80486, el Pentium, y sus hermanos mayores. Los nuevos diseños como el Pentium 4 siguen respetando este mismo factor tres de superescalaridad.

3.3 ► Combinación de segmentación y superescalaridad

pág. 60
compatibles...

Como ha quedado de manifiesto en la [figura 3.5](#), las dos filosofías de diseño anteriores son perfectamente compatibles. De hecho, puesto que la segmentación llega antes al diseño del procesador y es más barata de implementar, no conocemos de modelos comerciales que sean superescalares sin estar segmentados.

...pero
antagónicas
pág. 49

No obstante, existen ciertos conflictos a la hora de poner en práctica las dos ideas simultáneamente, y es que, tal y como ilustramos en nuestra pirámide de la [figura 3.1](#), cada técnica exige unos requisitos diferentes a la capa de bajo nivel del procesador:

- La segmentación descansa fundamentalmente sobre la base de una elevada frecuencia, en el sentido de que sólo un período de reloj muy corto permitirá descomponer cada instrucción en un número elevado de etapas.
- La superescalaridad, por el contrario, necesita de un ingente número de transistores para poder ser llevada a la práctica, y esto sólo se consigue con mejoras en la tecnología de integración.

conflictos

En definitiva, a la segmentación le estorba la superescalaridad porque ésta acarrea un desdoble de la circuitería existente, y por el famoso dicho de la microelectrónica “cuanto más grande, más lento”, tenemos un perjuicio sobre la frecuencia de reloj, perdiendo esperanzas de lograr un elevado número de etapas de segmentación.

De forma similar, a la superescalaridad le estorba la segmentación, porque cuando las etapas son tan minúsculas, se hace difícil incrementar su complejidad replicando circuitería.

dos escuelas

En la práctica, ocurre que los diseños fuertemente segmentados no utilizan un factor de superescalaridad elevado, y que los que apuestan por la superescalaridad reducen el número de etapas de segmentación del diseño. Por ejemplo, el Pentium 4 llega a las 20 etapas de segmentación pero sólo tiene un factor tres de superescalaridad, mientras que el K7 consigue un factor cinco de superescalaridad a costa de reducir a catorce el número de etapas en su cauce de ejecución entero. Al final, las dos estrategias alcanzan un grado de paralelismo inherente en torno a las

60-70 instrucciones simultáneas compatibilizando ambos conceptos, pero dando prioridad sólo a uno de ellos.

Supersegmentación

3.4

La palabra supersegmentación apunta en primera instancia una cosa que no es, puesto que nos lleva a la tentación de aplicar la ecuación *superescalar + segmentado = supersegmentado*, cuando en realidad la ecuación correcta es *segmentado + segmentado = supersegmentado*.

equivoco

En efecto, un procesador **supersegmentado** es aquel que aplica dos veces el concepto de segmentación, la primera al nivel del diseño global, y la segunda al nivel interno de sus unidades funcionales.

concepto

Por ejemplo, una descomposición en cinco etapas de segmentación como la del Pentium, compuesta de etapas de búsqueda, decodificación, lectura, ejecución y escritura, deja abierta la puerta a aplicar una nueva segmentación sobre cada una de las unidades funcionales que intervienen en cada ciclo: la caché de instrucciones, el decodificador de instrucción, la caché de datos, la ALU o el banco de registros, respectivamente.

Tomando como referencia la caché de instrucciones referenciada en la primera etapa, ésta puede ser una caché segmentada como las que invadieron el mercado en la segunda mitad de los años 90 (ver [sección 11.1.4](#)), donde una subdivisión en dos etapas permite simultanear la búsqueda de una instrucción y la localización de su celda de caché asociada, con la lectura en sí del código de instrucción de la anterior y su volcado al bus de datos camino del procesador. Llegamos así a los dos niveles de segmentación mostrados en la [figura 3.7.d](#).

caché
segmentada
p. 118/Vol. 2

pág. 64

Notar que como buena segmentación, ha provocado un desdoble de la señal de reloj, que discurre por el eje temporal de abscisas al doble de frecuencia que en los otros tres diagramas mostrados. En general, la supersegmentación lleva asociada la subdivisión del ciclo de reloj en tantos ciclos como etapas se hayan establecido para el segundo nivel de segmentación, lo que desemboca diseños de muy elevada frecuencia.

frecuencias
elevadas

Aunque en teoría la supersegmentación sea un concepto independiente de la **superescalaridad**, en la práctica, necesita de ella. Esto es así porque una buena implementación supersegmentada siempre incluye la ejecución de instrucciones *fuera de orden* para minimizar el efecto negativo que las dependencias de datos tienen sobre su rendimiento. Y como veremos enseguida, la ejecución fuera de orden sólo tiene cabida en procesadores que sean superescalares.

superescalar



Ejemplo 3.6: EL ALPHA 21264, O LA SUPERSEGMENTACIÓN EN SU MÁXIMA EXPRESIÓN

Un clásico ejemplo de diseño supersegmentado es el procesador Alpha 21264 de Digital, procesador RISC de principios de la década de los 90 y hermano menor del Alpha 21364 de Compaq, el que decodificó el mapa del genoma humano en la empresa Celera Genomics. El segundo nivel de segmentación de esta arquitectura incluye una descomposición en **nueve** etapas para el caso de la caché (ver [figura 3.11](#)), lo que dió lugar a una frecuencia de reloj de 600 MHz para la versión comercializada a finales de 1997, el marco temporal en el que los Pentium II y demás modelos para PC estaban a menos de la mitad de esa frecuencia.

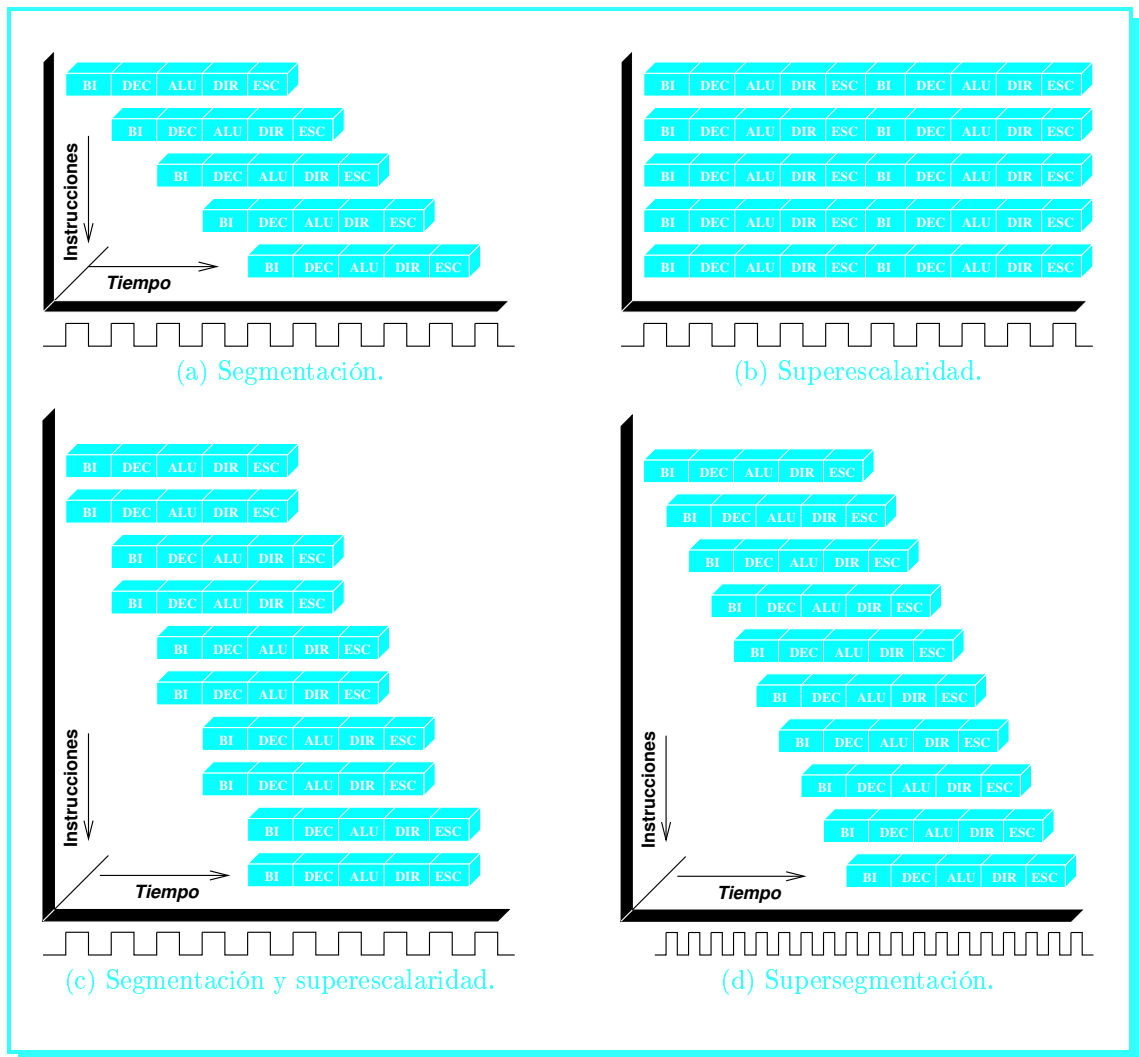


FIGURA 3.7: Comparación de las cuatro técnicas de paralelismo a nivel de instrucción que podemos encontrar en los microprocesadores actuales: (a) Segmentación. (b) Superscalaridad. (c) Segmentación y superscalaridad combinadas. (d) Supersegmentación. En los cuatro casos hemos supuesto que todas las instrucciones pueden descomponerse en cinco etapas: Búsqueda del código de operación (BI), decodificación de instrucción y búsqueda de operandos (DEC), ejecución de operación (ALU), generación de la dirección de destino (DIR) y escritura del resultado (ESC).

3.5 ▶ Dependencias: Las enemigas del paralelismo

dos mundos
diferentes

Cualquier forma de paralelismo a nivel de instrucción ve mermado su potencial de mejora de manera considerable debido a la estructura secuencial que guardan los programas en la capa software, que ignoran cualquier tipo de ejecución simultánea de instrucciones. Es decir, estamos diseñando un procesador que no se corresponde con la forma en que va a ser utilizado por la capa software.

un mundo
virtual

Podría pensarse que la multiprogramación, o más recientemente, la ejecución *multithread*, ayuda a sacar provecho de estos recursos hardware, pero no es así. Estos conceptos tan sólo reparten el tiempo del procesador entre un número de procesos o *threads*, pero en cada momento lo que se ejecuta en su interior es un único flujo de instrucciones.

La percepción de concurrencia que tenemos en nuestro PC no es más que una ilusión pro-

vocada por la enorme diferencia existente entre las dos escalas temporales, la nuestra y la del procesador. Dicho de otra manera, si echamos una foto al procesador en cualquier instante, lo pillaremos ejecutando siempre un único código, escrito para que sus instrucciones se ejecuten una detrás de otra. El compilador y el propio hardware colaboran para aprovechar los recursos disponibles redefiniendo la ejecución del código para habilitar algún tipo de concurrencia, pero siempre tienen la obligación de respetar la secuencialidad definida por el programador. Estos condicionantes introducen riesgos como los siguientes:

el mundo real

- 1 **Dependencias de datos.** Si alguno de los operandos fuente (o de lectura) de una instrucción B es el operando destino (o de escritura) de una instrucción anterior A, B no puede comenzar su ejecución hasta que A no haya finalizado.



Ejemplo 3.7: RIESGO POR DEPENDENCIAS DE DATOS

Sea la siguiente pareja de instrucciones máquina consecutivas dentro de un programa cualquiera.

```
.....
Instr. A:   add R1, R2, R3      # R1 = R2 + R3
Instr. B:   sub R7, R1, R8      # R7 = R1 - R8
.....
```

Podemos ver que la instrucción B toma uno de sus operandos de lectura de R1, por lo que necesita leer el valor de este registro para comenzar a trabajar. Sin embargo, puesto que este mismo valor es escrito por A justo antes de finalizar su ejecución, esto nos obliga a ejecutar B después de A en lugar de simultáneamente, deshabilitando cualquier forma de paralelismo a nivel de instrucción que pudiera tener implementado el microprocesador.

- 2 **Dependencias de control.** Una instrucción de salto condicional impide conocer la siguiente instrucción a ejecutar hasta que no se evalúe su condición de salto, y durante todo ese tiempo deberá detenerse la ejecución concurrente de instrucciones.



Ejemplo 3.8: RIESGO POR DEPENDENCIAS DE CONTROL

Sea la siguiente terna de instrucciones máquina consecutivas dentro de un programa cualquiera.

```
Instr. A:   beq R1, R2, C       # Salta a la instr. C si R1=R2
Instr. B:   sub R10, R11, R12   # R10 = R11 - R12
Instr. C:   add R20, R10, R20   # R20 = R10 + R20
.....
```

En este caso, la instrucción B no puede simultanear su ejecución con la de A: Debe esperar al menos a que ésta evalúe su condición de salto, ya que si resulta que los registros R1 y R2 contienen el mismo valor, la instrucción B no deberá ser ejecutada. Tampoco podemos simultanear la ejecución de las instrucciones A y C, pues C tomará el valor calculado en B si el programa finalmente no salta.

- ③ **Dependencias estructurales.** Una instrucción necesita en uno de sus ciclos de ejecución una unidad funcional que está siendo utilizada por otra instrucción en ese mismo instante.



Ejemplo 3.9: RIESGO POR DEPENDENCIAS ESTRUCTURALES

Sea la siguiente pareja de instrucciones máquina consecutivas dentro de un programa cualquiera.

```
.....
Instr. A:  lw R1, Memoria(1000)           # Carga en R1 el contenido de
                                           # la posic. de memoria 1000)
Instr. B:  lw R2, Memoria(2000)          # Carga en R2 el contenido de
                                           # la posic. de memoria 2000)
.....
```

Si el dato solicitado por la instrucción A a memoria se encuentra en la memoria caché, es obtenido de forma casi inmediata, y enseguida se podrá proceder a ejecutar la instrucción B. Pero si el dato que A necesita no se encuentra en caché, se deberá solicitar de memoria principal, lo que consumirá al menos un centenar de ciclos del procesador, tiempo durante el cual la instrucción B deberá esperar en el caso de que necesite también utilizar la memoria principal.

penalización

pág. 67

Los conflictos anteriores reducen notablemente las oportunidades que pueden presentarse en un código para la ejecución simultánea de instrucciones, y el impacto que cada una de estas dependencias tiene sobre un código dependerá de la naturaleza de la aplicación software. La [tabla 3.6](#) resume el porcentaje de uso de cada unidad funcional del procesador con objeto de darnos una idea de la penalización que supone cada tipo de dependencia en una aplicación entera y de punto flotante. Una regla heurística que se viene cumpliendo tradicionalmente es que una de cada seis instrucciones de un programa es una instrucción de salto que provoca un riesgo por dependencias de control.

soluciones

Las dependencias estructurales se producen por las limitaciones del hardware, y están ligadas a su disponibilidad. En general, su efecto puede mitigarse replicando unidades funcionales o incorporando un banco de registros o una memoria multipuerto.

Las dependencias de datos y control, por el contrario, pueden reducirse mediante técnicas software. A continuación describiremos las dos más importantes: La *ejecución fuera de orden*, para solventar los riesgos por dependencias de datos, y la *predicción de salto*, que hace lo propio con las dependencias de control.

Tipo de microoperación	Tipo de datos de la aplicación	
	Números enteros	Números reales
Ops ALU con datos enteros	50 %	25 %
Ops ALU con datos reales	0 %	30 %
Lectura de operandos	17 %	25 %
Escritura de operandos	8 %	15 %
Salto	25 %	5 %

TABLA 3.6: Frecuencia de uso de las microoperaciones de un programa en función de la naturaleza de la aplicación software. Esto nos da una idea de la influencia de cada tipo de dependencia en el rendimiento del código sobre un procesador con paralelismo a nivel de instrucción, ya que el riesgo por dependencias de datos aparece cada vez que se escribe un operando (8 % ó 15 % para códigos que utilizan exclusivamente números enteros y de punto flotante, respectivamente), y el riesgo por dependencias de control, cada vez que se produce un salto (25 % y 5 %, respectivamente).

3.5.1 Ejecución fuera de orden

La **ejecución fuera de orden** (del inglés, *out-of-order execution*) es una estrategia consistente en alterar, en tiempo de ejecución de las instrucciones, su orden de finalización preestablecido en el programa.

El riesgo que se asume esta vez es mucho más elevado: Cuando una instrucción detiene su ejecución ante una dependencia de datos, el procesador comenzará a ejecutar la siguiente en lugar de quedarse parado. Esta segunda instrucción puede así finalizar antes que la primera, por lo que hay que asegurarse de que no viole la semántica del código establecida por el programador, en particular, que no escriba en algún registro cuyo valor vayan a necesitar las instrucciones precedentes que se encuentren detenidas.

riesgos

El número de verificaciones a realizar para garantizar la consistencia de una ejecución fuera de orden es innumerable, y la complejidad de la unidad de control responsable, desbordante. Piénsese que en un procesador actual, el número de instrucciones que pueden estar activas en un momento dado puede superar perfectamente la cincuentena, y que cada una de ellas puede quedarse atascada en el cauce por razones muy variopintas.

complejidad

Es importante hacer notar que la ejecución fuera de orden sólo puede implementarse sobre un procesador que ya sea segmentado y superescalar. Esto es así porque la idea de la ejecución fuera de orden supone que unas instrucciones *adelanten* a otras durante su paso por el cauce, y ésto sólo es posible si alguna de sus etapas dispone de varias unidades funcionales que permitan a otras instrucciones progresar cuando la anterior se encuentra detenida en esa etapa.

sólo si es superescalar



Analogía 3.1: LAS INSTRUCCIONES Y EL TRÁFICO EN SU FORMA DE FLUIR

Para entender la necesidad de contar con un procesador superescalar si pretendemos incorporarle ejecución fuera de orden, resulta útil ver el cauce de ejecución como una carretera, sus unidades funcionales como los carriles de la misma, y las instrucciones como los vehículos que por ella circulan, todos en la misma dirección: Un coche sólo puede adelantar a otro cuando dispone de carriles alternativos por los que circular que no le hagan depender de la velocidad del vehículo que le precede.

Al igual que en carretera hay vehículos lentos y otros más veloces, en el cauce de ejecución también coexisten instrucciones rápidas (las enteras) con otras que no lo son tanto (multimedia) y un tercer grupo bastante más lento (las de punto flotante), sin contar con aquellas que mientras no se ejecuten no se sabrá su velocidad (las de acceso a memoria, que para no complicar el ejemplo vamos a descartarlas).

Desde la llegada del Pentium MMX, el procesador dispone de vías de circulación separadas para cada una de estas tres clases de instrucciones, y no por ello debemos pensar que ya es superescalar. La superescalaridad es un aspecto más ligado a la replicación de unidades funcionales de ejecución, esto es, a la facultad de ejecutar simultáneamente una serie de instrucciones *de la misma clase*. La ejecución fuera de orden actúa de forma separada sobre cada una de estas clases, ya que el hecho de que cada clase disponga de su banco de registros propio, le impide entrar en conflicto con las demás.

Así, el tráfico es siempre común en los dos primeros tramos de carretera correspondientes a la búsqueda y decodificación de instrucción, y a partir de ahí, la carretera se bifurca en derivaciones secundarias específicas para cada tipo de vehículo, donde cada uno de ellos sólo podrá adelantar a los de su misma clase, y únicamente en aquellos puntos en los que la arquitectura haya puesto un desdoble de carriles de circulación, esto es, múltiples unidades de ejecución. Esta multiplicidad será el aspecto que determinará el grado de superescalaridad del chip en cada una de sus etapas de ejecución.

La siguiente tabla trata de sintetizar la analogía comentada:

Flujo de vehículos	Flujo de instrucciones	Comportamiento
Coches deportivos	Instrucciones enteras	Muy rápido
Utilitarios	Instrucciones multimedia	Rápido
Tráfico pesado	Instrs. de punto flotante	Lento
Carretera principal durante búsqueda y decodificación	Tramo común del cauce segmentado	Tráfico sincronizado
Derivaciones secundarias durante lectura, ejecución y escritura de operandos	Subcauces especializados	Tráfico desacoplado
	Superescalaridad \Rightarrow	Adelantamiento

En cualquier caso, los cambios que se producen durante la ejecución de las instrucciones del programa no deben nunca alterar la tarea global a realizar por el mismo. Por lo tanto, la unidad de control del procesador monitorizará todos y cada uno de los posibles adelantamientos para examinar su licitud, impidiendo aquellos que puedan desembocar en resultados erróneos. La circuitería que se hace necesaria para ello provoca un incremento del área de integración del procesador, pudiendo incluso alterar su camino crítico, con el consiguiente impacto sobre la frecuencia de reloj.

verificaciones

3.5.2 Predicción de salto

concepto

La técnica de **predicción de salto** trata de eliminar las dependencias de control de un programa a través de una predicción en la que el microprocesador intenta adivinar lo que hará una instrucción de salto condicional *antes* de que se evalúe su condición de salto. Puesto que sólo puede hacer dos cosas, saltar o no saltar, la probabilidad de acertar en dicha predicción es bastante elevada, pero como en cualquier caso nuestra apuesta es una conjetura, a esta técnica también se le denomina **ejecución especulativa**.

La predicción puede ser de dos tipos:

- **Estática.** El procesador apuesta siempre por la misma premisa, a saber, “el programa salta

siempre que ejecuta una instrucción de salto” o “el programa no salta nunca”.

- **Dinámica.** El procesador es capaz de hacer una suposición u otra dependiendo del comportamiento del programa en su pasado más reciente.

Para implementar la predicción dinámica es necesario registrar el comportamiento histórico de las instrucciones de salto del programa. De esto se encarga la **BTB (Branch Target Buffer)**, o *búfer para los destinos de los saltos*, una nueva unidad funcional del procesador que guarda en cada una de sus entradas los siguientes **campos de información**:

- | | |
|--|---|
| <ol style="list-style-type: none"> ❶ El código de una instrucción de salto del programa. ❷ Su dirección de salto, esto es, por donde prosigue su ejecución el código en caso de que la instrucción salte realmente. ❸ Un grupo de bits que conforman el <i>historial</i>, siendo este grupo más numeroso cuanto mayor sea el pasado temporal que quiera archivarse para dicha instrucción. Con el historial crece el coste de la BTB, pero también su probabilidad de acertar en la predicción. | <p>BTB</p> <p>información en la BTB:</p> <ul style="list-style-type: none"> - código - dirección - historial |
|--|---|

La BTB **funciona** de forma similar a como enseguida veremos que trabaja una memoria caché:

- | | |
|---|---|
| <ul style="list-style-type: none"> ■ Si la instrucción de salto no se ha ejecutado nunca, entrará en ella sustituyendo a otra que sea de poco interés para el procesador. En ese caso, al no disponerse de historial para la instrucción, el procesador puede activar mecanismos de predicción alternativos como la propia predicción estática. En la práctica, casi todos los procesadores encadenan estas dos estrategias. ■ Si la instrucción ya se ha ejecutado antes, se encontrará en ella con mayor probabilidad cuanto mayor sea el tamaño de la BTB y menor sea el número de instrucciones de salto que se hayan ejecutado entre medias. En cada ejecución anterior, el procesador ha registrado en la BTB si esta instrucción saltó o no, y ahora utiliza dicha información para mostrar un comportamiento adaptativo: Si la instrucción presenta una marcada tendencia al salto, la consigna de la unidad de predicción de saltos será “suponer salto realizado”, y si su historial muestra un perfil secuencial, la consigna será “proseguir ejecutando secuencialmente”. | <p>funcionamiento</p> <p>fallo en la BTB:</p> <p>predicción estática</p> <p>acierto en la BTB:</p> <p>comportamiento adaptativo</p> |
|---|---|

Por otra parte, aunque hasta ahora sólo hemos hablado de los saltos condicionales, la BTB también soluciona las dependencias de control debidas a **saltos incondicionales**. Aunque pueda pensarse que éstos no introducen conflicto alguno debido a que su predecible comportamiento (siempre saltan), hay que resaltar que un salto incondicional lleva asociado el cálculo de una dirección de salto. Este cálculo se efectúa normalmente en una etapa bastante tardía del cauce segmentado, lo que detiene la entrada de nuevas instrucciones en él hasta tanto no se sepa por dónde prosigue el programa. Dado que un acierto en la BTB ya proporciona la dirección del salto, el conflicto desaparece, no siendo necesario esperar a la conclusión de dicha etapa.

En este sentido, podemos considerar a la BTB responsable de llevar a cabo una estrategia de anticipación en el tratamiento de las dependencias de control, de igual manera que también existen mecanismos más sofisticados para llevar a cabo la anticipación de valores en la resolución de las dependencias de datos.

saltos incondicionales

anticipación

SECCIÓN 3.4

Memoria caché integrada

Tradicionalmente se le presta mucha atención a la rapidez de cómputo, cuando en el interior del procesador tienen lugar muchos más accesos a datos que operaciones en sí.

Veámoslo con un ejemplo. La [figura 3.8](#) muestra la instrucción más típica de un procesador. En ella se requieren cuatro accesos a distintas fuentes de información para realizar una sola operación. El procesador se encuentra así mucho más limitado por la obtención de datos que por la celeridad de su computación.

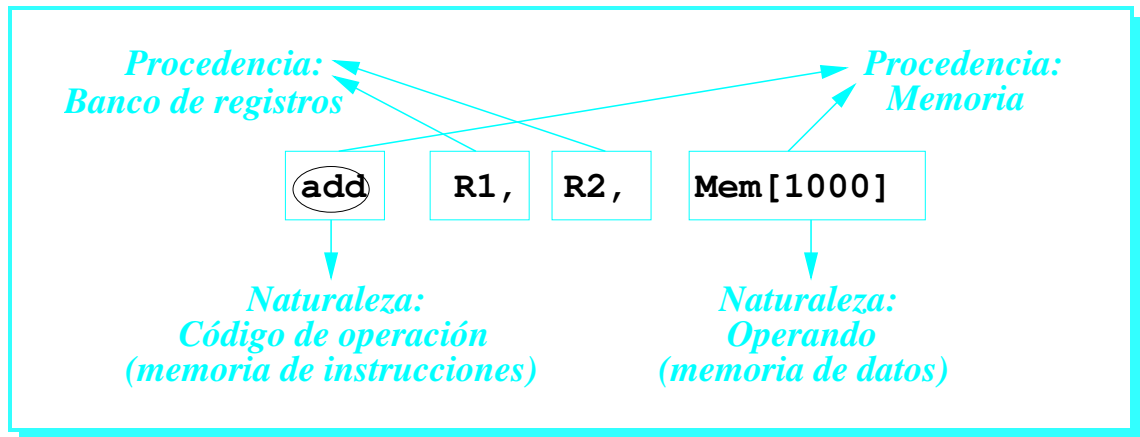


FIGURA 3.8: Las tareas involucradas en la computación de una instrucción típica por parte del procesador. Se han señalado con una elipse las operaciones a realizar, y con un rectángulo los accesos a datos. La proporción es de cuatro a uno en favor de estos últimos.

De los datos de la instrucción `add` en la [figura 3.8](#), dos proceden de memoria y otros dos del banco de registros del procesador. Pero como ya sabemos que un procesador actual dispone de un elevado grado de paralelismo a nivel de instrucción, la pregunta que surge es: ¿Cómo puede el procesador alimentarse del enorme volumen de información que esa operativa demanda, y todo ello a una velocidad superior a 1 GHz?

el dilema

la solución

pág. 73

Diremos en primer lugar que como la memoria no es tan rápida, el número de etapas de segmentación se verá ampliado cuando se procese información de este tipo. Y en segundo lugar, que se habilitarán hasta cinco unidades funcionales distintas dedicadas a la exclusiva tarea de proporcionar datos al procesador. Estas unidades funcionales serán presentadas en la [sección 3.4.2](#). Lo que ahora pretendemos ilustrar es su necesidad.



Ejemplo 3.10: LA NECESIDAD DE UNA JERARQUÍA DE MEMORIA

Según nuestras propias estimaciones, hoy en día la velocidad real de acceso a memoria principal es unas 3.000 veces más lenta que el acceso al banco de registros interno del procesador.

Esto significa que de no contar con una jerarquía de memoria que haga de intermediario entre estos dos niveles, un Pentium 4 de 2 GHz trabajaría en la actualidad a 666 KHz, puesto que al menos tendría que acceder a memoria una vez por cada instrucción que ejecuta para recoger su código de operación. Esa frecuencia de trabajo está por debajo de la frecuencia original del ENIAC, el primer computador de la historia, que data del año 1945.



Analogía 3.2: EL PAPEL DE LA FOTOCOPIADORA, O LOS RECURSOS QUE HACEN FALTA PARA SUMINISTRAR DATOS CON CELERIDAD AL PROCESADOR

Imagínese que compramos una fotocopidora en Málaga, nos preocupamos por adquirir un modelo ultrarrápido que realiza una copia en 3 sg., y luego tenemos que ir a Almería a por el folio de papel cada vez que queramos efectuar una copia. Este ejemplo coincide con un procesador sin caché, pues las 2 horas y media del trayecto encajan con el ratio 1:3000 que acabamos de apuntar en el ejemplo anterior.

Habilitar el cajón alimentador de papel tiene un propósito similar al del banco de registros del procesador: Garantizar el suministro inmediato, y es tan necesario, que el sistema no se concibe sin él. Pero se trata de un recurso limitado, así que cuando la bandeja agota sus folios, tenemos de nuevo visita pendiente a Almería.

Apilar unos paquetes de folios junto a la fotocopidora tiene el mismo efecto que habilitar una caché de primer nivel en el procesador: Cuando el sistema agote la bandeja, tomará el papel de otro sitio al que se tarda tres o cuatro veces más en acceder, pero que permite el funcionamiento autónomo por un tiempo, evitando tener que salir al exterior.

Con esta solución, el sistema funciona muy bien, pero si ahora decidimos adquirir una docena de fotocopadoras más, la solución se queda pequeña, como también se le quedó pequeña una sola caché al procesador cuando optó por replicar unidades funcionales en sus primeros diseños segmentados y superescalares.

Esto motivó la aparición de un segundo nivel de caché interno al procesador, dotado de una mayor capacidad, pero algo más lento. Lo mismo que si decidimos adquirir para nuestra fotocopidora un almacén contiguo en el que dar cabida a varios millones de folios. El proveedor principal de papel sigue estando en Almería, pero ahora nuestro sistema ha conseguido una autonomía propia de varios millones de copias. Las mismas operaciones que hoy en día puede realizar un microprocesador sin requerir el concurso de la memoria principal.

La memoria caché es objeto de nuestro estudio en el [capítulo 11](#), por lo que el lector puede realizar una incursión allí si no se encuentra suficientemente familiarizado con este concepto. Respecto a nuestras estimaciones sobre la lentitud de acceso a memoria principal comparada con caché, provienen de un análisis realizado en la [sección 13.2.2](#) en el contexto del futuro que dibujamos para la memoria caché.

← p.115/Vo1.2

← p.160/Vo1.2

Y es que cuando se trata de hablar de la memoria caché, ya no sabemos donde encaja mejor: Por el nombre y por su pasado, el [capítulo 11](#) es su ubicación natural; por su presente y futuro, su sitio está aquí, dentro de la cobertura dedicada al procesador, pues para eso hace ya un tiempo que al menos dos niveles de caché van enquistados dentro de él. Así que aquí la trataremos como una extensión del banco de registros del procesador, ilustrando las diferencias en su funcionamiento, su interrelación con las unidades funcionales del procesador, y los diferentes buses por los que establece su estrecha vinculación con él. Y dejaremos para el [capítulo 11](#) su cobertura como ente propio, donde visitaremos su tecnología distintiva y la serie de implementaciones comerciales de que dispone como chip independiente.

nuestro
tratamiento

Cuando uno afronta la compra de un procesador, su caché es uno de los puntos clave en los que debe fijarse. Y sin embargo, la documentación que los distribuidores proporcionan al respecto es pírrica, indicando únicamente su tamaño, y gracias, porque en más ocasiones de las que sería

pieza clave

deseable ni siquiera indican a qué caché se están refiriendo, y hay al menos tres dentro de un procesador actual.

metamorfosis A nivel técnico, la laguna tampoco es despreciable. Conocemos multitud de libros que explican de maravilla los principios de localidad espacial y temporal y los conceptos de línea o conjunto de caché, parámetros que irán eternamente ligados a su estructura interna. Pero la caché ha sufrido tal grado de metamorfosis en los últimos años, que podemos indicarle que en los libros anteriores a 1995 encontrará buenos fundamentos sobre ella, pero una contribución más bien escasa acerca de las claves de su rendimiento actual.

desafío La cantidad de variantes que han surgido desde esa fecha ha supuesto para nosotros todo un desafío a la hora de poner un poco de orden en esta sección. Trataremos de apoyarnos en continuas referencias a modelos comerciales con objeto de situar cada variante en su contexto dentro del mercado.

acelerador La complejidad anunciada no deja de ser, además, una paradoja. Conserve en mente una verdad de perogrullo: *Toda caché tiene como objetivo exclusivo acelerar el acceso a memoria; ninguna variante incorpora funcionalidad adicional al sistema, o espacio de almacenamiento suplementario. Por tanto, por lo que al diseño de cachés respecta, las cosas, o se hacen muy rápido, o no sirven absolutamente para nada.* Como esta premisa se encuentra bastante reñida con las ideas sofisticadas, veremos que en no pocas ocasiones el camino más simple será también el más acertado.

4.1 ▶ Breve sinopsis histórica

banco de registros En sus orígenes, los microprocesadores apenas disponían de unas pocas celdas de memoria dentro del propio chip para almacenar los operandos fuente y destino de sus instrucciones. Estas celdas se estructuraban en un banco de registros para almacenar los operandos fuente y destino de las instrucciones, que se referenciaban de forma directa y explícita en el propio código de instrucción

en los años 80 Ya durante los años 80, el concepto de memoria interna al chip se extendió a una memoria caché de unos pocos kilobytes, hito que situamos en la cuarta generación de microprocesadores para PC (1989) con la llegada del 80486 de Intel y el 68040 de Motorola. Esta memoria respondía mucho más rápido que la memoria principal externa ubicada en la placa base, por lo que si el cuello de botella del sistema se situaba en el acceso a memoria, se lograba una sustancial mejora en el rendimiento del microprocesador en general.

en los años 90 En la década de los 90, los diseñadores de microprocesadores fueron aumentando progresivamente el tamaño de estas memorias y estructurándolas en niveles. El buque insignia de estas transformaciones fue el procesador Alpha de Digital, que en su versión 21264 (1995) disponía ya de 112 Kbytes de caché interna organizada en tres memorias a dos niveles, resultando una jerarquía de memoria interna que es fiel reflejo de lo que hoy encontramos en los procesadores contemporáneos.

p. 8/Vol. 1.2 La [figura 9.2](#) ilustra la ubicación de todos estos niveles jerárquicos de memoria en el contexto de la cuarta, quinta, sexta y séptima generación de microprocesadores.

Dentro ya de los modelos comerciales, tenemos ejemplos concretos en abundancia:

pág. 74 ❶ El Pentium. La [foto 3.2](#) muestra su aspecto externo (arriba) y su área de integración (abajo), donde las dos cachés L1 para datos e instrucciones se han delimitado en su parte izquierda.

pág. 76 ❷ El Pentium Pro. La [figura 3.9](#) nos muestra una radiografía de este procesador, donde la caché L2 se suministra ya junto al microprocesador en lugar de venir integrada en la placa base.

- ③ El Pentium II. En la [figura 3.10](#) podemos observar un diagrama de cómo tiene dispuestos sus dos niveles de memoria caché. ➡ [pág. 77](#)
- ④ El Pentium III. En su versión de 0.18 micras, la caché L2 se incluye ya dentro del propio chip del procesador, según se aprecia en la parte derecha de la [foto 3.2](#). Este estatus permanece vigente durante toda la séptima generación, alcanzando ya el momento presente. ➡ [pág. 74](#)

La [sección 3.4.4](#) formalizará todas las posibles ubicaciones. Hasta llegar allí, la [sección 3.4.2](#) se ocupará de profundizar más en los diferentes niveles de memoria de forma separada, y la [sección 3.4.3](#) dará cuenta de algunas optimizaciones interesantes. ➡ [pág. 81](#)
➡ [pág. 77](#)

Jerarquía

◀ 4.2

Los niveles de memoria internos al procesador dan lugar a una jerarquía en función de la proximidad a su núcleo de ejecución. Comenzando por la capa más interior, podemos distinguir:

- ① **El banco de registros.** Su **tamaño** suele estar comprendido entre 32 y 512 registros, siendo mayor en microprocesadores de tipo RISC (ver [sección 3.5.2](#)). Con respecto a la **anchura** de cada registro, coincide con el tamaño de palabra que puede procesar la ALU. Por ejemplo, en un Pentium III, el banco de registros para datos enteros es de 32 bits, mientras que el banco de registros que alberga a los datos reales o de punto flotante tiene una anchura de 80 bits, pues ésa es la anchura de las unidades funcionales sumadora, multiplicadora y divisora que conforman su FPU. tamaño
➡ [pág. 97](#)
anchura
- ② **El búfer de prebúsqueda de instrucciones/datos.** Relacionado con el paralelismo a nivel de instrucción en general, y dentro de él más con el carácter superescalar del procesador, ya que la ejecución simultánea de varias instrucciones en cada ciclo de reloj asume implícitamente la existencia de algún mecanismo que suministre las instrucciones al ritmo que el procesador las ejecuta. superescalar

Este búfer no es más que un almacén intermedio, normalmente implementado mediante una cola ³, donde se han traído las instrucciones candidatas a ser ejecutadas en un futuro inmediato, y desde donde pueden dirigirse al procesador con una agilidad extrema en cuanto sean requeridas por éste. De esta manera, se cubre una doble función:

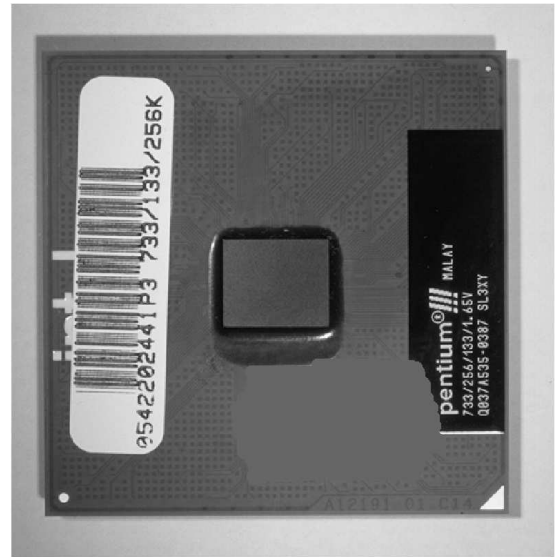
 - ① Desacoplar la entrada de información a las unidades funcionales del proceso de ejecución de instrucciones en sí. desacoplo
 - ② Actuar de acelerador en la captura de dicha información. acelerador
- ③ **La caché de primer nivel (L1).** Suele estar separada en dos: Una para datos y otra para instrucciones. Esta separación viene como consecuencia del carácter segmentado del procesador. En efecto, dado que una de las etapas en que se divide la ejecución de instrucciones accede a la caché de instrucciones para traerse el código de la instrucción a ejecutar, y otra etapa accede a la caché de datos para traerse los operandos fuente o almacenar el operando destino, la ejecución simultánea de varias instrucciones en distintas etapas del cauce segmentado pasa por implementar estas cachés como componentes independientes. segmentación

Obsérvese que, vista esta dualidad entre instrucciones y datos en el primer nivel de caché, sólo existe un nivel que le precede en dirección al procesador, ya que el banco de registros se interpone entre éste y la caché de datos, mientras que el búfer de prebúsqueda de instrucciones se ubica por delante de la caché de instrucciones. dualidad
- ④ **La caché de segundo nivel (L2).** Una caché más grande pero más lenta que la anterior, albergando conjuntamente datos e instrucciones, que se coloca en la placa base hasta el ubicación

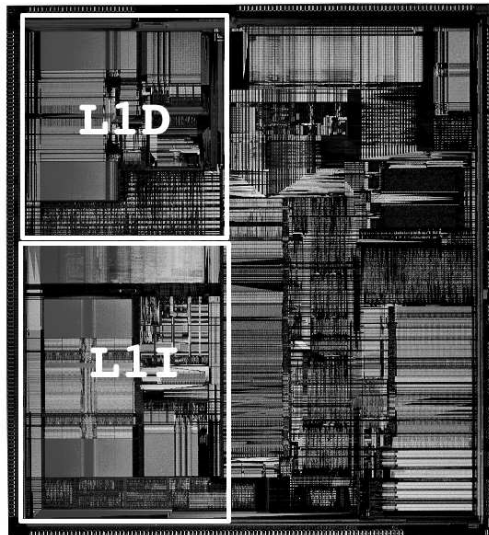
³Una estructura de datos FIFO - First In First Out - en la que los datos salen en el orden en el que llegan.



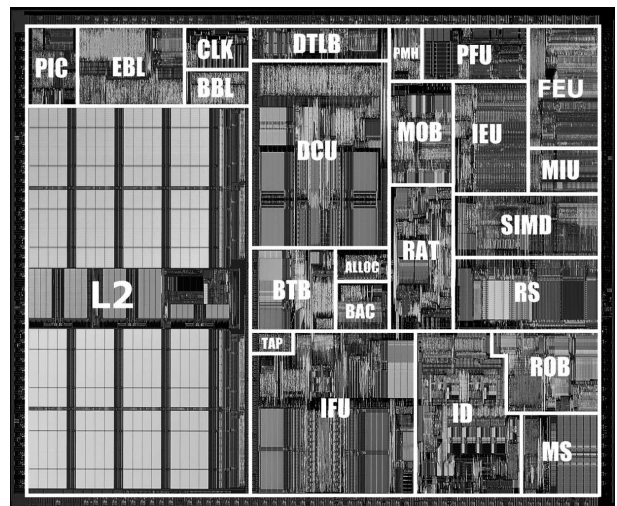
(a)



(b)



(c)



(d)

Fotos de las áreas de integración: Cortesía de Intel

FOTO 3.2: Las memorias cachés se van integrando en el chip procesador conforme se avanza en su integración. En el primer Pentium de 0.35 micras (foto a) sólo se disponía de dos cachés L1 para datos e instrucciones, ambas de 8 Kbytes, y marcadas en el área de integración (foto c). En el Pentium III de 0.18 micras (foto b) estas cachés son ya de 16 Kbytes cada una, marcadas en la foto (d) como DCU - centro arriba - e IFU - centro abajo. Además se incluye una caché L2 de 256 Kbytes - parte izquierda. Las fotos (a) y (b) están tomadas a escala real, por lo que puede apreciarse que el área de integración de los 28 millones de transistores del Pentium III (rectángulo de la parte central) es muy inferior a la del Pentium de sólo 3.1 millones.

Pentium (accesible a través del bus local), y que ha pasado a formar parte integrante del propio chip del procesador en casi todos los diseños contemporáneos.

Esta mejora ha venido obligada por la evolución de las aplicaciones software, cuya mayor dimensión ha hecho que la caché de primer nivel sea insuficiente para contener el uso masivo de datos externos al microprocesador. La caché L2 se hace entonces necesaria pa-

necesidad

Tipo de memoria interna al microprocesador	Hecho con el que puede relacionarse	Generación del microprocesador en la que aparece
Banco de registros	Indispensable en la arquitectura de un microprocesador	Primera
Búffer de prebúsqueda de instrucciones	Paralelismo a nivel de instrucción: Superescalaridad	Quinta
Memoria caché de primer nivel (L1)	4 Gbytes de memoria direccionable Paralelismo a nivel de instrucción: Segmentación Mejoras en la integración	Tercera Cuarta Quinta
Memoria caché de segundo nivel (L2)	Nuevo formato PCB multichip con zócalo tipo Slot para el microprocesador	Sexta

TABLA 3.7. Los diferentes niveles que componen la jerarquía de memoria interna de un microprocesador junto al hecho más relevante que provoca su aparición y la generación de microprocesadores en que esto sucede.

ra aprovechar la propiedad de *localidad* de los programas en el acceso a memoria, que se encuentra un tanto diluida por el enorme área de memoria sobre el que éstos se extienden.

La [figura 3.9](#) muestra la ubicación física de todos estos niveles de memoria para el primer microprocesador de Intel que los incorporó conjuntamente: El Pentium Pro. En la parte superior de la figura puede apreciarse el enorme espacio ocupado por el patillaje externo y la composición de su núcleo multichip: la caché L2 se integra separadamente en el chip de la derecha, mientras que el chip de la izquierda contiene al resto de unidades funcionales de la CPU (incluyendo las cachés de primer nivel), cuya distribución geográfica presentamos en la parte inferior de la figura.

El chip de la caché aglutina 15.5 millones de transistores, mientras que el de la CPU sólo contiene 5.5 millones. Y es que ya avisamos en la [sección 3.2](#) de que los transistores de la caché ocupan un espacio de silicio inferior al de otras unidades funcionales de la CPU. En general, los diseños actuales tienen una marcada tendencia a incluir cachés cada vez más grandes, que en los casos más extremos han llegado a copar hasta el 80 % del total de transistores.

A modo de recopilación para los niveles de memoria que componen la jerarquía interna del microprocesador, mostramos en la [tabla 3.7](#) la secuencia cronológica de aparición de todos ellos y su relación con las distintas generaciones de microprocesadores.

La arquitectura de un microprocesador con todos estos niveles de memoria internos contempla dos vías de comunicación separadas (ver [figura 3.10](#)):

- ❶ El **bus backside**, que conecta los dos chips que componen el microprocesador, y que se utiliza para el transporte de datos desde la caché L2 a la caché L1 y el microprocesador.
- ❷ El **bus frontside**, que conecta el patillaje externo del microprocesador con el juego de chips de la placa base y que se encarga de transferir la información entre la memoria principal y esta caché de segundo nivel.

A partir de ahora, utilizaremos las denominaciones de *bus trasero* para el primero y *bus frontal* para el segundo, sinónimo para nosotros del sempiterno bus local de la placa base que desemboca en el patillaje del procesador cuando no existen cachés de por medio.

➡ [pág. 76](#)
visión de conjunto

➡ [pág. 50](#)
espacio en silicio

recopilación

➡ [pág. 77](#)

bus trasero

bus frontal

sinónimos

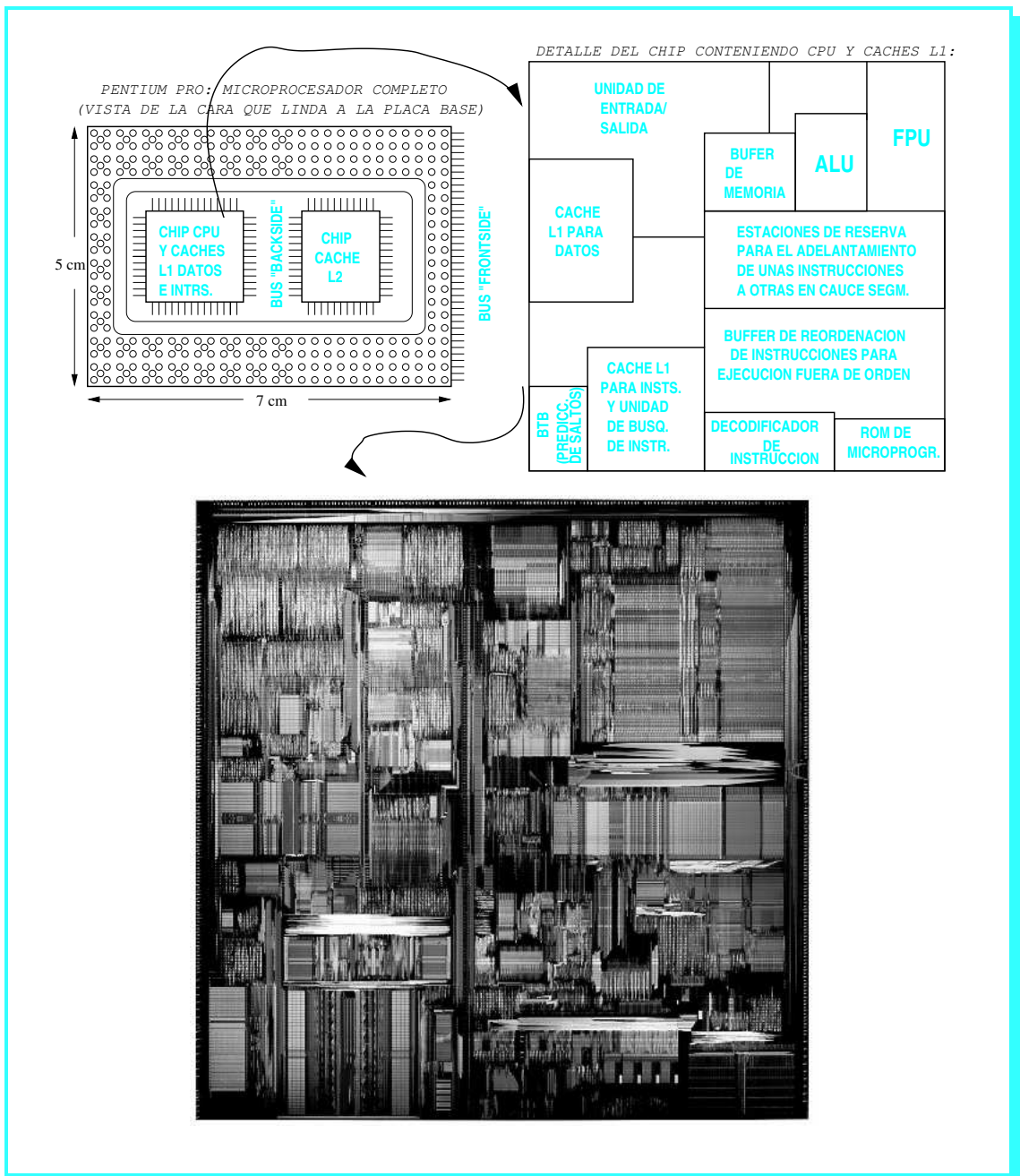


Foto interior: Cortesía de Intel

FIGURA 3.9: Radiografía del Pentium Pro para ilustrar la ubicación de cada uno de los niveles de su jerarquía de memoria. Los diagramas están realizados a escala real para mostrar el enorme espacio físico dedicado a los 387 pines del patillaje externo. Rodeados del mismo, arriba a la izquierda podemos distinguir dos chips: Uno para la caché L2, y otro para la CPU en sí y las cachés de primer nivel. A la derecha: Desglose de las unidades funcionales. Abajo: Detalle del área de integración en una foto real que detalla la constitución interna del chip.

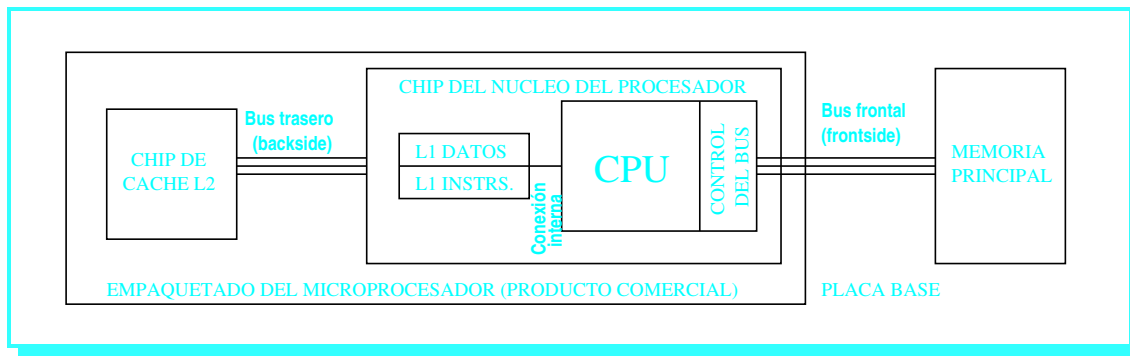


FIGURA 3.10: Las memorias cachés y los buses asociados a la arquitectura de un microprocesador de sexta generación.

Optimizaciones

4.3

Tomando como partida un microprocesador dotado de los niveles de memoria caché L1 y L2 y de los buses frontal y trasero ya descritos, el abanico de optimizaciones que pueden llevarse a cabo en su interior es muy amplio. A continuación recogemos las que a nuestro juicio son más representativas de la situación actual del mercado.

4.3.1 Buses desacoplados

La lógica de control y sincronización interna del microprocesador permite el funcionamiento independiente y simultáneo de los buses frontal y trasero. Esta característica ha sido desarrollada por muchos fabricantes. Intel la incorpora en sus procesadores domésticos a partir del Pentium Pro con el nombre de **Dual Independent Bus (DIB)** (ver [sección 14.4.3](#)).

DIB

➔ p.197/Vol.2

Esta optimización entra en juego cuando el dato que el procesador busca no se encuentra en ninguna de las cachés internas, requiriéndose un acceso a memoria principal. En ese caso, se debe traer una línea de datos íntegra de memoria principal, pero el procesador podrá trabajar con el dato que solicitó mientras se procede de forma simultánea a la carga de la línea en la caché L2.

4.3.2 Caché no bloqueante

Cuando una caché es **no bloqueante**, pueden realizarse nuevos accesos a la misma mientras uno o más fallos están siendo cursados.

La conjunción de esta característica con el uso de buses desacoplados nos permitirá que cuando un dato no se encuentre en la caché L2, puedan cursarse nuevas peticiones a L2 por el bus trasero mientras se resuelve el fallo anterior por el bus frontal. Si es en la caché L1 donde no hemos encontrado el dato, el funcionamiento es idéntico: El bus trasero se encarga de traer este dato de L2 al tiempo que nuevas peticiones del procesador pueden acceder internamente a L1.

funcionamiento

Las cachés no bloqueantes se incorporan a los microprocesadores para PC coincidiendo en el tiempo con la llegada del segundo nivel L2. En el caso que nos sirve de referencia, la L2 es la aliada natural de la arquitectura DIB en el Pentium Pro, perdurando ambas en todos los diseños posteriores de Intel (Pentium II, Celeron, Pentium III y Pentium 4).

origen

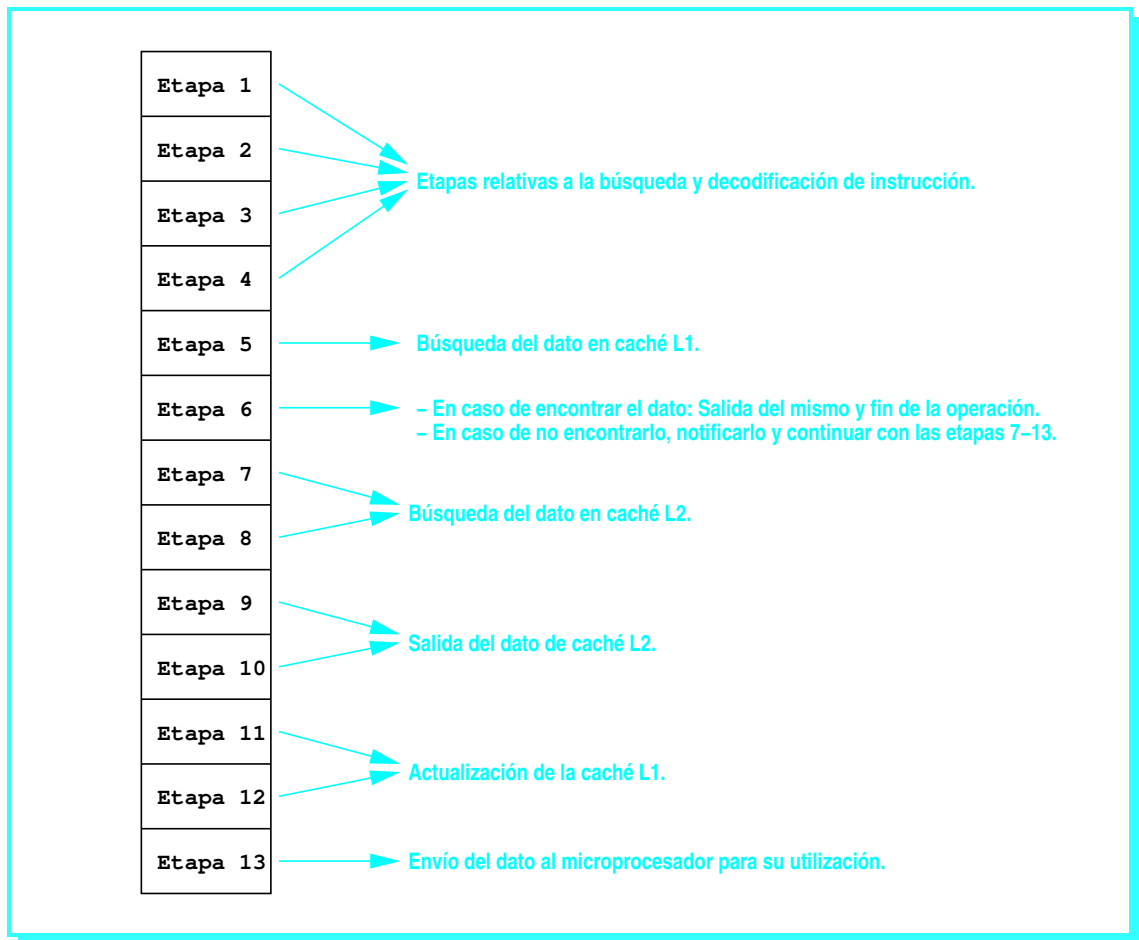


FIGURA 3.11: El diseño supersegmentado de los procesadores 21164 y 21264 de Digital contempla hasta un máximo de nueve etapas de segmentación dedicadas a la operativa de funcionamiento de sus múltiples cachés segmentadas.

4.3.3 Caché segmentada

Se trata de aplicar al funcionamiento interno de la memoria caché la idea de la **segmentación** que ya explicamos sobre los procesadores en la [sección 3.3.1](#).

☛ pág. 59

2 o 4 etapas

Por la velocidad intrínseca de la caché, lo más común es encontrarla segmentada en dos etapas si es interna o en cuatro si es externa y/o funciona a la mitad de frecuencia que el núcleo del procesador: En la primera etapa se realiza la consulta del dato a localizar en L1, y en la segunda se extrae dicho dato en caso de ser encontrado en L1 o se lanza la petición a L2 en caso contrario; suponiendo que el mayor tamaño de ésta última le obliga a funcionar a la mitad de velocidad que el procesador (y su caché L1), su funcionamiento sería similar pero consumiendo dos etapas para la salida del dato correspondiente.

supersegmentados

Alpha

La segmentación de la caché interna encuentra su ubicación más natural en procesadores supersegmentados, en los que una subdivisión del ciclo de reloj asociado a la respuesta de las cachés posibilita la distinción de estas subetapas de búsqueda y salida de datos. Por ejemplo, en los diseños 21164 y 21264 del procesador Alpha de Digital, esta subdivisión permitió la consecución de una frecuencia de reloj que marcó el techo de referencia durante toda la década de los 90, y donde un acceso a caché puede gastar hasta nueve ciclos, segmentados según se detalla en el diagrama de la [figura 3.11](#).

Desde el punto de vista del rendimiento de una jerarquía de memoria dotada de dos niveles de caché L1 y L2, podría resultar un tanto paradójico el hecho de que existan casos como el del Pentium Pro o los Pentium II y III Xeon en los que la caché L2 funciona a la misma velocidad que la caché L1. Sin embargo, son las tres optimizaciones vistas hasta ahora las que dan sentido a semejante configuración:

- ❶ La presencia de vías de comunicación desacopladas para el acceso a estas cachés permite reducir la probabilidad en la aparición de riesgos estructurales en procesadores con un elevado grado de paralelismo a nivel de instrucción, al permitirse que una instrucción acceda a la caché L1 de instrucciones, otra a la L1 de datos, y una tercera a la caché L2. acceso múltiple
- ❷ La utilización de cachés no bloqueantes y segmentadas permite al nivel L1 independizar el funcionamiento del procesador y la caché L2, y así, cuando no se encuentre un dato en L2, L1 puede estar sirviendo otros datos al procesador al tiempo que L2 busca el suyo en memoria principal. funcionamiento
desacoplado
- ❸ Al nivel de una petición individual a memoria, la diferencia entre acceder a caché L1 ó L2 la detallaremos en la [sección 3.4.7](#), dedicada al análisis de su rendimiento. acceso
individual
☛ pág. 88

4.3.4 Caché con lectura anticipada

Esta caché supone un refinamiento en el modo de proceder de una caché no bloqueante. Tiene por objeto acelerar la lectura de un dato por parte del procesador cuando éste no se encuentra en caché, simultaneando allí la recepción del bloque que lo contiene con el envío del dato concreto al procesador. refinamiento

En este sentido, disponemos de dos estrategias básicas: dos estrategias

- **Early restart** (reinicio prematuro). Aguarda a que llegue la palabra del bloque requerida por el procesador para enviarla al mismo, evitando por tanto la espera restante hasta que se complete la transferencia del bloque.
- **Critical word first** (primero la palabra crítica). Fuerza al siguiente nivel de memoria a enviar primero el dato requerido por el procesador seguido del resto del bloque.

Aunque la primera técnica pueda parecer peor a simple vista, en la práctica suele ser mucho más frecuente encontrarla en los diseños comerciales por el hecho de ser la única que tiene en cuenta las múltiples estrategias que las memorias habilitan en la actualidad para enviar datos consecutivos por grupos o ráfagas. Las [secciones 10.13.4.2](#) y [10.13.5.1](#) nos descubren este tipo de estrategias en el contexto de las memorias SDRAM y DDRAM, respectivamente. uso comercial
☛ p.60/Vol.2
☛ p.69/Vol.2

4.3.5 Caché víctima

Se trata de incorporar una diminuta caché L2 de unas pocas líneas, normalmente entre 8 y 16, que actúa de repositorio donde se le va a dar una oportunidad de permanecer a los **descartes** más recientes que tenga que realizar la caché L1, en lugar de enviarlos directamente al destierro que supone su expeditivo alojamiento en memoria principal. segunda
oportunidad



Ejemplo 3.11: LA CACHÉ VÍCTIMA DEL RENOVADO K7

Con la llegada de las 0.18 micras al microprocesador K7, AMD decidió mejorar la eficiencia de su sistema de cachés integradas incorporando un búfer o caché víctima con capacidad para albergar 8 líneas de caché (cuyo tamaño por cierto aumentó de 32 bytes a 64 bytes respecto a los primeros K7).

Se habilitó así un espacio intermedio de 512 bytes que restaba una media de 3 ciclos de penalización en el caso de no encontrar el dato solicitado en el primer nivel de caché.

4.3.6 Caché de tercer nivel (L3)

Adicionalmente a los niveles de memoria descritos hasta la caché L2, podríamos plantearnos la incorporación de una **caché de tercer nivel (L3)**, en los casos en los que viniese integrada en una placa base que a su vez acoplara un microprocesador con memoria interna hasta el nivel L2.

Sin embargo, no parece que este tercer nivel de caché vaya a introducirse en el interior del microprocesador a corto plazo: Fijándonos en la forma en que emergen los sucesivos niveles de memoria a lo largo de la historia, podemos inferir que la aparición de un nivel nuevo se encuentra asociado al paulatino distanciamiento de sus niveles anterior y posterior, hasta crear un vacío intermedio lo suficientemente amplio como para que sea ocupado por este nuevo nivel. Y en el contexto actual de la evolución de las memorias, no está sucediendo esto. Repasemos los puntos clave:

L2 crece

① La L2 interna está creciendo bastante en tamaño, y aunque la memoria principal es previsible que también lo haga, la distancia se sigue manteniendo en el mismo orden de magnitud. Esto reduce la dependencia que el procesador tenía del exterior.

MP y bus local
más veloces
p.73/Vol.2
p.165/Vol.2

② La memoria principal está ganando en velocidad, con sucesivas mejoras a 166 y 200 MHz y diseños que tienen a su alcance frecuencias muy superiores (ver [secciones 10.13.6](#) y [13.3.3](#)). Con ella, la frecuencia del bus local va también en progresivo aumento. Bajo esta situación, una hipotética caché L3 externa se encontraría bastante más próxima en velocidad a la memoria principal que a la caché L2 interna, justo lo contrario de lo que sería deseable para justificar su presencia.

L1 y L2
progresan

③ La brecha en velocidad que abre el procesador respecto a la memoria no es ahora como lo fue antaño, puesto que arrastra con él a sus cachés L1 y L2. Esto retrasa la creación del hueco que la caché L3 necesita para instalarse cómodamente en la arquitectura PC.

esquema similar

Sea como fuere, el funcionamiento de un microprocesador con tres niveles de caché es muy similar al descrito para dos niveles, prevaleciendo todas las ideas referentes a mayor tamaño y menor velocidad conforme nos alejamos del núcleo del procesador, y habilitando una nueva vía que comunicaría la caché L2 con la L3, dejando el bus frontal (frontside) para las comunicaciones externas con memoria principal.

Las posibilidades para adoptar operaciones concurrentes serían ahora aún mayores, pero si hemos visto un cauce segmentado de trece etapas cuando abordábamos un diseño concurrente

con dos niveles, debemos presuponer que para tres niveles sobrepasaríamos fácilmente las veinte etapas, y no resulta nada fácil controlar un cauce de semejante profundidad sin que la complejidad de la circuitería resultante desborde nuestro presupuesto.

Proximidad al núcleo del procesador

◀ 4.4

Antes de nada, conviene tener claro a qué nos referimos con “caché interna (o externa) al microprocesador”, ya que existen diferentes acepciones y formas de entender las cosas que se confunden más a menudo de lo que sería deseable.

- | | |
|---|---------------------|
| <p>❶ Desde un punto de vista comercial, la frontera entre lo externo e interno se encuentra delimitada por lo que nos suministra el vendedor cuando lo adquirimos (esto es, lo que viene <i>de serie</i> con él).</p> | comercial |
| <p>Bajo esta perspectiva, todas las cachés L1 desde el 80486 serían internas, mientras que para las L2 serían internas las del Pentium Pro Pentium II, CeleronA, Pentium III y Pentium 4, y todos los Xeon e Itanium por parte de Intel, y las del K6-III y K7 por parte de AMD, mientras que serían externas las del Pentium, MMX, K5, K6 y K6-2.</p> | ejemplos |
| <p>❷ Desde una vertiente funcional, podemos entender como <i>interno</i> todo aquello que fuese capaz de transmitir los datos al procesador siguiendo el mismo ritmo de su frecuencia de reloj.</p> | funcional |
| <p>Según este criterio, sólo una pequeña parte de las cachés internas desde el punto de vista comercial lo serían atendiendo a criterios de funcionalidad. Concretamente, seguirían siéndolo todas las cachés L1, mientras que para las L2, permanecerían en este grupo todas las del Pentium Pro, Celeron y Pentium III a 0.18 micras, Xeon, K6-III y sólo algunas configuraciones selectas del Itanium y K7.</p> | ejemplos |
| <p>❸ Desde el punto de vista de la integración, <i>interno</i> sería todo aquello que cohabita en el mismo chip de silicio de la CPU.</p> | integración |
| <p>Esto mantendría como internas todas las cachés de primer nivel, mientras que descartaría un gran número de cachés de segundo nivel, dejando tan sólo la de 128 Kbytes del CeleronA y la de 256 Kbytes de los Pentium III y 4 (todos en su versión de 0.18 micras) por parte de Intel, y la de 256 Kbytes del K6-III y los K6-2+ y K7 de 0.18 micras por parte de AMD.</p> | ejemplos |
| <p>Todos los Pentium Pro, II y las versiones Katmai del Pentium III integran la L2 en un segundo chip, y de igual forma proceden los K7 Athlon de 0.25 micras de AMD; además, a excepción del Pentium Pro, que integra conjuntamente el área de datos y el controlador, los Pentium II, III y K7 aprovechan su formato de cartucho cerámico para escindir el controlador de caché en un tercer circuito integrado que se coloca entre el chip del procesador y el chip del área de datos de caché.</p> | |
| <p>La razón por la que se integran estos elementos en chips aparte en cuanto el conjunto alcanza un tamaño de cierta consideración tiene una clara justificación en el coste asociado a su fabricación: Uno de los principios básicos del diseño de circuitos integrados cuantifica el coste de integración de un circuito como una función de la cuarta potencia del parámetro “área de integración”.</p> | función de coste |
| <p>Por lo tanto, sobrepasado un umbral en el que aún tiene cabida la L1, la fabricación en un sólo área es ocho veces más cara que la variante consistente en integrar dos áreas con la mitad de superficie, una para L2 y otra para el microprocesador. No debe extrañarnos por ello que los modelos de microprocesador que evolucionan fusionando la caché L2 en el mismo chip del procesador suelen recortar el tamaño de ésta, e incluso esperar para hacer el cambio con la transición hacia distancias de integración más pequeñas.</p> | ocho veces más cara |



FOTO 3.3: (a) El microprocesador K7 de 0.18 micras, que cuenta ya con una caché L2 integrada de 256 Kbytes. (b) Detalle del área de integración donde se encuentra esta caché.

Aunque desde una perspectiva rigurosa la mejor definición de caché interna a un microprocesador es la que se encuentra integrada de forma indivisible en su interior, somos conscientes de que el usuario percibe como interno a su microprocesador todo aquello que le suministran dentro del producto que adquiere en la tienda, así que para evitar inconsistencias, hemos adoptado la siguiente **nomenclatura**:

nomenclatura

- externa** ❶ Será **externa** toda caché ubicada en la placa base y por tanto ajena a lo que es la adquisición del procesador. Cuando llegemos a la descripción de las características de cada procesador, ésta en concreto vendrá simbolizada mediante el símbolo ✕.
- interna** ❷ Será **interna** toda caché que venga con el procesador cuando lo compramos pero que se encuentre colocada en un chip aparte. Distinguiremos esta característica en lo sucesivo con el símbolo ✓.
- integrada** ❸ Será **integrada** toda caché incluida con el procesador e integrada en el mismo chip de la propia CPU. Este rasgo lo denotaremos mediante el símbolo ☆. La [foto 3.3](#) muestra el aspecto de un K7 de 0.18 micras y un detalle de su área de silicio en la que se encuentra integrada una caché L2 de 256 Kbytes.

🏠🏠 **Analogía 3.3: LA UBICACIÓN DEL PAPEL DE LA FOTOCOPIADORA COMO MEDIO PARA RECONOCER CADA TIPO DE CACHÉ**

Continuemos con esa visión de la caché L2 como un almacén contiguo del que proveer con millones de folios a esa fotocopidora que era nuestro procesador. Si ese almacén fuese una ampliación del habitáculo donde está la máquina, estaríamos ante lo que es una caché integrada: Compartiendo el mismo espacio físico que el procesador y la caché L1, y sólo ligeramente más lenta que ésta, por no encontrarse tan a mano.

Proveedor de papel en una máquina fotocopidora	Proveedor de información en un procesador actual
Bandeja de carga de folios	Banco de registros
Torre de folios apilados siempre en el habitáculo de la máquina	Caché L1 siempre integrada junto al procesador
Almacén junto a las dependencias de la máquina	Caché L2 integrada en el procesador
Almacén en el mismo edificio de nuestra empresa	Caché L2 interna adquirida con el procesador
Papelería ajena de ámbito local	Caché L2 externa
Proveedor en Almería	Memoria principal

TABLA 3.8: Resumen del conjunto de analogías existentes entre una fotocopidora como procesador y el papel para copias como los datos necesarios para la ejecución de sus instrucciones.

La ubicación del almacén en el mismo edificio de nuestra empresa es el símil de la caché interna: Hay que salir del habitáculo, pero a otra dependencia de nuestra propiedad, y aunque el viaje es ya un poco molesto, aún podemos regresar con cierta presteza. Finalmente, el no disponer de almacén y buscar el proveedor en una papelería local sería el caso de contar con una caché externa: Ya hay que utilizar la vía urbana compartida con otros componentes (placa base) y acudir a un ente ajeno (chip adquirido separadamente), con lo que el retraso comienza a ser bastante grande, pero en cualquier caso, siempre será mejor que desplazarse a Almería a por el papel (tomar el dato de memoria principal).

En la [tabla 3.8](#) resumimos todas las similitudes encontradas entre el tándem fotocopidora-papel y procesador-caché. Por otro lado, la [tabla 3.9](#) utiliza la nomenclatura propuesta para clasificar todos los modelos comerciales de quinta y sexta generación con objeto de que vayamos familiarizándonos con ella al tiempo que afianzamos los conocimientos sobre las cachés de nuestros procesadores. Esta tabla constituye además un magnífico compendio que resume la tendencia del mercado en los últimos cinco años:

- ❶ **Caché L1: Siempre integrada.** Desde hace ya algún tiempo, asume el rol de banco de registros extendido para el procesador, y nadie cuestiona ya su ubicación.
- ❷ **Caché L2 externa.** Difícilmente vamos a verla ya. El acceso a memoria principal de una DDRAM o RDRAM actual se encuentra más penalizado por la conexión a través del bus que por la latencia del dispositivo, y una caché externa sólo consigue enjugar ésta última. En otras palabras: Flaco aliado es un bus lento para una caché en la que la velocidad es su razón de ser. Ni siquiera los nuevos buses de 400 y 533 MHz hacen que este sombrío panorama cambie, ya que en ellos el procesador es también mucho más rápido, y lo que cuenta aquí es la diferencia entre ambos. Además, la L2 ya no tiene razón para quedarse fuera del procesador, pues la evolución de la tecnología de integración juega a su favor.
- ❸ **Caché L2 interna.** Los procesadores en formato Slot supusieron su consolidación, al llevar un zócalo cuya razón de ser era precisamente la colocación de una L2 interna sin incurrir en un coste excesivo. Ha quedado demostrado que aquella solución sólo era una estación temporal en el camino de la L2 hacia la integración conjunta con el procesador, y no esperamos una vuelta atrás: El camino ha venido marcado por la reducción de las micras, y ésta es calle de una sola dirección.
- ❹ **Caché L2 integrada.** Aunque alguna marca ya se atrevió con ella en la época de las 0.25 micras, la L2 se integra en el procesador con la llegada de las 0.18 micras. No esperamos

← pág. 84

incuestionable

en extinción

coyuntural

consolidada

Caché de primer nivel ó L1		Caché de tercer nivel ó L3	
Integrada en todos los procesadores de Intel y AMD a partir del Pentium		Interna en aquellos en los que está presente, que son únicamente el Cascades y Foster de Intel, esto es, el Pentium III Xeon y el Pentium 4 Xeon de 0.18 micras	

	Tipo de caché de segundo nivel ó L2 y su símbolo asociado		
	Externa - X (todos en formato Socket)	Interna - ✓ (todos en formato Slot)	Integrada - ✱ (todos en Socket)
I N T E L	Pentium Pentium MMX Celeron (0.35)	CeleronA (0.25) Pentium II Pentium II Xeon Pentium III (0.25) P III Xeon (0.25)	CeleronA (0.18) Pentium III (0.18) P III Xeon(*) (0.18) Pentium 4 Pentium 4 Xeon(*)
L	En general, todos los de 0.35 micras	En general, todos los de 0.25 micras	En general, todos los de 0.18 micras
A M D	K6 K6-2	K7 (0.25)	K6-2+, K6-III, K6-III+ K7 (0.18), Duron

TABLA 3.9: Clasificación del tipo de caché que incorporan todos los modelos comerciales de microprocesadores de quinta y sexta generación. Entre paréntesis, el valor de la distancia de integración cuando actúa como factor discriminante al respecto. Vemos que Intel lo ha tenido bastante claro en relación a cómo fabricar las L2 de sus procesadores en función de su tecnología de fabricación. También existe una correlación clara con el tipo de zócalo utilizado, salvo en los dos casos señalados con(*), donde los Xeon se desmarcan en formato Slot debido a la presencia de la L3 interna. En negrita, los procesadores cuya L2 funciona a la misma velocidad del procesador.

que se mueva de aquí, puesto que en una arquitectura actual es su ubicación natural. Con la llegada de las 0.13 micras, aprovecha además para consolidar su posición y aumentar de tamaño.

en extinción

⑤ **Caché L3 externa.** En vías de extinción por razones muy similares a las esgrimidas para la L2 externa.

sólo para servidores

⑥ **Caché L3 interna.** Apenas sí hemos visto esta modalidad hasta la fecha. Tiene sus opciones para consolidarse en aquellas arquitecturas en las que esquivar el bus local sea casi una obligación. Un claro ejemplo son las versiones Xeon de los Pentium III y 4, sistemas concebidos para que múltiples procesadores se acoplen en una única placa base que comparte la vía de acceso a memoria. En todos los casos, se utilizará para ella una solución basada en zócalo Slot como ya ocurrió con la L2.

inédita

⑦ **Caché L3 integrada.** Aún inédita. Conforme la tecnología de integración avance y veamos chips con más de cien millones de transistores en la segunda mitad de la década, probablemente algún fabricante se anime a gastar parte de este ingente patrimonio en integrar una L3. En este caso, la L3 repetiría el mismo peregrinar ya realizado por la L2. No obstante, volvemos a recalcar que nos parece mejor opción decantarse por aumentar los tamaños de la L1 y la L2.

relación con los buses
pág. 85

Establecidas las tres posibilidades para la ubicación de la caché, debemos también clarificar la conexión al procesador para cada una de ellas. La [tabla 3.10](#) sintetiza la correspondencia entre las tres modalidades de caché vistas y los diferentes buses relacionados con el procesador que ya conocemos.

Modalidad de caché en relación al procesador	Ubicación en el sistema	Denominación(es) de su conexión al procesador
Externa	En placa base	Bus local o bus frontal (frontside bus)
Interna	En un segundo chip junto al de la CPU	Bus trasero (backside bus)
Integrada	Como parte del propio chip de la CPU	Conexión interna

TABLA 3.10: Interrelación entre los diferentes tipos de caché, su ubicación en la arquitectura del sistema y el bus que las liga al microprocesador.

Ubicación del controlador de caché

4.5

Muchas veces, cuando describimos el procesador, solemos fijarnos en su Unidad de Proceso, que es donde realmente se efectúan las operaciones, y pocas veces recaemos en la Unidad de Control. Con la memoria caché ocurre algo similar: Describimos su tamaño, su velocidad, sus líneas, ... en definitiva, todo lo que conforma su área de datos, sin reparar en que también existe un área de control desde donde se gobiernan todas las operaciones.

área de control

La parte de control más importante de una caché es su **directorio caché**, que es donde se consultan las etiquetas de memoria principal para determinar si un dato buscado se encuentra allí o no, y a partir de ahí, obtener su dirección de acceso.

directorio
caché

Las posibilidades de ubicación de una memoria caché dentro del conjunto del sistema se completan con una eventual separación de las áreas de datos y control, pudiéndose situar el controlador en tres emplazamientos diferentes:

- ❶ Fuera del procesador, en un chip aparte junto con su área de datos. VARIANTE 1
- ❷ Dentro del procesador, pero gestionando los datos de una caché ubicada en el exterior. VARIANTE 2
- ❸ Dentro del procesador, integrado junto con los datos que controla. VARIANTE 3

A continuación estudiaremos por separado estas tres posibilidades, que al haber sido enumeradas en orden cronológico, irán apareciendo en el recorrido temporal que vamos a efectuar.

En los primeros PC, cuando la caché era considerada un artículo de lujo, algunas placas base habilitaban un zócalo donde opcionalmente se podía incorporar una pequeña placa de circuito impreso con la caché y su controlador si se disponía del dinero suficiente para hacer frente a su coste. El aspecto de esta placa es muy parecido al de los módulos de memoria principal actuales, tal y como se aprecia en la [foto 10.2](#).

zócalo opcional

← p. 27/Vol. 2

Posteriormente, la caché se hizo imprescindible y las ventas masivas y la competencia en el sector encargado de su fabricación le hicieron perder ese elitismo. Cuando la caché comenzó a montarse de serie en la placa base, el controlador de caché se incluía de forma independiente en la geografía de la placa base, junto al circuito integrado que albergaba las celdas de datos.

popularización

VARIANTE 1

Con posterioridad se instaló en la arquitectura de las placas base una corriente que trató de reducir espacio y minimizar el número de chips. Apareció así el concepto de juego de chips, o serie de circuitos integrados que aglutinan multitud de controladores que antes se encontraban diseminados por la placa base en chips independientes (DMA, interrupciones, temporización, ...).

juego de chips

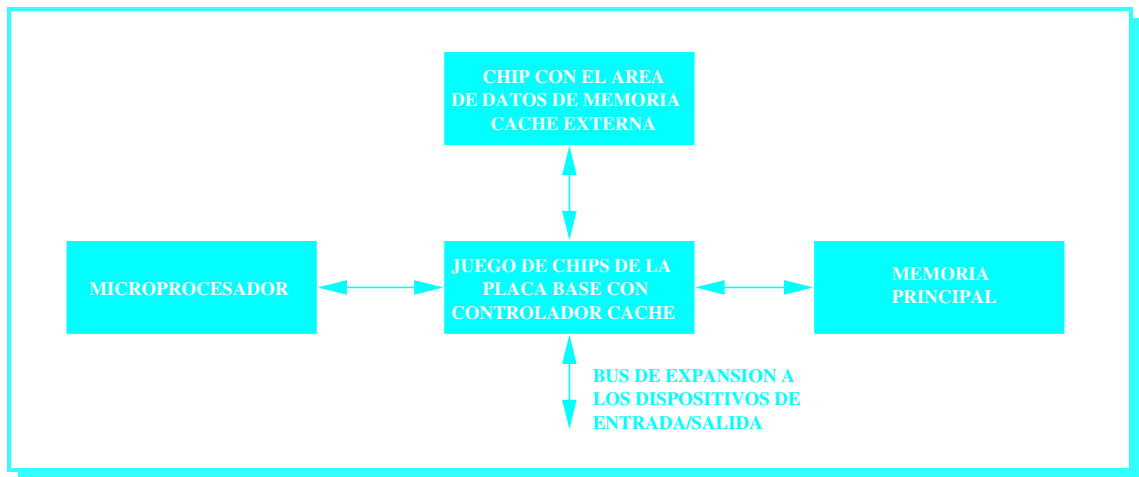


FIGURA 3.12: Ubicación del área de datos de caché y su controlador en el contexto de una placa base de quinta y sexta generación.

absorción El controlador para la caché L2 no fue una excepción, y también fue absorbido por este juego de chips central. El área de datos se mantuvo en sus chips dedicados independientes, y como el fabricante de la placa base es el responsable de montar ambos componentes, no existe conflicto de compatibilidad alguno. El resultado es el esquema que aparece en la [figura 3.12](#).

VARIANTE 2 La situación anterior conduce a una importante conclusión: No es el microprocesador el que dialoga con la caché externa de forma directa, sino el juego de chips donde se ubica el controlador, y que hace de puente entre ambos. Esto provocó más adelante que cuando el microprocesador comenzara a absorber a la caché externa, en ocasiones no pudiese abarcar tanto el controlador como el área de datos, decantándose por integrar el primero, que es con quien realmente dialoga, y dejando fuera al segundo. Esto permite al procesador averiguar de forma local si el dato que se busca se encuentra o no en la caché externa, con lo que saldremos al bus únicamente cuando estemos seguros de encontrar el dato buscado, minimizando su uso.

descoordinación Los microprocesadores que implementan esta aproximación son escasos, ya que dependen en exceso de la bondad del fabricante de la placa base: Si éste decide incorporar una caché externa, debe saber que el controlador está ubicado en el microprocesador, y por tanto limitar el área de datos al rango que éste puede gestionar. Por otro lado, si el fabricante de la placa base decide sacar productos de gama más baja que carecen de caché, el microprocesador contendrá un controlador de caché al que no vamos a sacar provecho pero que hemos pagado religiosamente.

Ejemplo 3.12: CONTROLADOR DE CACHÉ L2 INTEGRADO, ÁREA DE DATOS EXTERNA

En el ámbito de los RISC de gama alta, encontramos algunos ejemplos de este tipo de configuraciones en la primera mitad de la década de los noventa. Quizá el más significativo sea el de la familia del microprocesador R10000 de Silicon Graphics, que evoluciona pasando por el R4000 con sendas cachés L1 internas de 16 Kbytes y ausencia de caché L2, el R8000, con idénticas cachés L1 pero que ya incorpora el mencionado controlador interno para la L2, y los R5000 y R10000, en los que se duplica la capacidad del primer nivel de caché manteniendo internamente el controlador para la caché L2 externa.

**Ejemplo 3.13: CONTROLADOR DE CACHÉ L2 INTEGRADO, ÁREA DE DATOS INTERNA**

El mejor ejemplo que tenemos de esta configuración es el K7 de 0.25 micras, que incorporó una caché L2 de 512 Kbytes interna en un chip separado de la CPU pero dentro del cartucho cerámico del procesador. El directorio caché L2 estaba integrado en el chip de la CPU, y daba cobertura completa a este área de datos. Si posteriormente se optaba por aumentar el tamaño del chip de la caché L2, entonces este directorio caché proporcionaba sólo una parte de las etiquetas de dirección necesarias, teniéndose que colocar la parte sobrante en el chip de datos de la caché.

Resultaba así una implementación flexible y eficiente al mismo tiempo. En la práctica, lo de distribuir el directorio caché L2 en dos porciones hubiera dado lugar a un algoritmo de consulta de dos niveles, actuando el primero en el chip de la CPU y eventualmente el segundo en el chip de la caché, pero no llegó a utilizarse porque el K7 nunca llegó a superar el tamaño de 512 Kbytes de caché L2 en su versión de 0.25 micras.

En los diseños en los que se opta por incorporar la L2 conjuntamente con su área de datos de forma integrada en el propio chip procesador, lo que se plantea es si además se incorpora el controlador de la caché L3. Aquí también disponemos de algunos ejemplos.

VARIANTE 3

**Ejemplo 3.14: CONTROLADOR DE CACHÉ L3 INTEGRADO, ÁREA DE DATOS EXTERNA**

Los ejemplos comerciales de esta variante se encuentran, en la primera mitad de los años 90, con alguna versión puntual de la familia de procesadores Alpha de Digital, y en la segunda mitad de los noventa, con el K6-III de AMD.

Este último caso representó de lo poco malo que tenía aquel procesador, ya que se apostó por encontrar una L3 en la placa base que muy pocos fabricantes de ésta realmente incorporaron (por entonces, la idea de la L3 ya se encontraba en franco declive). Paradójicamente, fueron las placas base para K6-2, que sí llevaban caché externa (hacia las veces de L2 para ese procesador), las que, al compartir el zócalo Super 7 con el del K6-III permitieron a éste utilizar la L2 como L3.

Finalmente, llegamos a la situación actual, en la que todo este caos intermedio ha quedado despejado: La L2 está toda integrada dentro del chip del microprocesador, y la L3, de asomarse, lo hace en su versión interna donde el chip dedicado a ella también integra conjuntamente datos y controlador.

situación
actual

4.6 ▶ Velocidad

Estamos ante otro de los parámetros que más confusión genera en una caché por la ligereza con que se trata. La publicidad suele proclamar “Microprocesador XXX con caché a la misma velocidad del procesador”. Inocente frase, pero difícilmente más ambigua:

- tipo
 - ❶ Primera omisión: El tipo de caché de que se trata. Si ésta es integrada, la información acerca de la velocidad de caché es superflua (toda caché integrada en el mismo chip procesador funciona a su misma velocidad). Del contexto de la frase hemos de suponer que se trata de una caché interna, pero no siempre es así, y cada vez lo va a ser menos.
- nivel
 - ❷ Segunda omisión: El nivel de caché de que se trata. Con buena voluntad, pensamos que la frase debe estar refiriéndose a la L2, y esta vez sí tenemos una elevada probabilidad de acertar.

otra falacia más
 No cuesta tanto incorporar una letrita y un numerito a la susodicha frase para evitar confusiones, pero es que todo esto esconde una negligencia aún mayor: La de desconocer el significado real de la frase. La publicidad señala la velocidad de la caché como único responsable de su rendimiento, pero es otra falacia más del mercado.

pregunta 1
 pág. 91 ➡
 Un usuario de Pentium 4 ó de K6-III podría preguntarse: “Mi caché L1 funciona a la misma velocidad del procesador, y mi L2 también, ¿Qué diferencia existe entre que mi procesador recoja el dato de la primera o que lo haga de la segunda?”. La respuesta lógica es *ninguna*; la respuesta válida, *otra bien distinta*, que será desvelada tras el [ejemplo 3.19](#).

pregunta 2
 pág. 90 ➡
 Un usuario de Pentium III ó K7 de 0.25 micras podría también haberse formulado la siguiente pregunta cuando salió la versión de 0.18 micras de su procesador: “Si mi L2 funciona a la mitad de la velocidad del procesador, ¿Significa esto que mi procesador tarda el doble en obtener el dato de ella con respecto a los nuevos modelos de 0.18 micras en los que la caché L2 funciona a la misma velocidad del procesador?”. La respuesta lógica es *afirmativa*; la respuesta válida, *otra bien distinta*, que será desvelada tras el [ejemplo 3.17](#).

información oculta
 La explicación a todo esto comienza por desvelar que sólo se está proporcionando al usuario una parte marginal de la información que necesita para conocer el rendimiento real de su memoria caché. Se dice que (a) la caché responde a cierta velocidad, pero se omite que (b) antes de que la caché responda, debe llegarle la petición del procesador, y que (c) una vez devuelto el dato solicitado, éste, y todos los que le acompañan en su misma línea de caché, deben emprender todo el camino de regreso hacia el procesador.

cinco configuraciones
 Para aprender a relativizar la importancia que tiene la velocidad de una caché frente al resto de parámetros aquí estudiados, analizaremos a continuación cinco configuraciones comerciales distintas en las que hemos cubierto un amplio espectro de variantes. Eso nos permitirá seguir un criterio certero en la identificación de los modelos de microprocesador más ventajosos cuando presentemos sus características en nuestra cobertura generacional de capítulos posteriores.

4.7 ▶ Análisis del rendimiento de caché en relación al procesador

pág. 91 ➡
 Con objeto de realizar un análisis del rendimiento que sea mínimamente comprensible y didáctico, no queda más remedio que simplificar suponiendo peticiones aisladas a memoria. No vamos a tener en cuenta el alto grado de concurrencia que tiene lugar entre los buses, las distintas cachés y el procesador, y que han sido el recurso utilizado por algunas de las optimizaciones vistas, pero aún con estas limitaciones podremos ordenar muchas de las estrategias en relación al potencial de mejora que revierten sobre el sistema de una forma bastante realista. Cuando lleguemos al [ejemplo 3.18](#), comprobaremos que efectivamente nuestro análisis apenas difiere de los

resultados hechos públicos por AMD para el rendimiento de sus cachés comerciales actuales.



Ejemplo 3.15: CASO 1: BÚSQUEDA DE UN DATO EN LA CACHÉ L2 EXTERNA DEL K6 (L2 A LA MITAD DE LA VELOCIDAD DEL PROCESADOR)

Comenzaremos con el primer K6 que vió la luz, aunque el ejemplo es también válido para su versión más reciente de 0.25 micras y para el K6-2 (no ya para el K6-2 de 0.18 micras, que goza de una pequeña L2 integrada). Los parámetros que completan su configuración de memoria son los siguientes:

- Tamaño de línea L2: 32 bytes.
- Tamaño de línea L1: 32 bytes.
- Anchura del bus local que en este caso conecta L1 y L2: 64 bits (8 bytes).

Si en la obtención del dato de una instrucción se tarda un ciclo en el acceso directo al banco de registros, en el acceso a memoria L2 se tardarían los siguientes:

- Ciclo 1: Conversión de la dirección relativa a dirección virtual (el direccionamiento relativo es el más utilizado en las instrucciones de acceso a memoria).
- Ciclo 2: Traducción virtual a física en L1 de datos (todos los procesadores emiten direcciones virtuales que hay que mapear sobre el espacio de direcciones físico de cada usuario en función de la cantidad de memoria de que disponga).

La **TLB (Translation Look-Aside Buffer)** es la unidad funcional responsable de retener las traducciones más recientes (funcionando como una caché cuyos datos son direcciones físicas, y si se acierta en el acceso a la misma, la traducción tiene lugar en un solo ciclo; con objeto de simplificar nuestro estudio, supondremos que siempre va a ser así).

- Ciclo 3: Selección de la línea de caché L1 a sustituir y emisión de la petición al chip L2 por el controlador del bus local.
- Ciclo 4: El controlador del bus local arbitra la obtención de este recurso compartido. Supongamos que gana el bus en el ciclo 10. Mientras tanto, ya se ha actualizado la etiqueta en la línea de caché L1 donde se alojará el dato cuando se obtenga.
- Ciclos 10, 11, 12 y 13: Envío de la dirección al controlador de la caché L2 ubicado en el juego de chips. Suponemos que el bus local funciona con un divisor de 4 respecto al procesador (esto es, 100 MHz para el bus local y 400 MHz para el K6-2 según la configuración más vendida de este procesador).
- Ciclo 14: Traducción virtual a física en L2 (TLB).
- Ciclo 15: Consulta en directorio caché de L2, obteniéndose la dirección de caché en la que se encuentra el dato de memoria principal.
- Ciclos 16 y 17: Tiempo de respuesta de la caché L2 (dos ciclos por el hecho de que la caché responda a la mitad de velocidad del procesador; éste es el único paso que se ve afectado por la velocidad de la caché).
- Ciclo 18: Comienza la arbitración del bus local para el viaje de regreso. Suponemos que se conserva la apropiación de este recurso.
- Ciclos 19 al 34: Transporte de la línea de 32 bytes hacia la L1 (4 viajes de 8 bytes por el bus local, gastándose 4 ciclos para cada viaje).
- Ciclo 35: Almacenamiento en L1 de la línea procedente de L2 (parte de esta operación se solapa con los últimos viajes por el bus).
- Ciclo 36: Envío al procesador de la palabra que solicitó.

Conclusión: Cuando el bus local entra en juego, es el responsable de la mayor parte de la penalización en el acceso a memoria caché dada su lentitud y arbitración. No nos extrañe pues la paulatina extinción de las cachés externas del mercado.

 **Ejemplo 3.16: CASO 2: BÚSQUEDA DE UN DATO EN LA CACHÉ L2 INTERNA DEL K7 (L2 A 1/3 DE LA VELOCIDAD DEL PROCESADOR)**

Estamos ante el primer Athlon que salió al mercado, el de 0.25 más orientado al segmento doméstico. Estos son los parámetros que completan su configuración de memoria:

- Tamaño de línea L2: 64 bytes.
- Tamaño de línea L1: 32 bytes.
- Anchura del bus trasero que conecta L1 y L2: 64 bits (8 bytes).

Y ésta, su operativa de funcionamiento:

- ▶ Ciclo 1: Cálculo de la dirección virtual procedente del direccionamiento relativo.
- ▶ Ciclo 2: Traducción virtual a física en L1 de datos (TLB).
- ▶ Ciclo 3: Consulta en directorio caché de L1. El dato no está.
- ▶ Ciclo 4: Selección de línea víctima en L1 y emisión de la petición a L2 por el bus trasero.
- ▶ Ciclo 5: En L1, actualización de la etiqueta en la línea de caché donde se alojará el dato una vez obtenido. En L2, traducción virtual a física (TLB).
- ▶ Ciclo 6: Consulta en directorio caché de L2. Obtención de la dirección para el dato.
- ▶ Ciclos 7, 8 y 9: Tiempo de respuesta de la caché L2.
- ▶ Ciclos 10, 11, 12 y 13: Transporte de la línea de 32 bytes hacia la L1 (4 viajes de 8 bytes por el bus trasero).
- ▶ Ciclo 14: Almacenamiento en L1 de la línea procedente de L2.

- ▶ Ciclo 15: Envío al procesador de la palabra solicitada (el K7 dispone de la estrategia *critical word first* para enviar primero la palabra de la línea que ha solicitado el procesador).

Conclusión: En el caso 1, con una caché externa más rápida se gastan más del doble de ciclos que en el caso 2 con una caché interna más lenta. Se pone de manifiesto la importancia del tipo de caché frente a lo anecdótico de su velocidad.

 **Ejemplo 3.17: CASO 3: BÚSQUEDA DE UN DATO EN LA CACHÉ L2 INTERNA DEL K7 (L2 A LA MISMA VELOCIDAD DEL PROCESADOR)**

Se trata de la versión del Athlon de 0.25 micras que fue candidata en su tiempo al segmento de servidores, incorporándole una caché L2 DDR SRAM que se sincronizaba a la velocidad del microprocesador, entonces por los 600 MHz. Los modelos de Athlon que se vendieron bajo este aditivo fueron muy escasos debido a su elevado coste.

En relación a su caché L2, el comportamiento de este procesador es mimético al de todos los modelos de Pentium II/III Xeon de Intel: La caché es más grande y esta vez funciona a la misma velocidad del procesador, aunque sigue siendo interna.

Esta variante ahorra los ciclos 8 y 9 anteriores, reduciendo el montante total a sólo 13 ciclos.

Conclusión: Parecía que el acceso a L2 en el caso 3 se aceleraría en un factor de 3 en relación al caso 2 (esta era la respuesta lógica a nuestra pregunta 2 de la sección anterior). Sin embargo, la respuesta válida es una aceleración de apenas un 13%. Dejando a un lado otro tipo de mejoras, la velocidad no es un parámetro tan decisivo como sugiere a primera vista.

**Ejemplo 3.18: CASO 4: BÚSQUEDA DE UN DATO EN LA CACHÉ L2 DEL K7 (L2 INTEGRADA)**

Este procesador es el Thunderbird, el K7 de 0.18 micras comercializado por AMD bajo formato SocketA. Al integrarse la L2 dentro del chip CPU, se prescindió del bus trasero y se instaló un puerto de conexión L1-L2 de 256 bits.

Como resultado, la operativa del caso 2 conlleva ahora sólo 10 ciclos, pues se siguen ahorrando los ciclos 8 y 9 del caso 3 debido a la menor latencia del dispositivo, y además, se ahorran los ciclos 11, 12 y 13 puesto que la comunicación de la línea completa puede ahora completarse en un solo ciclo.

Según AMD, bajo la ejecución de aplicaciones comerciales, el tiempo medio que su procesador K7 de 0.18 micras tarda en tomar un dato en el caso de no encontrarlo en la caché L1 y sí en la L2 puede oscilar entre 11 y 20 ciclos dependiendo de la actividad del procesador, esto es, considerando todas las variantes de flujos concurrentes que pueden darse. En la gran mayoría de los casos se sitúa en el valor mínimo de 11 ciclos, al que se le restarían tres ciclos más por la actuación de una caché víctima compuesta de 8 líneas de caché que se sitúa entre la L1 y la L2 como aditivo que la compañía incorporó en la transición a 0.18 micras. Nosotros no hemos considerado la intervención de esta nueva caché aquí porque restaría limpieza a la comparativa que tratamos de ilustrar, en la que sólo intervienen L1 y L2.

Conclusión: Cuando aparentemente parecía que no lograríamos mejoras en relación a la configuración del caso 3 (pues su velocidad era la misma), la ganancia ha resultado ser del 23%. De nuevo tenemos ante nosotros una prueba inequívoca de la importancia del tipo de caché, aunque eso sí, el salto en rendimiento desde caché externa a interna es muy superior al que se produce desde caché interna a integrada.

**Ejemplo 3.19: CASO 5: BÚSQUEDA DE UN DATO EN CACHÉ L1**

Este caso es similar en todos los procesadores, y su operativa de funcionamiento, muy sencilla:

- ▶ Ciclo 1: Conversión de la dirección relativa de memoria a dirección virtual.
- ▶ Ciclo 2: Traducción de la dirección virtual a física en L1 de datos (se utiliza su TLB).
- ▶ Ciclo 3: Consulta en directorio caché de L1. Encontramos el dato y lo enviamos al procesador.

Conclusión: Creíamos que daba igual traer el dato de la L2 que de la L1 simplemente porque ambas iban igual de rápido (esta era la respuesta lógica a la que fué nuestra pregunta 1 en la sección anterior). Acabamos de ver la importancia del nivel de caché: La aceleración del caso 5 respecto al caso 4 es superior a un factor 3.

Aprovecharemos para recalcar que si el dato es servido por el banco de registros en lugar de

Modelo de microprocesador	Memoria caché analizada			Bus que entra en juego	Número ciclos
	Nivel	Tipo	Velocidad		
K6 ó K6-2	L2	Externa	La mitad	Bus local	36
K7 0.25 micras	L2	Interna	Un tercio	Bus trasero	15
K7 0.25 micras	L2	Interna	La misma	Bus trasero	13
K7 0.18 micras	L2	Integrada	La misma	Ninguno	10
K7 0.18 micras	L1	Integrada	La misma	Ninguno	3

TABLA 3.11: Comparativa de rendimiento de cinco configuraciones distintas de memoria caché escogidas según su distinta relación con el microprocesador.

por la caché L1, la aceleración también es superior a un factor 3, pues aunque ambos respondan igual de rápido, la dirección de acceso al banco de registros se obtiene de forma directa a partir del código de operación de la instrucción, ahorrándose los dos primeros ciclos de traducción de la operativa anterior (y un eventual tercer ciclo si fallamos en el acceso a la TLB y hemos de realizar la traducción de forma manual).

Curiosa circunstancia: La respuesta lógica que dimos a la pregunta 1 de la sección anterior estuvo muy cerca de ser la respuesta válida a la pregunta 2, y la respuesta lógica de la pregunta 2, muy cerca también de convertirse en la respuesta válida para la pregunta 1. Menudo trabalenguas hemos compuesto. Es el signo más evidente de lo traicioneras que pueden resultar las cosas de la caché si no son escudriñadas con esmero. Pasemos a limpiar las conclusiones obtenidas, pues condensan todo un recital didáctico.

La tabla 3.11 resume los resultados de los cinco casos analizados, y el balance que éstos arrojan en la interacción del procesador con la caché nos sirve para ordenar los parámetros estudiados según su influencia en el rendimiento. El orden de mayor a menor importancia es el siguiente:

respuestas con
trabalenguas

RÁNKING FINAL:

- nivel ❶ El nivel de caché: Su posición en la jerarquía.
- tipo ❷ El tipo de caché dentro de un mismo nivel: Externa frente a interna incide más que interna frente a integrada.
- bus ❸ El bus en el tipo de caché externa ó interna: Incide más su velocidad en el primer caso, y su anchura en el segundo, pero en ambos casos, tanto en el camino de ida como en el de vuelta.
- velocidad ❹ La velocidad de la caché respecto a la del procesador (influye sólo en el acceso).
- tamaño línea ❺ El tamaño de la línea de caché (influye sólo en el transporte de vuelta).
- tamaño TLB ❻ El tamaño de la TLB (influye en la traducción de ida).

labores ocultas

El directorio caché, la TLB y el bus trasero son elementos que casi nadie suele tomar en consideración. Sin embargo, aunque no tengan en su mano mejoras porcentuales de tres dígitos, sí realizan una labor fundamental en el conjunto del sistema caché que conviene destacar:

- dir. caché TLB
 - El directorio caché y la TLB, porque son los responsables de localizar un dato en caché a partir de una dirección que no es la suya, sino la de memoria principal. A la caché se le obliga a ser ultrarrápida al tiempo que transparente al resto del sistema, así que nadie puede facilitarle información más útil que aquella destinada a otras partes del sistema.
- bus
 - La velocidad del bus que conecta la caché y el procesador, que puede oscilar entre 1, 2 ó 3 en las diferentes implementaciones de caché L2 interna incluso dentro de un mismo modelo de microprocesador, originando así diferentes alternativas de coste y rendimiento de cara al usuario sin modificar un ápice la arquitectura interna del procesador.

		Pentium clásico	Pentium Pro L2 256 Kbytes	Pentium Pro L2 512 Kbytes
Fecha lanzamiento		Marzo, 1993	Marzo, 1995	Junio, 1995
Patillaje		296 pines	387 pines	387 pines
Número de chips		1 (CPU+L1)	2 (Uno con CPU+L1, otro con L2)	
Zócalo		Socket 5 y 7	Socket 8	Socket 9
Número de transistores	Bloque CPU	2.1M	4.5M	4.5M
	Bloque L1	1M	1M	1M
	Chip L2	Externo	15.5M	31M
Espacio físico ocupado	Chip CPU+L1	91 mm ²	306 mm ²	195 mm ²
	Chip L2	Externo	202 mm ²	242 mm ²
	Encapsulado	25 cm ² (**)	35 cm ²	35 cm ²
Integración L2 (micras)		SRAM (0.8) (*)	CSRAM (0.6)	CSRAM (0.35)
Velocidad caché L2		Asíncrona (*)	Síncrona con el procesador	
Coste inicial del conjunto		935 dólares (**)	1200 dólares	1600 dólares

TABLA 3.12: Los principales parámetros relativos a la integración de las cachés L1 y L2 en los dos modelos de Intel más representativos a este respecto durante la pasada década de los 90. Hemos incluido numerosos parámetros del chip CPU en sí para poder comparar sus valores con los de las cachés. (*) = Estos datos hacen referencia a una caché L2 externa de 256 Kbytes incorporada en la placa base. (**) = Datos sin contar la caché L2 de la placa base, para cuyo tamaño de 256 Kbytes estimamos unas dimensiones de 250 mm² y un coste de unos 100 dólares de aquellas fechas.

Análisis del coste asociado a una caché

◀ 4.8

4.8.1 Caché interna

La gama más alta, compuesta por los modelos con caché sincronizada a la velocidad del microprocesador, es la más cara. Por ello, si son modelos orientados al segmento doméstico, tendrán un tamaño reducido con objeto de que el coste no se dispare (128 Kbytes en la L2 del primer Celeron y 512 Kbytes en el primer Athlon, por ejemplo).

tamaños
domésticos

Para tamaños superiores, hay que buscar en la L3 y en procesadores del segmento servidor. Por ejemplo, los Xeon de Intel disponen de configuraciones de este tipo hasta los 8 Mbytes, pero el coste de la configuración básica, que comienza en 1 Mbyte, es ya superior a los 3000€.

tamaños para
servidor

La [tabla 3.12](#) compara el coste de dos configuraciones de tipo servidor frente a otra de corte doméstico para los primeros sistemas servidores de 1995. El coste que se muestra se expresa en dólares de aquella época, es decir, sin actualizar por el efecto de la inflación. Tratamos de comparar los tres tipos de caché (externa, interna e integrada) y los dos primeros niveles (L1 y L2), y para encontrar un sistema real con tantas variantes, no queda más remedio que remontarse un poco atrás en el tiempo. En nuestra defensa, diremos que la tecnología de caché avanza a un ritmo más lento que el procesador, y los precios y prestaciones de la tabla no están tan lejos de la situación actual del mercado como podría pensarse.

comparativa

El directorio caché es también un ingrediente de particular relevancia en la formación de precios. La presteza con la que éste tiene que llevar a cabo la búsqueda del dato solicitado le obliga a utilizar una memoria asociativa para alojar las etiquetas, de forma que todas ellas puedan ser consultadas simultáneamente en el mismo ciclo de reloj. El problema de esta memoria asociativa es que su coste es exponencial con el tamaño, por lo que un directorio caché el doble de grande resulta cuatro veces más caro.

memoria
asociativa

4.8.2 Caché integrada

tamaños
domésticos

El tamaño de las cachés integradas está entre los 32 y los 64 Kbytes para la L1, y entre los 256 y los 512 Kbytes para la L2 a 0.13 micras (Northwood, Barton), y ya en 1 Mbyte para la L2 a 0.09 micras (K8).

En cualquiera de estos casos, la caché L2 se llevará, ella sola, más de la mitad de los transistores del chip microprocesador, de los que buena parte de ellos se encontrarán en su controlador ó directorio caché.

Ahora bien, en las áreas de integración de los procesadores, algunas que ya hemos mostrado y otras que irán desfilando más adelante, puede observarse que en ningún caso una L2 interna ocupa más de la mitad del área de silicio. Esto es así por dos razones básicas:

silicio

- 1 La celda básica del área de datos de caché que almacena un bit está optimizada para ser integrada con seis transistores proporcionalmente más pequeños que los de otras unidades funcionales del procesador, y por lo tanto, ocupa bastante menos espacio en silicio.

metal

- 2 En segundo lugar, el directorio caché contiene fundamentalmente conexiones de metal. En realidad, la densidad de este retículo metálico es tan grande, que no pocos fabricantes habilitan sabiamente la superficie de silicio que queda despejada por debajo de él para colocar circuitería de otras unidades funcionales del procesador. En estos casos, resulta injusto atribuir al cableado el espacio ocupado en lugar de a estas unidades extra allí ubicadas.

dos frenos

Pero tampoco podemos aspirar a cachés integradas muy grandes aunque estemos decididos a asumir el elevado coste que supone en transistores y silicio. Porque el área de silicio influye exponencialmente en el coste de integración, pero también incide en la velocidad del conjunto: Los retardos de las señales eléctricas en el interior de un chip grande suponen en la práctica uno de los frenos más claros para la frecuencia del procesador.

un remedio

Estas dos razones explican que el máximo tamaño de caché integrada haya estado históricamente condicionado por la tecnología, pues una distancia de integración más corta pone remedio a esos dos obstáculos: Primero, hace que el transistor resultante ocupe menor área de silicio, y segundo, permite disfrutar de mayor velocidad de conmutación.

varios ejemplos

Así, durante la época de las 0.35 micras no vimos a la caché L2 integrada en el procesador, y ya durante las 0.25 micras empezaron a asomar los primeros modelos, como el K6-III. Con la llegada de las 0.18 micras, fue ya una práctica generalizada a todos los modelos existentes.

SECCIÓN 3.5

Conjunto de instrucciones

personalidad

El conjunto de instrucciones máquina que es capaz de entender un procesador es un parámetro clave para entender su diseño, y condiciona lo que podríamos catalogar como su personalidad. Esta sintetiza cuatro aspectos básicos:

abstracción

- 1 El nivel de abstracción con que se le proporcionan las instrucciones. En un mayor nivel de abstracción, sólo diríamos qué queremos hacer, encontrándose las instrucciones más cercanas a nuestro lenguaje natural. Un menor nivel de abstracción aboga en cambio por un mayor nivel de detalle, una visión más cercana a la circuitería en la que ya se dice cómo se ejecutan las instrucciones en su arquitectura interna, trascendiendo los aspectos de su diseño a la capa software de más bajo nivel.

Hace unos años, esta capa podía ser el propio usuario si éste era capaz de fajarse al nivel del lenguaje ensamblador de la máquina. En la actualidad, los programadores que están dispuestos a hacer así las cosas son una especie en vías de extinción, pues se busca cada vez más simplificar el desarrollo de los programas frente a la consecución de unos puntos porcentuales de rendimiento extra.

- ② La amigabilidad del interfaz. Un mayor nivel de abstracción debería facilitar el diseño de un interfaz más amigable, más cómodo al usuario, aunque en la práctica no ha sucedido así: El conjunto de instrucciones 80x86 es uno de los que mayor nivel de abstracción presenta, y sin embargo, parece un lenguaje diseñado por el peor de nuestros enemigos. En cambio, ciertos diseños RISC, cuyo nivel de abstracción es siempre bajo, presentan un lenguaje tremendamente sencillo de manejar aprovechándose de su simpleza. amigabilidad

- ③ La rapidez de asimilación, o cómo de rápido decodifica e interpreta las instrucciones que le llegan. Si todo el software se escribiera para el procesador sobre el que va a ser ejecutado, estaríamos hablando siempre de una única operación de decodificación que gastaría un solo ciclo, pero muchos programas ejecutables son compilados para una plataforma anterior a otra que los reutiliza para garantizar la compatibilidad con las aplicaciones software ya existentes en el mercado. asimilación

En el mundo del PC, esta historia nos es muy familiar, pues llevamos veinte años ejecutando en nuestros procesadores código escrito para el procesador 8086. El cómo lleve internamente a cabo el procesador esta labor de conversión al que es su código nativo es un bastión nada despreciable en su rendimiento.

- ④ La riqueza del lenguaje, o el arte de diseñar un conjunto de instrucciones que responda a lo que los programadores desean ejecutar en la máquina. En este sentido, el lenguaje máquina evoluciona de la mano de las aplicaciones, y ahí está la retahíla de conjuntos de instrucciones multimedia que han emergido en los últimos cinco años al calor de la fiebre por las aplicaciones gráficas, de sonido e Internet. riqueza

Remontándonos atrás en el tiempo, la historia ha sido **pendular** respecto al comportamiento del conjunto de instrucciones. Comienza con una primera fase que data de finales de los años 70 y principios de los 80 en la que el conjunto de instrucciones va engordando paulatinamente, y a finales de los 80 invierte su tendencia y evoluciona en sentido opuesto hacia conjuntos de instrucciones cada vez más simples. Ultimamente, la tendencia parece invertirse de nuevo, con la llegada de las instrucciones multimedia que amplían el conjunto de instrucciones del procesador y aumentan su complejidad, como las populares MMX y 3DNow! en quinta generación, sus sucesoras las SSE y Enhanced 3DNow! en sexta generación, o iniciativas más complejas como la VLIW que se enmarca ya dentro de la séptima generación de microprocesadores. historia
pendular

CISC versus RISC

◀ 5.1

Volviendo a los orígenes, diremos que los microprocesadores comienzan su andadura con un repertorio de instrucciones simples. A finales de los años 70 se origina la primera corriente evolutiva hacia diseños de mayor complejidad, la cual vino respaldada por cuatro aspectos básicos:

- ① **El auge de los lenguajes de programación alto nivel.** El programador escribe sus programas en un lenguaje cada vez más potente y alejado del lenguaje ensamblador. La responsabilidad de generar código eficiente ya no es del programador: Se ha trasladado hacia el compilador. compilador
- ② **La aparición de familias de microprocesadores.** Los fabricantes utilizan una estrategia de marketing en la que cada nuevo microprocesador es compatible con el anterior, pero mejorado con nuevas características, lo que supone ampliar su conjunto de instrucciones y complicar su circuitería. compatibilidad

velocidad ③ **La migración de funciones desde el software hacia el hardware**, motivada por la ganancia en velocidad que la implementación hardware de una instrucción proporciona frente a su homóloga software.

empaquetado ④ **La lentitud de la memoria respecto al procesador**. Esto ralentiza la fase de búsqueda de una instrucción, por lo que se trata de empaquetar muchas instrucciones en una sola con el fin de minimizar el número de operaciones de búsqueda necesarias para completar un programa.

CISC De esta manera, el procesador va incorporando cada vez más modos de direccionamiento de operandos, más funciones potentes y especializadas en tareas concretas, y más registros de propósito general, surgiendo el diseño CISC, o de *conjunto de intrucciones complejo* (del inglés, *Complex Instruction Set Computer*). Este diseño se caracteriza por una extensa circuitería, sobre la que la capa software ha delegado parte de sus funciones. Un buen ejemplo es la saga de microprocesadores 80x86 de Intel.

Pero la tendencia CISC se rompe a finales de los 80, con la llegada de nuevos personajes que cambian el trasfondo de la situación:

- ① Aparecen las memorias caché, provocando una drástica disminución del tiempo de búsqueda de una instrucción y posibilitando así una eventual descomposición de las instrucciones en otras más sencillas.
- ② Se alcanza un punto en el cual la incorporación de nuevas instrucciones proporciona una funcionalidad cada vez más rebuscada, que apenas puede ser aprovechada por el compilador, y que por el contrario complica el diseño del microprocesador, haciéndolo cada vez más lento y costoso.
- ③ Atendiendo a las necesidades de los programas más populares, se demuestra que el código máquina de éstas contiene un aplastante predominio de instrucciones sencillas.

RISC Estos tres motivos van a provocar, en primer lugar, un freno a la ampliación del conjunto de instrucciones de un procesador, y, posteriormente, su paulatina disminución. Se eliminan así aquellas instrucciones más complejas que puedan implementarse mediante otras más simples, lo que poco a poco desemboca en una filosofía de diseño tipo RISC, o de conjunto de instrucciones reducido (del inglés, *Reduced Instruction Set Computer*). Así, la responsabilidad de obtener una ejecución rápida se traslada de nuevo a las capas software del sistema, como el compilador y el sistema operativo.

diseños híbridos Los microprocesadores que aquí estudiaremos se encuentran en una extraña confluencia entre las corrientes CISC y RISC. Disponen de ciertos rasgos RISC que cada vez tratan de acentuarse más, pero no pueden considerarse como tales debido a su obligada compatibilidad con diseños CISC de la familia Pentium a la que tributan vasallaje.

dos escuelas Sea como fuere, hemos de admitir CISC y RISC como una dualidad más en el diseño de computadores, cada una de ellas con sus ventajas y con sus carencias, y serán siempre factores exógenos los que sobreponderen las ventajas de uno frente a las de otro, provocando un desplazamiento del mercado en esa dirección.

un solo objetivo A continuación vamos a comparar estas dos grandes escuelas, y de paso demostraremos cómo, desde perspectivas enfrentadas, ambas persiguen un mismo objetivo: Minimizar el tiempo que un microprocesador invierte en la ejecución de un programa.

tres factores: Este tiempo puede obtenerse como el producto de tres factores:

- NI ① NI = Número de instrucciones máquina en que se descompone el programa fuente.

- ② CPI = Número medio de ciclos de reloj que se necesitan para ejecutar cada una de las instrucciones máquina anteriores. CPI
- ③ T = Tiempo del ciclo de reloj anterior (o su frecuencia F como magnitud inversa). T

Una filosofía de diseño CISC trata de reducir el primero de esos factores, proporcionando para ello instrucciones de muy alto nivel capaces de llevar a cabo operaciones complejas. Por el contrario, un diseño RISC está orientado a minimizar el segundo de los factores. Las dos alternativas tratan de aprovecharse de las mejoras en la tecnología de integración de chips (velocidad de conmutación de los transistores) para reducir al máximo la duración del ciclo de reloj del procesador, el tercero de los factores. CISC: NI y T
RISC: CPI y T

La Unidad de Proceso de un RISC es del tipo carga/almacenamiento, esto es, las operaciones de lectura y escritura a memoria se aíslan del resto y el compilador las trata de forma separada para conseguir un alto grado de concurrencia en la ejecución de instrucciones. En cambio, una arquitectura CISC no puede aislar estas operaciones, al estar presentes en un mayor número de instrucciones. U. Proceso

La Unidad de Control, que en un procesador RISC es cableada, en uno de tipo CISC se implementa de forma microprogramada. Cada instrucción de un CISC tarda una serie de ciclos (entre 4 y 20 aproximadamente), y para cada uno de ellos la Unidad de Control tiene que activar unas señales de control que gobiernen el funcionamiento de la Unidad de Proceso. U. Control

La palabra de control de cada ciclo se almacena en una memoria de microprograma, donde la secuencia de palabras de control perteneciente a cada instrucción se agrupa formando microrutinas. Este diseño facilita la posterior modificación del procesador con un simple cambio en su memoria de microprograma, cualidad que han sabido aprovechar muy bien diseños contemporáneos como el reciente Crusoe de Transmeta. También ha permitido a otros fabricantes como Intel corregir sobre la marcha errores descubiertos en sus modelos con posterioridad a su lanzamiento al mercado, como el archiconocido de la unidad de punto flotante del Pentium. La cara negativa de la memoria de microprograma es que, puesto que la Unidad de Control tiene que esperar a que ésta responda para cada ciclo de ejecución del procesador, su funcionamiento se ralentiza bastante, y además, ocupa bastante área de integración en silicio. memoria de microprograma
versatilidad
ralentización

La necesidad de ejecutar una instrucción por ciclo obliga al RISC a cablear la Unidad de Control primando la velocidad por encima de la versatilidad. Como además disminuye su espacio de integración, tiene en su mano la consecución de frecuencias más elevadas. ventajas RISC

La mayoría de los aspectos negativos de un RISC aparecen precisamente como consecuencia de sus ventajas: La simplicidad de las instrucciones, por ejemplo, provoca que el rendimiento de una máquina RISC dependa mucho de la eficiencia del compilador. Por ello, el tiempo de desarrollo del software para una máquina RISC es potencialmente más elevado que para una CISC. El mayor número de instrucciones máquina que un programa RISC posee también repercute negativamente en el espacio que el programa ejecutable ocupa en memoria. inconv. RISC

La [tabla 3.13](#) muestra una comparativa que resume las principales diferencias entre ambas alternativas de diseño. ← pág. 98


Diseño RISC

El denominador común de un microprocesador avanzado de los años 90 se asienta sobre los principios básicos de la filosofía de diseño RISC. Son los procesadores que se montan en la amplia gama de computadores que existen por encima de los PC: Estaciones de trabajo, servidores, computadores paralelos, ...

En este punto del capítulo, estamos en condiciones de dar un paso adelante para ilustrar cómo trabajan al nivel más ligado al conjunto de instrucciones.

	RISC	CISC
Hardware		
Tiempo de desarrollo	Bajo	Alto
Unidad de Control	Cableada	Microprogramada
Banco de registros	Extenso (256 o más)	Reducido (16, 32)
Espacio de integración	Reducido	Grande
Software		
Tiempo de desarrollo	Alto	Bajo
Compilador	Complejo y fundamental para la eficiencia	Más sencillo y menos crítico
Programa objeto	Muy largo	Compacto (20-30 % menor)
Grado de abstracción del HW	Bajo	Alto
Diseño		
Formato de las instrucciones	Fijo	Variable
Conjunto de Instrucciones	Pequeño (128 o menos)	Grande (200-500)
Modos de direccionamiento	Pocos y simples (hasta 4)	Muchos y Complejos
Características de las instrucciones	Sencillas y rápidas (1 ciclo de duración)	Potentes y lentas (de 4 a 20 ciclos)
Ejemplos comerciales	PowerPC(Motorola/IBM) R10000(SGI), Alpha(DEC)	80x86(Intel) 68000(Motorola)

TABLA 3.13: Características RISC y CISC frente a frente.

 **pág. 58**
 segmentación y
 superescalaridad

En primer lugar, diremos que un diseño tipo RISC favorece la implementación de las estrategias de paralelismo a nivel de instrucción vistas en la [sección 3.3](#): Su reducido conjunto de instrucciones simples hace que todas ellas tengan una duración similar, lo que permite una mejor segmentación y superescalaridad al estar sus etapas de ejecución más compensadas entre sí. Por otro lado, la propia simplicidad del procesador deja espacio de silicio libre para incluir cachés internas, ejecución fuera de orden, extensiones multimedia, y alguna que otra maravilla más.

proceso
 iterativo

El diseño de un procesador RISC transcurre como un proceso iterativo compuesto de dos fases que se realimentan entre sí con el fin de optimizar al máximo el resultado final: La selección del conjunto de instrucciones del procesador, y el diseño de la circuitería sobre la que éstas se ejecutan.

5.2.1 Selección del conjunto de instrucciones

base

Paso 1. Elección del núcleo básico. El proceso de obtención del conjunto de instrucciones del procesador parte de la selección de un núcleo de instrucciones básico, compuesto por instrucciones imprescindibles en cualquier procesador. Para ello se utiliza la experiencia previa que proporcionan los diseños de procesadores anteriores, y por intersección de los conjuntos de instrucciones más populares, llegamos al que será nuestro punto de partida.

ampliación

Paso 2. Selección de candidatos. A partir de ahí, se considera la extensión de este conjunto de instrucciones mínimo con operaciones y modos de direccionamiento candidatos a formar parte de la funcionalidad del procesador. La selección de candidatos se realiza en base al carácter que se le quiera dar al procesador y a parámetros de afinidad y coste.

purga

Paso 3. Criba de candidatos. Los candidatos que hayan pasado todos los filtros anteriores son entonces sometidos a pruebas de rendimiento sobre aplicaciones reales para cuantificar el beneficio que producen cuando el compilador las utiliza para la generación de código. Si la instrucción candidata produce una mejora significativa en la mayoría de códigos testeados, la instrucción

es finalmente incluida en el conjunto de instrucciones. Tanto el porcentaje de mejora como el de aplicaciones sobre las que produce el efecto deseado son parámetros que determinan el grosor del conjunto de instrucciones. A mayores porcentajes, mayor es la criba de candidatos y menor el conjunto de instrucciones, la funcionalidad, y el coste del microprocesador resultante. (Ejemplo: En el diseño del procesador MIPS, una instrucción se admitió si mejoraba en un 1 % el código de al menos el 90 % de los programas que se escogieron para las pruebas).

Paso 4. Completitud. El resultado de todo este proceso es un conjunto de instrucciones en buena sintonía con las necesidades reales de uso de un lenguaje de alto nivel. Cada instrucción es, o bien estructuralmente necesaria (esto es, no puede obtenerse en función de otras ya existentes), o bien ampliamente demandada durante el proceso de compilación de un programa.

Paso 5. Eficiencia. Una vez seleccionado un conjunto de instrucciones simple, debemos ocuparnos del segundo de los objetivos inherentes al diseño RISC: La ejecución de una instrucción por ciclo de reloj. De entre las instrucciones que dificultan este logro, sobresalen las de acceso a memoria y las de salto. A continuación comentaremos las optimizaciones más sobresalientes que un RISC realiza sobre ellas para lograr salirse con la suya.

□ Acceso a memoria: Carga retrasada

Para realizar operaciones con valores almacenados en memoria, tan sólo necesitamos estructuralmente una operación de carga del valor de una posición de memoria en un registro y su operación inversa de almacenamiento (escritura en memoria desde el banco de registros). El resto de operaciones necesitan referirse únicamente al banco de registros para obtener operandos y/o guardar resultados. Por eso se dice que una máquina RISC implementa una arquitectura de carga/almacenamiento. Las principales ventajas de este tratamiento en el acceso a memoria son básicamente tres:

- ① La reducción del número de accesos a memoria. Puesto que se dispone de un gran banco de registros, muchos de los valores requeridos por las instrucciones pueden encontrarse allí, ahorrando un eventual acceso a memoria. Esto permite relajar los requerimientos de ancho de banda entre el procesador y la memoria.
- ② El hecho de que todas las operaciones se realicen con los registros simplifica el conjunto de instrucciones y los modos de direccionamiento necesarios.
- ③ La eliminación de operaciones con memoria posibilita una mejor estrategia de alojamiento de valores en el banco de registros por parte del compilador. Esto termina de optimizar el número de accesos a memoria a la vez que reduce el ratio del número de instrucciones necesarias para llevar a cabo una tarea.

Estos tres factores ponen al alcance la ejecución de una instrucción por ciclo de reloj del procesador. Para los accesos a memoria que sean inevitables (incluido el fallo en caché), el procesador se ralentiza en principio el número de ciclos que la memoria tarde en responder.

Una forma de aprovechar estos ciclos de espera del procesador consiste en utilizar instrucciones de carga retrasada, esto es, redefinir la semántica de la instrucción de carga para que lleve asociada la ejecución de una serie de k instrucciones de relleno de forma inmediatamente consecutiva. Una instrucción de relleno puede ser cualquiera del conjunto de instrucciones siempre que reúna las siguientes dos condiciones: (a) No utilizar como operando el valor que se está trayendo de memoria en la operación de carga anterior, y (b) respetar la secuencia de ejecución de todas aquellas dependencias de datos y control que contenga el programa.

El valor de k o tamaño de la ventana de relleno para una instrucción de carga vendrá determinado por el tiempo de respuesta de la memoria en ciclos del procesador. Los buenos compiladores conocen este valor y se encargan de buscar instrucciones máquina que cumplan las condiciones

sintonía

presteza

arquitectura de
carga/almacen.

tres ventajas:

menos accesos

menos modos

menos
instrucciones

carga retrasada

instrucciones
de rellenoventana de
relleno

de relleno, así como de reestructurar el código objeto para llenar las ventanas de relleno de las instrucciones de carga en la medida de lo posible. Los compiladores son bastante eficientes realizando este tipo de tareas, aunque su porcentaje de éxito será menor cuanto mayor sea el número de dependencias del programa y/o el tamaño de la ventana de relleno. En el peor de los casos, la ventana de relleno se completa con instrucciones NOP (de no operación) que simplemente dejan al procesador inactivo hasta que llegue de memoria el dato con el que ponerse a trabajar.

compiladores
dependencias

□ Instrucciones de salto: Salto retrasado

El principal problema que introducen las instrucciones de salto proviene de su negativo impacto en el cauce segmentado que todo procesador RISC implementa para la ejecución de instrucciones: La dirección de destino del salto normalmente no se conoce hasta la última etapa de segmentación, es decir, una vez la instrucción de salto ha sido buscada y decodificada, se han obtenido sus operandos, y se ha evaluado la condición de salto en la etapa de ejecución.

dirección de
salto

Por tanto, el procesador comienza la etapa de búsqueda de la instrucción que sigue a la del salto cuando ésta se encuentra en su fase terminal de ejecución. Esto produce ciclos en los que tenemos varias unidades funcionales paradas (por ejemplo, las correspondientes a las fases de decodificación, búsqueda de operandos y ejecución).

Es posible aprovechar estos ciclos ociosos utilizando para las instrucciones de salto la misma técnica de ventana de relleno ya utilizada para las instrucciones de carga: Redefiniendo la semántica de las instrucciones de salto con saltos retrasados para que contengan una ventana de relleno de tres instrucciones. Notificando esto al compilador, éste puede buscar instrucciones del programa que puedan ser insertadas en las posiciones de relleno de los saltos y reestructurar el código máquina de forma apropiada para que se aprovechen los ciclos de penalización asociados a la instrucción de salto.

salto retrasado

5.2.2 Soporte software para una arquitectura RISC

Uno de los aspectos más criticados en los diseños RISC es el elevado número de instrucciones máquina en el que tiene que transformarse un programa para ser ejecutado. Dado que esta transformación es responsabilidad del compilador, resulta inevitable ligar la popularidad de los procesadores RISC con las mejoras en las técnicas de compilación.

□ Compilador

Puede decirse que no hay una técnica de compilación específica para un procesador RISC. Los métodos que se presentan a continuación también se aplican con arquitecturas CISC. Sin embargo, la simplicidad de una máquina RISC hace que el compilador encuentre en ella muchas más oportunidades de optimización que en una CISC.

Los compiladores más actuales son el resultado de una evolución en el proceso de traducción de lenguaje de alto nivel a lenguaje máquina. La eficiencia de un compilador se mide básicamente por el tamaño y la velocidad del código objeto que genera. Las técnicas avanzadas de compilación que mejor rendimiento producen en una arquitectura RISC son las siguientes:

- **Planificación de instrucciones:** La primera tarea que se exige a un compilador es la de aprovechar la presencia de instrucciones de carga y salto retrasado en el conjunto de instrucciones del procesador. Para ello, el compilador debe identificar las instrucciones máquina del código que puedan ser utilizadas como instrucciones de relleno y, posteriormente, reorganizar la ejecución del programa para que estas instrucciones cubran las ventanas de relleno de las instrucciones retrasadas.



Ejemplo 3.20: USO DE LA VENTANA DE RELLENO DE UNA INSTRUCCIÓN DE CARGA RETRASADA POR PARTE DEL COMPILADOR

Considerar el programa fuente $C := A + B; F := 10$; transformado en el siguiente programa objeto:

```
Load R1, A
Load R2, B
Add R3, R1, R2      ← Espera la llegada de A y B de memoria
Load R4, 10        ← Instrucción de relleno
```

Un compilador más optimizado generaría la siguiente secuencia de instrucciones:

```
Load R1, A
Load R2, B
Load R4, 10        ← Se ejecuta la instrucción...
Add R3, R1, R2    ← ...mientras se esperan datos de memoria
```



Ejemplo 3.21: USO DE LA VENTANA DE RELLENO DE UNA INSTRUCCIÓN DE SALTO RETRASADO POR PARTE DEL COMPILADOR

Considerar el siguiente programa objeto:

```
Move R1, R2
Move R3, R4        ← Instrucción de relleno
Add R1, R1, 1
Jump R1, 0, A      ← Instrucción de salto retrasado
...
A: Sub R5, R5, 1
```

Un compilador optimizado aprovecharía la ventana de relleno de la instrucción de salto para ejecutar la segunda de las instrucciones justo a continuación de aquella:

```
Move R1, R2
Add R1, R1, 1
Jump R1, 0, A
Move R3, R4        ← Instrucción cambiada de lugar
...
A: Sub R5, R5, 1
```

- **Uso optimizado de los registros:** El compilador emplea registros para almacenar los datos más frecuentemente utilizados, con el fin de minimizar el número de accesos a memoria.



Ejemplo 3.22: OPTIMIZACIÓN EN EL USO DE REGISTROS POR PARTE DEL COMPILADOR

El siguiente fragmento de código objeto:

```
Load R1, B
Load R2, C
Add R3, R1, R2
Store R3, A
```

que ejecuta la sentencia $A := B + C$ podría reducirse a una sola instrucción máquina si los valores de A , B y C residen en los registros del procesador Ra , Rb y Rc respectivamente. Esto es:

```
Add Ra, Rb, Rc
```

- **Eliminación de redundancias:** El compilador busca oportunidades para reutilizar resultados parciales y eliminar así computaciones redundantes.



Ejemplo 3.23: ELIMINACIÓN DE COMPUTACIÓN REDUNDANTE DESDE EL COMPILADOR

Sea el siguiente código objeto, que ejecuta las sentencias $A := B + X * Y$ y $D = C + X * Y$:

```
Mul R1, Rx, Ry
Add Ra, Rb, R1
Mul R2, Rx, Ry
Add Rd, Rc, R2
```

Para computar el valor a almacenar en D , el compilador puede aprovecharse de que el producto ya se computó anteriormente y está aún alojado en $R1$ para tomarlo directamente de allí en lugar de volverlo a computar. Como resultado ahorramos una instrucción. El programa ahora quedaría:

```
Mul R1, Rx, Ry
Add Ra, Rb, R1
Add Rd, Rc, R1
```


- **Optimización de bucles:** El compilador optimiza también las operaciones que aparecen dentro de los bucles de un programa, con el fin de identificar expresiones invariantes y sacarlas fuera de éstos. invariantes
- **Optimización de operaciones:** En ocasiones una misma operación de alto nivel puede llevarse a cabo con distintas instrucciones máquina. En este caso, el compilador debe seleccionar aquella que sea más rápida. Por ejemplo, $A := A + 1$ puede efectuarse a nivel máquina mediante `ADD Ra, Ra, 1` como suma en la ALU, o mediante `INC A`, que es mucho más rápida al tratarse únicamente de un incremento.
- **Eliminación de llamadas a subrutinas:** La ingeniería del software ha propugnado siempre la escritura de programas en estilo procedural, esto es, descomponer el programa en una serie de rutinas y/o procedimientos que encapsulan funciones a las que se llama desde el programa principal. Sin embargo, una instrucción máquina `CALL` (o de llamada a subrutina) resulta muy costosa de ejecutar para un RISC, principalmente por la necesidad de salvar el contexto del programa (el contador de programa, los registros a utilizar por la subrutina, ...) previamente al salto que realiza.

Podemos ahorrarnos estas operaciones efectuando desde el compilador lo que se conoce como **code inlining**, esto es, suprimir la instrucción `CALL` a costa de duplicar literalmente el código de la subrutina en el programa principal cada vez que se llama a ésta. Como resultado, el programa se ejecuta más rápidamente a costa de ocupar un mayor espacio en memoria. code inlining

Esta estrategia de compilación está teniendo una popularidad creciente que se sustenta en el hecho de que hoy día el tiempo de ejecución de un programa es un parámetro más prioritario que el espacio que ocupa en memoria. No obstante, aunque el *inlining* lo soportan muchos compiladores, la mayoría de ellos no lo realiza a no ser que el usuario así se lo indique de forma explícita mediante alguna de las opciones o niveles de compilación disponibles.

□ Sistema Operativo

Mientras los sistemas operativos orientados a máquinas CISC suelen proporcionar un conjunto de servicios muy elaborados, los principios de diseño RISC apuestan más por la calidad que por la cantidad de los servicios prestados, tratando en todo momento de evitar la complejidad salvo en casos plenamente justificados. Se favorece así a las operaciones que son más utilizadas, proporcionando una buena velocidad de operación a través de controles mínimos y simples. minimalista

Algunos de los mecanismos que los diseños RISC utilizan a nivel de sistema operativo para aumentar el rendimiento de una máquina sin añadir una complejidad excesiva a su hardware son los siguientes: recursos

- **Búfer para la traducción de direcciones virtuales a físicas (TLB):** Agilizar esta traducción que tiene lugar por cada operación de acceso a memoria resulta esencial para la implementación de un potente sistema operativo. Aunque la TLB no es un mecanismo exclusivo de los procesadores RISC, sí es cierto que su reducido espacio de integración permite que la TLB pueda ser integrada dentro del propio chip o extenderse a lo largo de un espacio mayor de silicio, ahorrando el tiempo que se pierde para transferir la dirección virtual a una TLB externa en el primer caso, y reduciendo el riesgo de no encontrar la traducción en la TLB en el segundo. TLB
- **Mecanismos de protección:** Los sistemas operativos utilizan modos de funcionamiento que restringen el acceso del usuario a ciertas partes delicadas del sistema que son gestionadas en exclusiva por parte del sistema operativo. Frente a los múltiples modos y mecanismos de

protección que se suministran en una arquitectura CISC, los RISC proporcionan un control que normalmente se limita a la simple distinción entre modo usuario y supervisor.

- **Gestión de interrupciones:** La gran mayoría de los eventos externos al procesador son gestionados por éste a través de mecanismos de interrupción. Muchos procesadores CISC proporcionan controladores hardware dedicados a la gestión de interrupciones (como el PIC 8259 usado en la familia de los Intel 80x86) para salvar una gran cantidad de información de estado del procesador y generar la dirección del vector de interrupción al que transferir el control en respuesta a la interrupción. Esto añade complejidad hardware, pero no necesariamente simplifica la tarea del sistema operativo. Por ejemplo, muchos sistemas operativos no usan los diferentes vectores de interrupción, sino que en su lugar ejecutan un manejador de interrupciones común a todas ellas que determina de forma precisa las necesidades de procesamiento de la interrupción y la información de estado del procesador que se necesita salvar.

vectores de
interrupción

5.3 ▶ Diseños VLIW

En el diseño de los nuevos conjuntos de instrucciones como el IA-64 de Intel para su Itanium y el x86-64 de AMD para su K8, se ha retomado un concepto que data de comienzos de los años 80: El **VLIW (Very Long Instruction Word)**.

origen
compilador

Este concepto emerge en un contexto histórico muy particular, al calor de los primeros resultados que arrojan un balance favorable en la capacidad de un compilador para identificar las oportunidades de ejecución simultánea que esconde un programa secuencial. Se trata así de que el compilador conecte directamente con alguna(s) de las formas de paralelismo a nivel de instrucción descritas en la [sección 3.3](#).

pág. 58

información
privilegiada

El compilador puede ser capaz de generar un código en el que se da por hecha la presencia de, por ejemplo, tres unidades funcionales de suma (superescalaridad), desgranando operaciones para cada una de ellas de forma explícita en el código de la instrucción. También puede conocer la presencia de, por ejemplo, diez etapas de ejecución segmentadas, así como la incidencia que tienen las dependencias en el código que pasa por sus manos, siendo (supuestamente) capaz de analizarlo y transmitir esta información en el propio formato de instrucción máquina.

pág. 105

idea

pág. 60

La [figura 3.13](#) muestra la idea que hay detrás de una filosofía de ejecución de instrucciones en una arquitectura VLIW. Podemos contrastarla con la [figura 3.5](#), en la que mostrábamos la ejecución segmentada y superescalar utilizada en todos los diseños de la quinta y sexta generación de microprocesadores.

el pionero

El primer diseñador que apostó por esta idea fue Josh Fisher en su proyecto ELI (Univ. Yale - 1981), lo que derivó en la iniciativa empresarial Multiflow, fundada por él en 1984. Según unas fuentes, Multiflow vendió más de 100 multiprocesadores, algunos hasta con 28 microprocesadores trabajando en paralelo. Según otras, sólo se vendió una máquina que tampoco terminó de funcionar del todo bien. El caso es que económicamente la idea resultó un fiasco, y la empresa cerró en 1990 tras serios problemas financieros.

ambición

Pero una cosa es el mundo mercantil, y otra muy distinta el ingenieril. La historia de la computación está llena de proyectos sabiamente concebidos que no tuvieron calado en el mercado y otros a los que éste apadrinó aún no se sabe cómo. Desconocemos si la implementación física de aquella máquina estaba o no a la altura de la idea, pero hay algo que sí cargaríamos en el debe de Fisher: Su excesiva pretenciosidad. El concepto VLIW vale para aplicarlo a cuatro o seis caminos de ejecución independientes (tres ha sido el número escogido por Intel y HP en el IA-64 de su Itanium, y cuatro el seleccionado por Transmeta en su Crusoe), pero colocar un formato de instrucción de muchos cientos de bits y tratar de coordinar con él hasta 28 caminos de ejecución de forma simultánea cuando el software continúa conceptualmente varado en una ejecución secuencial, parece un sueño demasiado bonito como para hacerse realidad sin contratiempo alguno.

utopía

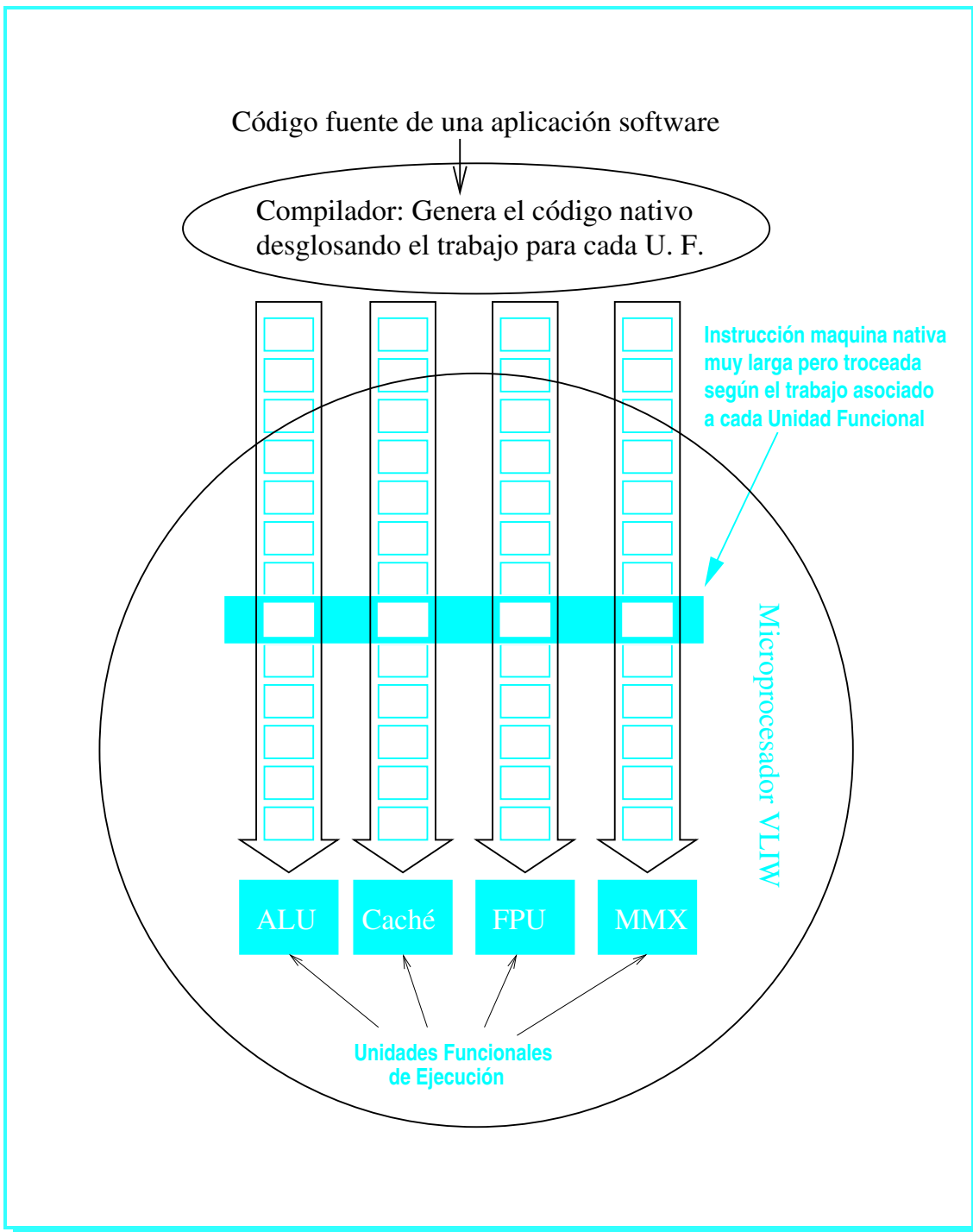


FIGURA 3.13: La filosofía de diseño VLIW (Very Long Instruction Word). Se asume un código máquina muy largo compuesto de compartimentos estancos dedicados a cada unidad funcional existente en el hardware, y el responsable de este desglose es el compilador en la capa software del sistema.

En general, la aspiración de una arquitectura VLIW consiste en crear una máquina con muchas unidades funcionales que puedan trabajar de forma simultánea, cada una según le indica un segmento del formato de instrucción. De este hecho podemos deducir que la instrucción tiene

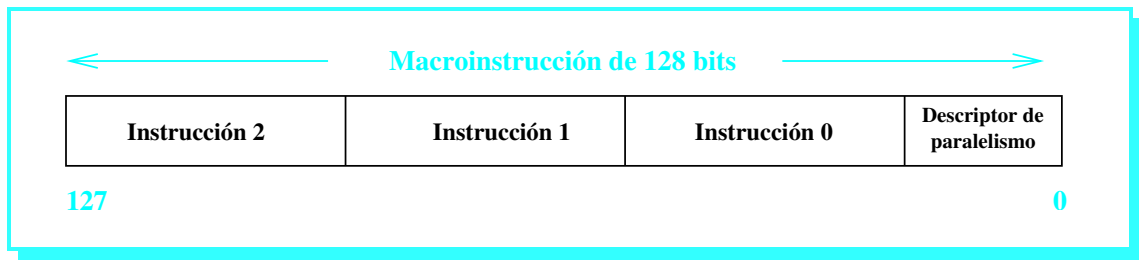


FIGURA 3.14: Formato de instrucción IA-64 de 128 bits correspondiente al microprocesador Itanium de 64 bits. Consta de tres instrucciones máquina agrupadas, junto con un campo adicional que describe el grado de paralelismo existente entre ellas.

calidades

una anchura muy generosa, **calidad** de la que precisamente deriva el nombre VLIW. Se retoma así la máquina microprogramada ancha (claro concepto CISC), en la que las microinstrucciones que controlan la ejecución de una instrucción no son generadas por la unidad de control (como ocurre en los CISC), sino directamente desde el compilador (hecho más ligado a los RISC).

híbrido
RISC/CISC

En realidad, el mejor resumen que podemos hacer de VLIW es que trata de quedarse con lo mejor de los RISC y de los CISC; desgraciadamente, también arrastra de forma compartida algunas de sus carencias. La principal es una mayor dependencia del compilador que las máquinas RISC, lo que ya nos parece excesivo. Digamos que RISC sabe quedarse en un punto de equilibrio: Aquel en el que se le puede sacar más partido al compilador de lo que lo hace el CISC, porque se aprovechan tareas en las que el compilador se desenvuelve con maestría, pero sin llegar a responsabilizarle de otras tareas en las que se encuentra desamparado porque además de no realizarlas tan bien, el programa fuente y la circuitería se encuentran mirando para otro lado.

En el mundo de la informática se han sucedido recientemente dos iniciativas VLIW de notable repercusión, aunque ninguna de ellas tiene como epicentro la arquitectura PC:

Itanium

- El Itanium, antaño conocido como Merced, en el mercado de grandes estaciones de trabajo y servidores. El formato de instrucción para su conjunto de instrucciones IA-64 se adjunta en la [figura 3.14](#).

Crusoe

- El Crusoe de Transmeta orientado al segmento de los portátiles de muy bajo consumo. La [figura 3.15](#) muestra su formato de instrucción, mientras que en la [figura 3.16](#) incluimos un bosquejo de su arquitectura.

[pág. 107](#) ➔
[pág. 107](#) ➔

5.4 ▶ Instrucciones multimedia

revolución

El fenómeno de las instrucciones multimedia ha constituido toda una revolución en lo concerniente al conjunto de instrucciones de un procesador, provocando fuertes repercusiones sobre su arquitectura hardware.

CISC
operandos

La senda descrita por las instrucciones multimedia supone un nuevo acercamiento hacia una filosofía CISC, en tanto en cuanto se apuesta por instrucciones de compleja decodificación que llevan encapsulado en su formato una serie de operandos variable en número y longitud.

propósito
específico

Además, nos encontramos frente a una discontinuidad en el concepto de conjunto de instrucciones para un microprocesador de propósito general, ya que se incluyen instrucciones cuya finalidad está claramente sesgada por las aplicaciones específicas a las que se encuentra orientado.

exigencias del
mercado

La decisión de incorporar instrucciones multimedia al repertorio de instrucciones de un procesador hay que buscarla en las exigencias del mercado. Si la tecnología siempre se ha construido en función de las demandas establecidas por la sociedad a la que sirve, el giro dado por el mer-

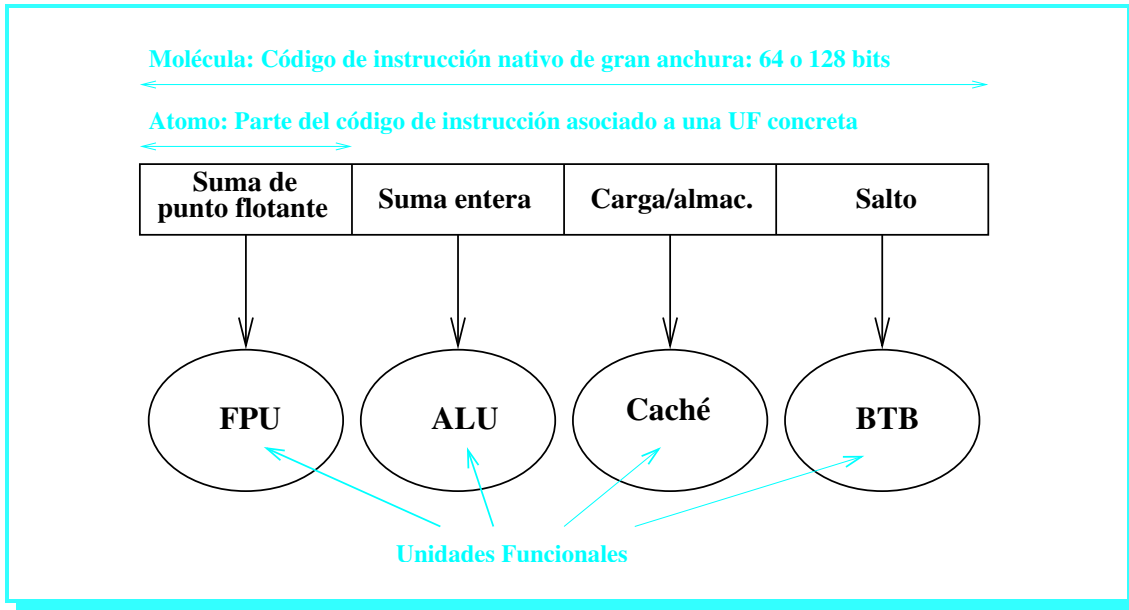


FIGURA 3.15: La filosofía de diseño VLIW (Very Long Instruction Word) aplicada sobre el código de instrucción del microprocesador Crusoe de Transmeta.

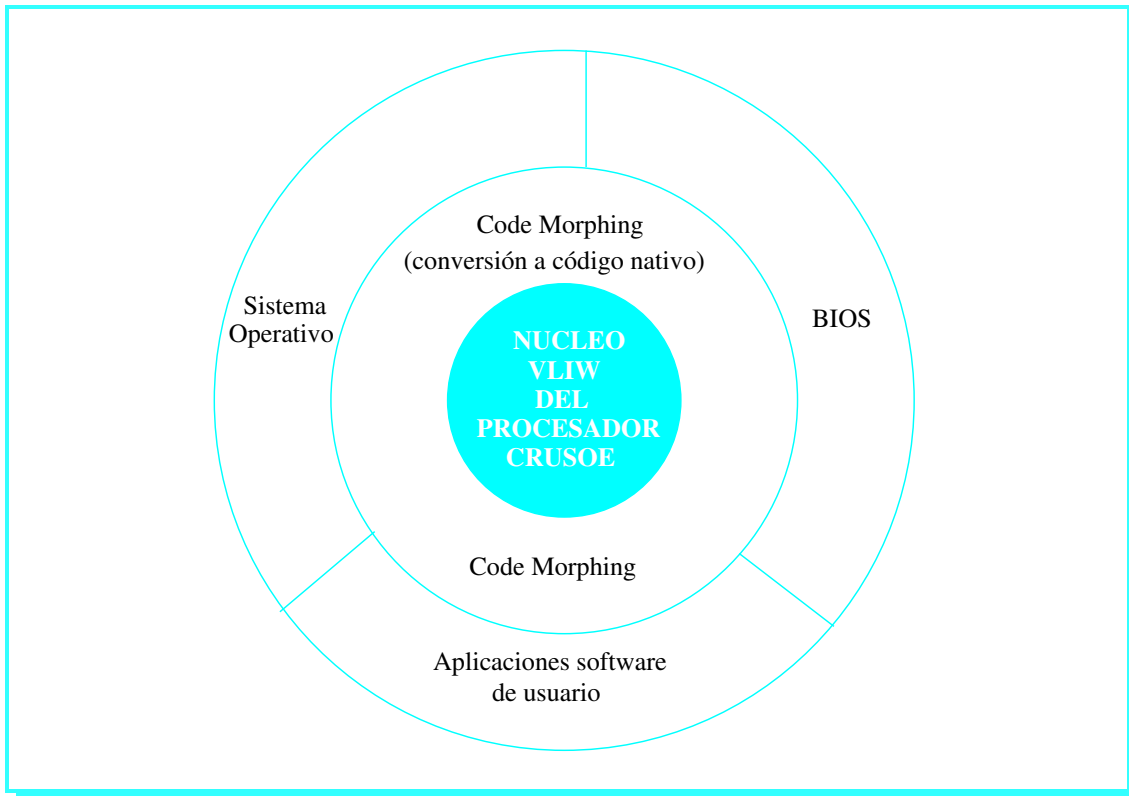


FIGURA 3.16: Composición del procesador Crusoe de Transmeta y relación con la capa software con la que habilita una frontera muy difusa.

cado hacia aplicaciones multimedia (vídeo interactivo, gráficos 3D, animación, sonido, realidad virtual, ...) exigía a los fabricantes de procesadores estar a la altura de las circunstancias.

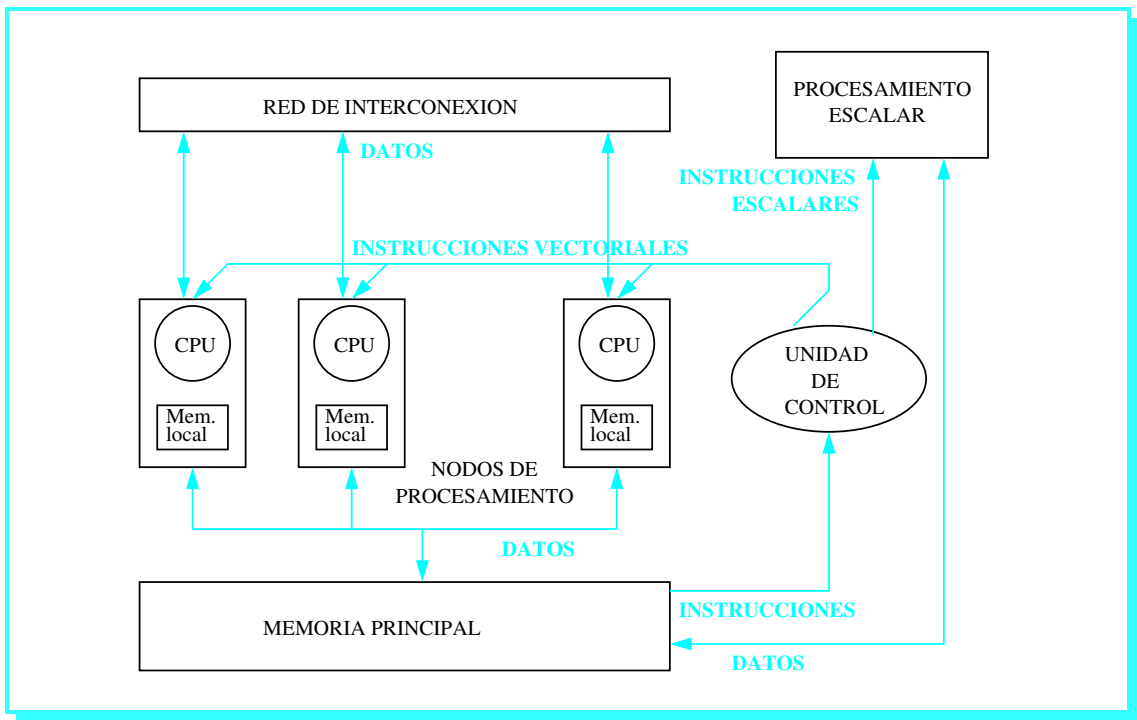


FIGURA 3.17: Diagrama de bloques de un computador SIMD (Simple flujo de Instrucciones, Múltiple flujo de Datos).

5.4.1 El concepto: SIMD

La idea básica sobre la que subyacen todas las instrucciones multimedia parte del concepto SIMD, **originado** en los años 70 en el ámbito de los supercomputadores, esto es, arquitecturas compuestas de múltiples procesadores.

Un computador **SIMD (Simple Instruction Multiple Data)** se compone de un conjunto de nodos de procesamiento y un procesador escalar, todos operando bajo las órdenes de una Unidad de Control común que centraliza el funcionamiento de toda la máquina (ver figura 3.17).

La Unidad de Control busca y decodifica instrucciones de la memoria principal y, dependiendo de su tipo, envía las correspondientes señales de control al procesador escalar o a los nodos de procesamiento para su ejecución. Así, si se trata de una instrucción escalar, sólo funcionará el procesador escalar; en caso contrario, funcionarán todos los nodos de procesamiento en paralelo, los cuales ejecutarán la misma instrucción pero sobre datos diferentes.

La aplicación del concepto SIMD a un solo microprocesador es análoga a la ya comentada. Una única Unidad de Control busca y decodifica las instrucciones convencionales y las SIMD. Cuando llega una instrucción normal, el procesador actúa como siempre, en semejanza con el procesador escalar anterior. En cambio, cuando se trata de una instrucción SIMD, la Unidad de Control envía señales de control a cada una de las unidades en punto flotante, las cuales ejecutan la misma operación pero sobre distintos datos almacenados en sus bancos de registros.

Conceptualmente, SIMD trata de explotar el paralelismo que presenta el conjunto de datos de una aplicación, en contraposición con el paralelismo a nivel de instrucción, donde se paraleliza la secuencia de ejecución de las instrucciones de la aplicación. Por tanto, el rendimiento de un procesador con extensiones SIMD será mucho mayor en programas con abundante cálculo sobre vectores o *arrays* de datos de grandes dimensiones. Por otra parte, el sincronismo de instrucción inherente a la sección SIMD del procesador deja poca flexibilidad para su diseño.

Instrucción	Operandos	Variantes	Duración	Descripción de la operación
Aritméticas				
PADD	B, W, D	3	1	Suma con redondeo
PADDSS	B, W	2	1	Suma con acarreo
PADDUS	B, W	2	1	Suma sin acarreo
PSUB	B, W, D	3	1	Resta con redondeo
PSUBS	B, W	2	1	Resta con acarreo
PSUBUS	B, W	2	1	Resta sin acarreo
PMULHW	HW	1	3	Producto del byte alto
PMULLW	LW	1	3	Producto del byte bajo
PMADDWD	WD	1	3	Explicado en el pie de figura
Comparativas				
PCMPEQ	B, W, D	3	1	Comparación <i>igual-que</i>
PCMPGT	B, W, D	3	1	Comparación <i>mayor-que</i>
Lógicas				
PAND	Q	1	1	Aplican los operandos AND, NAND, OR y XOR resp. sobre el operando fuente y destino, guardando el resultado en este último.
PNAND	Q	1	1	
POR	Q	1	1	
PXOR	Q	1	1	
De conversión				
PACKUSWB	WB	1	1	Empaqueta sin acarreo
PACKSS	WB, DW	2	1	Empaqueta con acarreo
PUNPCKH	BW, WD, DQ	3	1	Desempaqueta datos de mayor peso
PUNPCKL	BW, WD, DQ	3	1	Desempaqueta datos de menor peso
Para el desplazamiento de bits en un registro				
PSLL	W, D, Q	6	1	Desplazamiento lógico a izquierda
PSRL	W, D, Q	6	1	Desplazamiento lógico a derecha
PSRA	W, D	4	1	Desplazamiento aritmético a derecha
Para el movimiento de datos entre registros				
MOV	D, Q	4	1	Transferencia entre registros MMX
Para el cambio de estado de los registros				
EMMS	E	1	1	Pone a cero el byte de estado MMX

TABLA 3.14: El conjunto de instrucciones MMX. Los operandos que acepta cada instrucción van en función de las variantes que admite y su duración. La instrucción PMADDWD multiplica por separado las 4 palabras empaquetadas del operando destino por las cuatro del origen. Los dos resultados de multiplicar las dos parejas de palabras de menor peso se suman, y el resultado se redondea para ser almacenado en la palabra doble de menor peso del operando destino. Con las dos palabras de mayor peso se procede exactamente igual.

5.4.2 El embrión: MMX

El punto de partida en la inclusión de instrucciones multimedia en los microprocesadores para PC fue el conjunto MMX (MultiMedia eXtensions en primera instancia, y luego redefinido por Intel como Matrix Math eXtensions), desarrollado al alimón por Intel y AMD para sus procesadores Pentium MMX y K6, respectivamente.

Con anterioridad a la llegada de las instrucciones MMX, el procesamiento de tipos de datos de 8 o 16 bits en los microprocesadores infrautilizaba los recursos hardware, ya que el ancho de banda para datos en las unidades funcionales de cálculo de los microprocesadores era de 32 o 64 bits, de los que sólo se empleaban los 8 o 16 bits menos significativos.

antecedentes

SIMD

MMX agrupa estos datos en grupos de 64 bits que luego son procesados individual pero concurrentemente mediante la aplicación del concepto SIMD ya comentado, con lo que se aprovechan mejor los recursos de que dispone el microprocesador. Esto, unido a la explotación del paralelismo inherente a la mayoría de algoritmos multimedia, pone al alcance de estas aplicaciones mejoras en velocidad de entre el 50 % y el 100 %.

mejoras en
velocidadindependencia
de la
arquitectura
secuelas

Otra importante característica con que se dota la implementación MMX es la de conservar su independencia de la arquitectura microprogramada, tanto del Pentium como del K6, con el fin de que el juego de instrucciones MMX resultara fácilmente escalable con futuros diseños arquitecturales o frecuencias de reloj más elevadas. Esta virtud se vería ampliamente refrendada con el paso del tiempo, al potenciar la fácil aparición de secuelas que fueron paulatinamente extendiendo el conjunto de instrucciones multimedia original.

57
instrucciones
pág. 109

Este primer conjunto de instrucciones estuvo formado por un total de 57 instrucciones que se resumen en la [tabla 3.14](#). Como podemos apreciar, muchas de ellas son variantes de una misma operación, que puede ser definida sobre diferentes subconjuntos de datos, todos ellos de tipo entero. Las distintas variantes han sido abreviadas de la siguiente forma:

- ❖ B: Byte. 8 operandos de entrada de 8 bits cada uno.
- ❖ W: Word. 4 operandos de entrada de 16 bits cada uno.
- ❖ D: Double word. 2 operandos de 32 bits.
- ❖ Q: Quad word. Un único operando de entrada de 64 bits.
- ❖ WB: Word - Byte. 4 operandos de entrada de 16 bits cada uno y 8 operandos de salida de 8 bits cada uno.
- ❖ DW: Double word - Word. 2 operandos de entrada de 32 bits cada uno y 4 operandos de salida de 16 bits cada uno.
- ❖ QD: Quad Word - Double Word. Un operando de entrada de 64 bits; 2 de salida de 32 bits.
- ❖ BW: Byte - Word. 8 operandos de entrada de 8 bits y 4 de salida de 16 bits.
- ❖ WD: Word - Double Word. 4 operandos de entrada de 16 bits y 2 de salida de 32 bits.
- ❖ DQ: Double Word - Quad Word. 2 operandos de entrada de 32 bits y 1 de salida de 64 bits.
- ❖ HW: High Word. Los 8 bits más significativos de un dato de 16 bits.
- ❖ LW: Low Word. Los 8 bits menos significativos de un dato de 16 bits.
- ❖ E: Etiqueta. El byte de etiqueta que señala para cada registro de punto flotante su uso como tal o como registro MMX. Se utiliza un bit de la etiqueta por cada 8 bits de datos MMX.

5.4.3 Criterios para la selección de instrucciones

Seleccionar una instrucción para que forme parte del repertorio que acepta un procesador es una tarea bastante más peliaguda de lo que a simple vista nos parece. Para que la finalidad última de mejorar el rendimiento se cumpla, las instrucciones elegidas deben satisfacer las tres premisas siguientes:

sencillas

- ❶ Ser sencillas. Se trata de que no desentonen con las ya existentes. Si estamos en un procesador CISC, tendremos algo más de margen, pero si es un RISC, una excesiva complejidad afectará a la velocidad que alcanzaba el procesador sobre las instrucciones antiguas.

- ② Ser utilizadas por una amplia mayoría de aplicaciones multimedia. Se trata de identificar las primitivas más representativas de sus cualidades intrínsecas. Sólo así se amortizará el coste de la circuitería responsable de su ejecución. Una nueva instrucción tampoco es gratuita para el tiempo de decodificación de instrucción, ni para el tiempo que dedica la Unidad de Control a secuenciar todos los eventos en el corazón del microprocesador. Recordemos esa máxima del hardware: *Más grande, más lento*. útiles
- ③ Poder aprovechar las mejoras tecnológicas por venir, principalmente un número creciente de transistores y una mayor frecuencia de reloj. optimizables

El último criterio quedó satisfecho con el diseño escalable que ya comentamos. Para cumplir el primero, lo primordial es conocer el conjunto de instrucciones existentes. Y para cumplir el segundo, lo esencial es estudiar el comportamiento de una aplicación multimedia, cuya caracterización pasamos a desglosar a continuación.

□ Caracterización software de una aplicación multimedia

Las aplicaciones multimedia tienen todos unos rasgos muy similares. Desde la perspectiva software más ligada a su constitución, destacaríamos los tres siguientes:

rasgos
software

- ① **Tipos de datos de tamaño reducido organizados en estructuras grandes.** Por ejemplo, una imagen en la pantalla se compone de infinidad de píxeles de 8 bits (12 ó 16 en media y alta resolución), y una partitura musical, de muestras de sonido de 16 bits (24 bits si el sonido es alta fidelidad). datos pequeños
- ② **Operaciones repetitivas simples y regulares.** Por ejemplo, el suavizado de una imagen (actualizar cada píxel con la media de una pequeña región de la imagen centrada en él con la finalidad de reducir su contraste). operaciones repetitivas
- ③ **Alto grado de paralelismo inherente.** Por ejemplo, el suavizado anterior puede realizarse concurrentemente sobre distintas partes de la imagen. paralelismo

□ Caracterización hardware de una aplicación multimedia

Desde la perspectiva hardware más ligada al microprocesador, señalaríamos cuatro rasgos como los más sobresalientes de una aplicación multimedia:

rasgos
hardware

- ① **Respuesta en tiempo real.** Por ejemplo, a la hora de visualizar una secuencia de vídeo, resulta más adecuado prescindir de algunos fotogramas que mostrar todos y ralentizar la imagen. Se hace necesario reservar recursos y anticipar el tiempo necesario para realizar una tarea. respuesta
- ② **Pobre localidad temporal y alta localidad espacial en el acceso a los datos.** En general, el volumen de datos referenciado más recientemente por el procesador (concepto de **conjunto de trabajo** para una aplicación) es superior al primer nivel de memoria caché, siendo necesaria la intervención del segundo nivel. localidad
- ③ **Control de múltiples flujos simultáneos.** Por ejemplo, los datos de una secuencia de imágenes y sus efectos especiales de sonido asociados. control
- ④ **Elevado ancho de banda.** Principalmente, entre los datos y las unidades funcionales del procesador en que son procesados. ancho de banda

Para cumplir con los dos primeros requisitos, se dobló la capacidad de la caché L1 en la versión MMX del propio Pentium, y posteriormente la velocidad de la L2 en los procesadores que

recursos
hardware

albergaron las futuras extensiones, como las SSE de Intel y las Enhanced 3DNow! de AMD. Estas dos también habilitaron instrucciones especiales para un control más exhaustivo de los recursos de la memoria caché, así como bancos de registros dedicados.

Respecto a las dos últimas cualidades, se trataron de cubrir mediante la ampliación del carácter superescalar del procesador en lo que respecta a la interrelación de las unidades funcionales con las demás (replicación de unidades multimedia) y la habilitación de puertos de conexión independientes para ellas.

5.4.4 Compatibilidad

❑ A nivel hardware: Registros extensos

la clave Un aspecto fundamental para el éxito de la iniciativa MMX era garantizar la compatibilidad con los mismos modelos de microprocesadores sin extensiones MMX. Se sabía que si se suministraba rendimiento adicional pero no se podían ejecutar las aplicaciones software ya existentes, el mercado daría la espalda a la idea por las muchas limitaciones que tendría que soportar el usuario final.

coexistencia Pero el problema era más complejo aún, pues se necesitaba asegurar la coexistencia de las viejas aplicaciones con las nuevas MMX en una ejecución multiproceso. Esto se consiguió conmutando el procesador a modo de ejecución en punto flotante cuando los procesos multimedia solicitaban sus servicios, reutilizándose el banco de 8 registros en punto flotante de 80 bits de que disponían tanto el Pentium como el K6 para almacenar los operandos multimedia de 64 bits.

cómo se ejecutan De esta manera, desde el punto de vista de las instrucciones, la compatibilidad estaba asegurada al definir las instrucciones MMX como enteras normales, y desde el punto de vista de los datos, los tipos de datos MMX de 64 bits se mapeaban sobre registros en punto flotante de 80 bits. Con esta operativa en marcha, cuando una aplicación se ejecuta, comprueba antes la presencia de hardware MMX: Si el procesador se encuentra dotado de él, utiliza las nuevas instrucciones y se ejecuta disfrazándose de aplicación en punto flotante; en caso contrario, se ejecutará como una aplicación entera normal.

❑ A nivel software: Compiladores, ensambladores y API

varios enfoques La tecnología MMX se aprovechó de la capa software del sistema para transformar las aplicaciones existentes en otras que pudieran beneficiarse del nuevo repertorio de instrucciones. La conexión puede aprovecharse a diferentes niveles, obteniéndose en cada caso un aumento de rendimiento distinto respecto a una misma aplicación:

reescribir código **1 Vía 1: Priorizar el rendimiento.** El método de programación más eficaz consiste en transformar la aplicación en un programa nativo para instrucciones MMX. Para ello, reescribiremos las funciones que consumen más tiempo para que puedan llamar a las nuevas instrucciones, lo cual puede realizarse de forma automática o manual dependiendo del software que nos ayude:

- compilador
- Si disponemos de un compilador que soporte las nuevas instrucciones, bastaría con recompilar el programa fuente (escrito en C, por ejemplo) y utilizar el nuevo fichero ejecutable.
 - La alternativa manual consiste en instalar un parche para el Macro Assembler de Microsoft que haga a éste generar los códigos de operación de las nuevas instrucciones. Tanto Intel como AMD han proporcionado estos parches a los clientes de sus microprocesadores con extensiones multimedia a través de sus páginas Web. De optar por esta vía, deberemos codificar directamente en lenguaje ensamblador las rutinas cuyo rendimiento se desee mejorar, y hacer uso de los mnemotécnicos de las nuevas instrucciones.
- ensamblador

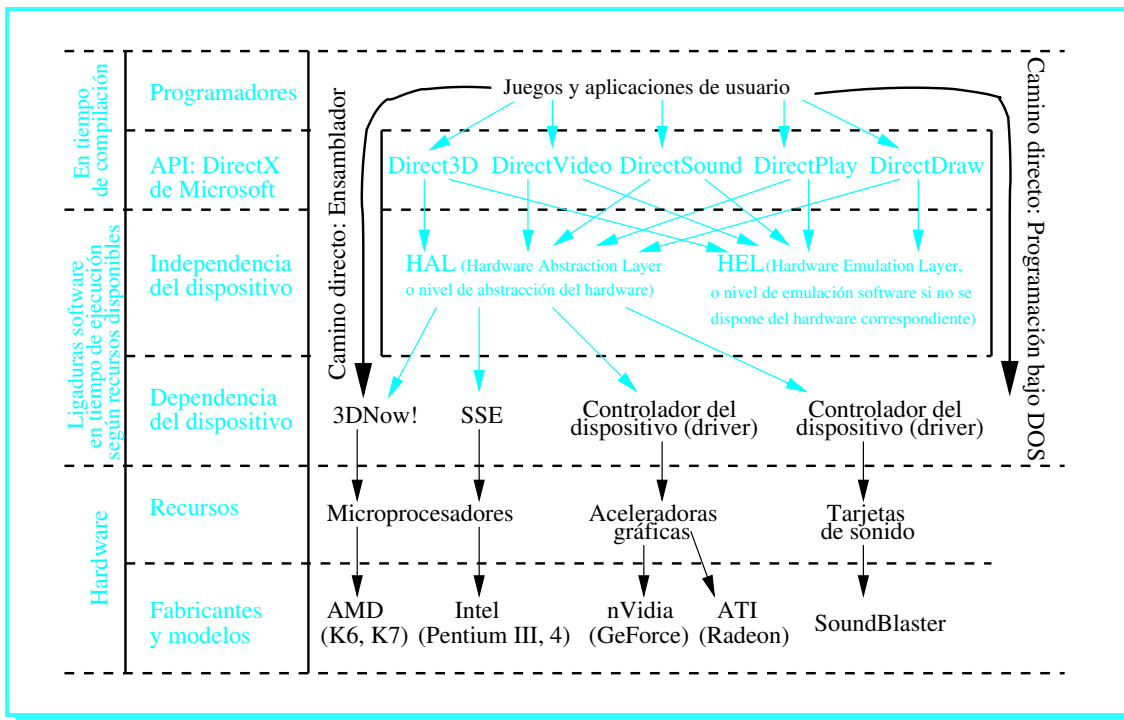


FIGURA 3.18: Las dos vías para la programación de aplicaciones utilizando los conjuntos de instrucciones multimedia. El empleo de un mayor número de capas sacrifica el rendimiento del hardware en favor de la compatibilidad y la facilidad de programación software.

- ❷ **Vía 2: Priorizar la facilidad de programación.** También es posible conseguir un aumento de rendimiento si se utiliza una API que esté escrita a bajo nivel utilizando las extensiones MMX.

Una **API (Application Program Interface)** es un conjunto de funciones y librerías especializadas que define un interfaz a bajo nivel para los programas de aplicación de los usuarios. Estas funciones se ejecutan desde los programas como las tradicionales llamadas al sistema, sólo que en vez de suministrarlas el sistema operativo, lo hace una capa superior que se suministra en un paquete software instalable de forma similar a un *driver*.

De esta manera, llamando a las funciones de esa API especial para MMX, nos aprovecharíamos indirectamente de las nuevas instrucciones, aunque la ganancia esperada sería inferior a la obtenida en el caso anterior.

La [figura 3.18](#) sintetiza las dos alternativas descritas. Ambas ofrecen una dicotomía software similar a la que existe entre programar en ensamblador y en un lenguaje de alto nivel, apostando las principales compañías del sector por la segunda alternativa con objeto de reducir al máximo el tiempo de desarrollo de sus productos.

En esta línea, los principales ejemplos son firmas como 3Dfx Interactive con su API Glide para MMX, Silicon Graphics con el estándar OpenGL y Microsoft con DirectX, que aglutina diferentes API para gráficos, sonido y todo tipo de dispositivos en el contexto de los juegos para PC.

DirectX ha proporcionado sucesivamente versiones más ampliadas para dar cobertura a las posteriores extensiones multimedia, como DirectX 6.0, API que da cobertura al conjunto de instrucciones 3DNow!, DirectX 7.0, API para SSE y Enhanced 3DNow!, y finalmente DirectX 8.0, API para SSE2 y 3DNow! Professional. Microsoft también incluye paulatinamente las nuevas versiones de DirectX en sus viejos sistemas operativos. La cobertura de las API de Microsoft para las distintas extensiones multimedia se resume en la [tabla 3.15](#).

usar una API

qué es una API

dicotomía

principales API para MMX

versiones de DirectX

☛ pág. 114

Versión de DirectX	API que incorpora y características más reseñables	Cobertura multimedia	Primer S.O. que lo incluye
1.0 (Abr'95)	DirectDraw, DirectInput, DirectPlay, DirectSound		Windows'95
2.0	Direct3D		
3.0 (primera vers estable)	_Sound3D suplanta a _Sound y se amplía DirectInput	MMX básico	
5.0	Setup (autoconfiguración), multimonitor, media layer, vertex buffers	MMX completo	Windows'98
5.2	Amplía DirectPlay		
6.0 (Oct'98)	Compresión de texturas, stencil buffers bump mapping, nuevo panel de control	MMX óptimo, 3DNow!	
6.1 (Feb'99)	DirectMusic		
7.0 (Oct'99)	Interfaz con VisualBasic, vertex blending, mejores gráficos y sonido 3D	SSE,Enhanced 3DNow!	Windows'2000, Windows Me
8.0 (Nov'00)	_Draw y _3D se funden en _Graphics, _Music y _Sound se funden en _Audio, DirectShow	SSE2, 3DNow! Professional	
8.1 (Ago'01)	Actualización de imagen más rápida, soporte para amplio número de jugadores		Windows'XP

TABLA 3.15: Evolución histórica de la API DirectX de Microsoft y cobertura de los sucesivos conjuntos de instrucciones multimedia y sistemas operativos en cada una de sus versiones.

La clara orientación de DirectX hacia la industria de los juegos es lo que explica su floja penetración en Windows'NT y la reciente ampliación de DirectX en el ámbito de la video-consola X-Box. DirectX actúa así como capa software independiente de la plataforma hardware, con dos claras ventajas:

- 1 Para la industria de los juegos para PC, permite que las novedades de su ingente mercado puedan salir al mismo tiempo para X-Box sin coste de desarrollo adicional.
- 2 Para el usuario de PC, permite que cada vez que éste incorpore nuevos recursos hardware, pueda aprovechar sus prestaciones multimedia sin más que actualizar la versión de DirectX, disponible a través de la World Wide Web en <http://www.microsoft.com/directx>. Siempre que la aplicación del usuario esté programada utilizando el API DirectX, todo quedará transparente al usuario.

Web!

Fuera del mundo Microsoft, tanto en arquitecturas más potentes y de grandes recursos gráficos (como las estaciones de trabajo) como en otros sistemas operativos (léase Linux), la mejor elección es utilizar el estándar OpenGL.

5.4.5 Ampliaciones al conjunto MMX

Después de una guerra de pleitos entre Intel y AMD por atribuirse la autoría de la idea MMX (guerra que terminó en los juzgados con decisión salomónica), se sucedió una cruenta batalla por desarrollar versiones mejoradas de la original.

AMD respondió antes, a mediados del año 1998 con su 3DNow! para el K6-2, mientras que Intel lo hizo de forma más contundente un año más tarde con su SSE para el Pentium III (70 nuevas instrucciones frente a sólo 21 en 3DNow!). La historia volvió a repetirse poco más tarde: AMD extendió el conjunto con 24 nuevas instrucciones en el Enhanced 3DNow! para su K7 (verano de

la batalla por la sucesión

3DNow!
SSE

Enhanced 3DNow!

PC y X-Box

estaciones
Linux

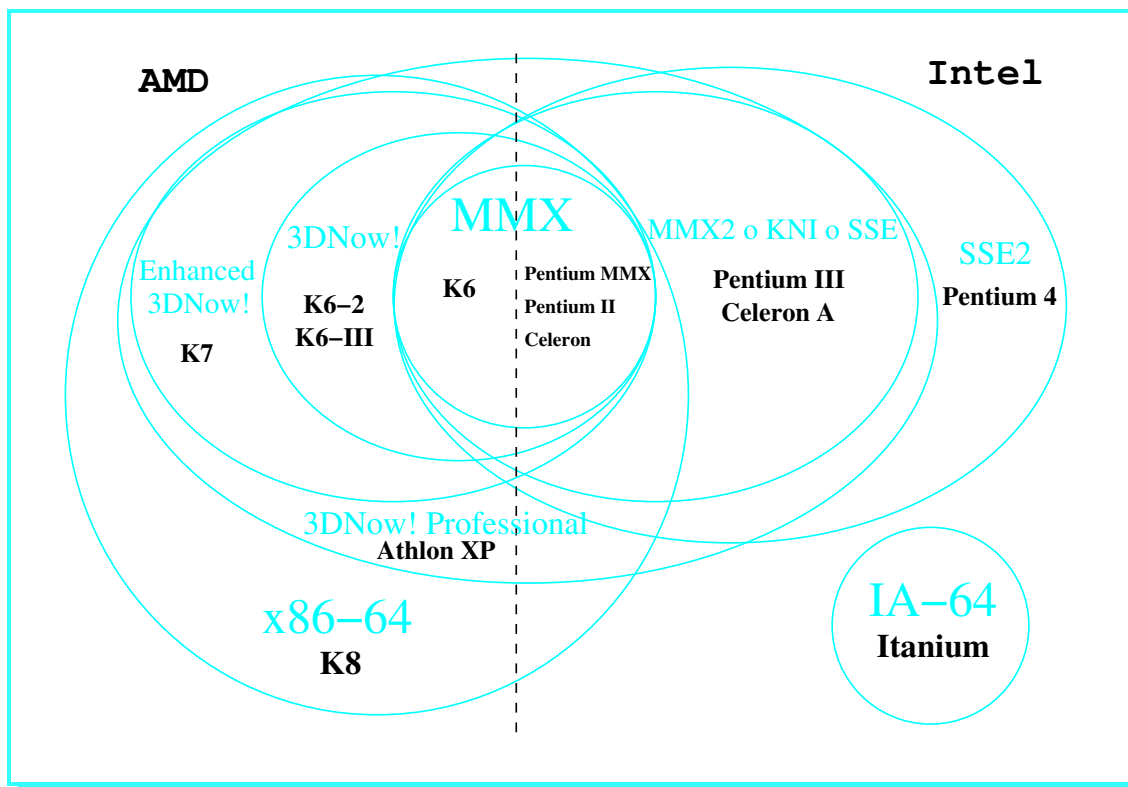


FIGURA 3.19: Las sucesivas ampliaciones del conjunto de instrucciones multimedia desarrolladas por Intel y AMD en años pretéritos frente a las inminentes iniciativas que nos aguardan con la llegada de nuevos conjuntos de instrucciones de 64 bits.

1999), e Intel le respondió un año más tarde con 144 nuevas instrucciones en el conjunto SSE para Pentium 4. La figura 3.19 resume toda esta evolución.

SSE

pág. 115

Todas estas iniciativas no son más que un refinamiento de la idea original, así que para no aburrir con su descripción, vamos a detallar únicamente sus aspectos más sobresalientes:

□ 3DNow! de AMD

Fue la primera ampliación del conjunto de instrucciones MMX. Comandada por AMD y finalizada en el verano de 1998, se incorporó al mercado como una de las principales novedades del procesador K6-2. Además, algunos fabricantes de microprocesadores compatibles de aquella época, como Cyrix y Centaur, decidieron no esperar más y pronto anunciaron la utilización de la tecnología 3DNow! en sus productos, lo que supuso un importante respaldo por parte del mercado.

Al igual que en la tecnología MMX, la base de 3DNow! es la técnica SIMD para el procesamiento simultáneo de operadores asociativos sobre muestras de datos de un mismo tipo. En el repertorio de operaciones disponibles tampoco aparecen variaciones con respecto a aquéllas. La novedad estuvo en el rango de operandos contemplados, incorporándose la manipulación de datos en formato real de simple precisión. Para ello, se habilitó un nuevo tipo de datos *Paired Simple* de 64 bits que empaquetaba dos de estos datos (32 bits cada uno) y que era precisamente el tipo de dato utilizado por el API Direct3D de Windows 98 para la representación gráfica de polígonos 3D, o la reproducción de vídeo MPEG-2 y sonido AC-3.

base

rango de operandos

La forma de implementar las instrucciones 3DNow! es también similar a la MMX, renombrando los registros de punto flotante como MM0..MM7 y utilizando su gran longitud para empaque-

implementación

tar una serie de datos más cortos. La novedad de la circuitería subyacente estuvo en la incorporación de una unidad de procesamiento adicional a la MMX, dedicada en exclusiva al cálculo de las nuevas instrucciones y en la que además se podía disponer de superescalaridad de factor 2 por encontrarse replicada.

21 instrs.
2 grupos

Las nuevas instrucciones son un total de 21, y se dividen en dos grupos distintos: vectoriales y escalares. Las primeras operan simultáneamente sobre dos operandos de 32 bits colocados en las mitades inferior y superior de un registro MM o de una palabra de memoria, mientras que las segundas lo hacen sobre un único operando de 32 bits colocado en la parte baja de dicha palabra.

funcionalidad:

Respecto a la funcionalidad, cubre las siguientes:

- enteros
 - 3 de aritmética entera: Media de 8 datos de 8 bits, permutación de datos de 16 bits, y multiplicación con redondeo.
- reales
 - 10 de aritmética de punto flotante: Suma y resta, suma y resta acumuladas, multiplicación, tres tipos de comparaciones, y máximo y mínimo.
- conversión
 - 4 de conversión: De entero de 16 y 32 bits a punto flotante y viceversa.
- raíz cuadrada
 - 2 para computar la raíz cuadrada de números de punto flotante, que se desglosan en un total de 5 subinstrucciones.
- conmutación
 - Salida del modo MMX del procesador.
- prebúsqueda
 - Prebúsqueda de una línea de caché en L1D.

□ MMX2 ó KNI ó SSE de Intel

etimología

Intel anduvo bastante vacilante con el bautismo de su primera secuela al conjunto MMX. Primero, pensó eternizar esas tres letras, optando por MMX2; después eligió KNI para reflejar en él el nombre del primer procesador suyo que lo incorporaba ⁴; finalmente optó por enfatizar el carácter SIMD de la idea original, acuñando como nombre comercial SSE (*Streamind SIMD Extensions*), que es el que ha quedado para la posteridad.

70 instrs.

El conjunto SSE es bastante más numeroso que el 3DNow!: Cuenta con 70 instrucciones en total, aunque buena parte de ellas son desdobles de una misma operación sobre operandos cuyo abanico de posibilidades se expandió a la computación de punto flotante, incorporándose los formatos estándar IEEE utilizados por Intel: 32 bits para simple precisión, y 64 bits para doble precisión. La otra contribución novedosa de SSE estuvo en las instrucciones dedicadas al control explícito de los contenidos de memoria caché.

El desglose de las 70 instrucciones puede resumirse de la siguiente forma:

- reales
 - 50 nuevas instrucciones para computación de punto flotante siguiendo el mismo desdoble de operandos que posibilita la idea SIMD.
- caché
 - 8 nuevas instrucciones para el control de caché que no tienen nada que ver con la filosofía SIMD desde el punto de vista de su implementación. Controlan de forma individual sobre cada línea de caché: La política de *write-through* o actualización simultánea en memoria principal de las líneas de caché, la compartición de líneas entre varios procesadores acoplados a una misma placa base, y los riesgos de inconsistencias en que puede incurrirse al actualizar la información.
- NMI
 - 12 instrucciones denominadas NMI, de corte muy similar a las ya existentes en el conjunto MMX original.

⁴KNI significa *Katmai New Instructions*, y Katmai era el código de referencia del primer Pentium III.

Dados los requisitos, se hizo necesaria la utilización de un nuevo banco de registros cuya anchura fuese al menos de 128 bits. Así, más que en la concepción, la principal novedad estuvo en su implementación. Este nuevo banco contaba con 8 registros exactamente de 128 bits, y se denominó XMM, incorporándose a los microprocesadores de Intel a partir del Pentium III. La arquitectura del procesador se amplió con una nueva unidad de procesamiento conectada al resto del sistema por un puerto separado en el que no interfería el tráfico de las instrucciones MMX originales, ni tampoco el de las de punto flotante con las que éstas se solapaban.

recursos
hardware

❑ Enhanced 3DNow! de AMD

La nueva réplica de AMD tuvo lugar casi de forma inmediata al lanzamiento de las SSE por parte de Intel. Son un total de 24 instrucciones, que podemos desglosar en tres grupos:

24 instrs.

- 12 típicas de cálculo de números enteros desdoblados. enteros
- 7 dedicadas a optimizar el uso de las cachés, controlando explícitamente la permanencia de datos en ellas e incluso la habilitación de un bypass en el acceso a caché. caché
- 5 aceleran funciones relacionadas con MP3 (MPEG2), sonido Dolby Digital (AC3) y módem ADSL. MP3, AC3, ADSL

Las 19 primeras son sospechosamente parecidas a otras tantas ya existentes en SSE, siendo las últimas 5 realmente novedosas.

Las instrucciones Enhanced 3DNow! se incorporaron en primer lugar al procesador K7 en Junio de 1999, donde ofrecieron una distinción entre tratamiento escalar y vectorial muy similar al que ya comentamos para sus antecesoras sobre el K6-2.

❑ SSE2 de Intel

Aprovechando el lanzamiento de la nueva arquitectura Pentium 4 (Noviembre de 2000), Intel incorporó a este procesador una extensión del conjunto multimedia SSE, denominado SSE2. Consta de un total de 144 nuevas instrucciones, donde vuelve a utilizarse el banco de registros XMM de 128 bits y el puerto de conexión separado de las MMX y de punto flotante.

recursos
hardware
144 instrs.

Respecto a su funcionalidad, el desglose por grupos afines es el siguiente:

- 7 controlan el uso de la caché de forma explícita. caché
- 16 se dedican a efectuar todas las conversiones posibles entre datos enteros empaquetados de 32 bits y punto flotante de simple y doble precisión. conversión
- 8 se dedican al movimiento de datos entre registros MMX y XMM. transporte
- 68 son operaciones enteras similares a las MMX, pero aplicadas sobre los registros XMM, que al ser más anchos, admiten una mayor cantidad de operandos. enteros
- El resto son operaciones de punto flotante. reales

❑ 3DNow! Professional de AMD

La última extensión multimedia de AMD, denominada 3DNow! Professional no introduce novedad alguna en su implementación, pero sí en su concepción.

Son 72 instrucciones adicionales basadas de nuevo en el paradigma SIMD y orientadas tanto a computación entera como a computación de punto flotante. Pero tienen una peculiaridad: 52 de ellas son compatibles con las SSE de Intel. El primer procesador en el que se incorporan es el Athlon XP, lanzado a finales de 2001.

72 instrs.

Tipo	Conjunto de instrucciones		Nº bits de los registros	Conjunto de instrs. antecesor	Primer procesador que lo incorpora		Fecha de salida
	Nombre	Nuevas instrs. incorp.			Intel	AMD	
MUL	MMX	57	64	IA-32	Pent MMX	K6	May'97
	KNI ó SSE	70	128	MMX	Pent III	-	Mar'99
TI	SSE2	144	128	SSE	Pentium 4	-	Nov'00
ME	3DNow!	21	128	MMX	-	K6-2	Ago'98
DIA	Enh. 3DNow!	24	128	3DNow!	-	K7	Jun'99
	3DNow! Prof.	52	128	E. 3DNow!	-	Athlon XP	Nov'01
NUE	IA-64	Todas	64	Ninguno	Itanium	-	May'01
VO	x86-64	N/D	64	3DNow! Prof.	-	K8	Abr'03

TABLA 3.16: Resumen de las principales aportaciones al conjunto de instrucciones de los microprocesadores para PC en los últimos cinco años.

estandarización

Con esta maniobra por parte de AMD se da un certero paso en la estandarización que el software demandaba y en el rendimiento que el hardware anhelaba tras la concepción de las instrucciones multimedia a mediados de los años 90. Unificando este interfaz de bajo nivel, las API definidas en DirectX (que constituyen el siguiente estrato software según ilustramos en la [figura 3.18](#)) ya no necesitan preguntar el hardware que se tiene disponible y ramificarse por un flujo de ejecución diferente en función de él, permitiendo concentrar las optimizaciones del código por una única senda.

pág. 113

DirectX

Caminar de espaldas a Intel y Microsoft durante tantos años ha enseñado a AMD que los grandes fabricantes de software se deben siempre al líder en ventas, y por ello, la mejor manera de garantizar a los clientes del Athlon XP que aprovecharán todas sus prestaciones es comulgar con Microsoft en un soporte plenamente consolidado como el API DirectX.

resumen

La [tabla 3.16](#) recopila a modo de resumen de toda esta sección las principales novedades que han aparecido en los microprocesadores comerciales de Intel y AMD en el periodo 1997-2002 al respecto del conjunto de instrucciones.

pág. 119

relación con
DirectX

La información se completa con la [figura 3.20](#), donde se ilustra su relación con las diferentes versiones del API DirectX de Microsoft así como su secuencia evolutiva, primero siguiendo caminos divergentes desde Intel y AMD, y poco a poco convergiendo junto con Microsoft.

5.4.6 Otras extensiones multimedia

Hay que decir como colofón que prácticamente cada fabricante de microprocesadores tiene ahora en el mercado su propio conjunto de instrucciones multimedia. Los enumeramos a continuación:

Hewlett-Packard

- 1 MAX (*Multimedia Acceleration eXtensions*). Desarrollado por Hewlett-Packard para su familia de microprocesadores RISC PA7x00-PA8x00.

Sun

- 2 VIS (*Visual Instruction Set*). Desarrollado por Sun Microsystems para su familia de microprocesadores UltraSparc..

Sil. Graphics

- 3 MDMX (*Mips Digital Media eXtensions*). Elaborado por Silicon Graphics para su familia de microprocesadores MIPS-V y R2000-R10000.

Compaq

- 4 MVI (*Motion Video Instructions*). Diseñado por Digital (ahora Compaq) para su familia de microprocesadores Alpha.

Motorola

- 5 AltiVec. Desarrollado por Motorola para su familia de microprocesadores Power PC.

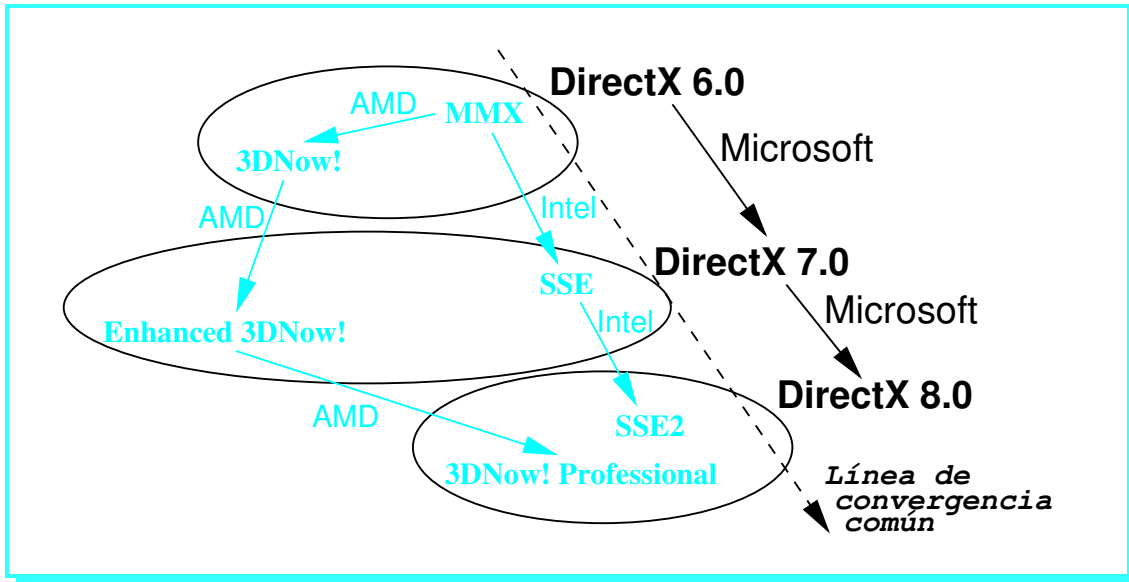


FIGURA 3.20: Secuencia evolutiva de los conjuntos de instrucciones multimedia y su relación con las sucesivas versiones del API DirectX de Microsoft, donde destacamos la divergencia en los primeros pasos y la convergencia hacia un estándar común en los últimos.

La proliferación de conjuntos de instrucciones multimedia no es sino el síntoma más inequívoco de que están siendo utilizados como verdaderos reclamos publicitarios por los fabricantes de microprocesadores. En la práctica, son las compañías de software las responsables de que el usuario saque partido de ellas, y lo cierto es que su potencial está muy infrautilizado en la actualidad. Cuando su utilidad es clara incluso en un contexto de mero cálculo científico, ni siquiera las firmas de juegos por ordenador que son las más beneficiadas apuestan de forma unánime por ellas, habiéndonos encontrado demasiados casos en los que sus posibilidades son sencillamente ignoradas.

infra-
utilizadas

🕒 **Resumen** 🕒

El microprocesador es un componente extraordinariamente complejo. Entre la multitud de parámetros que influyen en su rendimiento, hemos seleccionado los cinco que consideramos más importantes. La [tabla 3.17](#) sintetiza los aspectos del procesador en los que cobra mayor protagonismo cada uno de ellos.

complejidad

➔ [pág. 120](#)

- ❶ La frecuencia mide la velocidad con que se suceden los ciclos del procesador, y no necesariamente revierte sobre su rendimiento. Puede ser un reflejo del mayor número de etapas por instrucción o incluso de una extrema simplicidad en el hardware de la arquitectura.
- ❷ La distancia de integración del transistor es la magnitud menos conocida, pero es, con diferencia, la que mayor influencia ejerce sobre las otras cuatro. En el pasado ha sido la principal responsable del progreso simultáneo de todas ellas, pero dado que los límites del transistor de silicio ya acechan por el horizonte, el relevo debería ser tomado en magnitudes de más alto nivel como las que prosiguen.
- ❸ El paralelismo a nivel de instrucción rompe con la ejecución del código tal y como la planteó el programador, introduciendo una enorme complejidad en su procesamiento lógico.

frecuencia:
reclamo

integración:
la clave

Nivel de abstracción	Parámetro	Incidencia sobre el microprocesador
Bajo: Circuitería	Frecuencia	❖ Velocidad. ❖ Precio final del producto.
	Tecnología de integración	❖ Potencial futuro de mejora. ❖ Comportamiento de variables eléctricas. ❖ Coste de fabricación.
Intermedio: Arquitectural	Paralelismo a nivel de instrucción	❖ Carácter: Rendimiento frente al software a través de la ejecución concurrente de instrucciones.
	Memoria caché interna	❖ Independencia: Protección frente a ralentizaciones procedentes de componentes externos.
Alto: Funcional	Conjunto de instrucciones	❖ Compatibilidad con la capa software. ❖ Favoritismo ante ciertas aplicaciones de moda (RISC, multimedia, ...).

TABLA 3.17: Resumen de las cinco magnitudes más sobresalientes del procesador y principales aportaciones de cada una de ellas.

paralelismo:
tedioso

Los riesgos en los que incurre se solventan introduciendo circuitería dedicada: En este nivel residen las estrategias de diseño más imaginativas, pero la carencia de ideas ha sido alarmante a lo largo de las generaciones que serán objeto de nuestro seguimiento: En no pocas ocasiones hemos visto al mercado tomar una dirección y posteriormente salir en la dirección opuesta buscando sus orígenes. Por lo tanto, no podemos trazar una tendencia aquí, sino tan sólo hablar de movimientos cíclicos.

caché:
complemento

- ④ La memoria caché tiene un nombre (L1D, L1I, L2) que denota su posición en la jerarquía y revela su tamaño aproximado, y un apellido (externa, interna o integrada) que delata su velocidad. Se encarga de alimentar al procesador con instrucciones y datos a gran velocidad, erigiéndose en su complemento ideal.

instrs:
publicidad

- ⑤ El conjunto de instrucciones x86 es el responsable de la compatibilidad software desde hace veinte años, proporcionando un estándar para comunicarse con el procesador. Su enrevesada concepción ha dificultado la innovación y desaprovechado grandes recursos hardware, surgiendo entonces las extensiones multimedia como parches para ayudar a la escritura de nuevos programas. Estas nuevas instrucciones gozan de mejor diseño, pero la ausencia de un estándar ha dificultado su aprovechamiento por parte de la capa software, convirtiéndose más en un reclamo publicitario.

efectos
laterales

Todas estas magnitudes se encuentran muy ligadas entre sí, y a su vez, con los niveles software, de manera que actuando sobre una de ellas se producen multitud de efectos laterales sobre las demás. En consecuencia, no podemos postular una estrategia ganadora en el diseño de microprocesadores, sino tan sólo aspirar a conocerlas con objeto de explotar todas sus ventajas cuando la tecnología o el software nos indiquen que se dan las circunstancias oportunas para apostar por cada una de ellas.

Múltiplos del SI			Submúltiplos del SI		
Factor	Prefijo	Símbolo	Factor	Prefijo	Símbolo
10^3	kilo-	K	10^{-3}	mili-	m
10^6	mega-	M	10^{-6}	micro-	μ
10^9	giga-	G	10^{-9}	nano-	n
10^{12}	tera-	T	10^{-12}	pico-	p
10^{15}	peta-	P	10^{-15}	femto-	f
10^{18}	exa-	E	10^{-18}	atto-	a
10^{21}	zetta-	Z	10^{-21}	zepto-	z
10^{24}	yotta-	Y	10^{-24}	yocto-	y

TABLA 3.18: Principales prefijos para los múltiplos de las unidades del Sistema Internacional (SI).

Nuevos múltiplos binarios del SI					Exceso respecto a 10^x
Factor	Prefijo	Procedencia	Símbolo	Valor decimal	
2^3	kibi-	KIloBInary	Ki	1.024	2.40 %
2^6	mebi-	MEgaBInary	Mi	1.048.576	4.85 %
2^9	gibi-	GIbiBInary	Gi	1.073.741.824	7.37 %
2^{12}	tebi-	TEraBInary	Ti	1.099.511.627.776	9.95 %
2^{15}	pebi-	PEtaBInary	Pi	1.125.899.906.842.624	12.59 %
2^{18}	exbi-	EXaBInary	Ei	1.152.921.504.606.846.976	15.29 %

TABLA 3.19: Los nuevos prefijos para los múltiplos binarios definidos por la ISO y la IEC

✿ La anécdota: Magnitudes oficiales y oficiosas ✿

El universo numérico mantiene ciertos desencuentros entre la comunidad científica en general, acostumbrada a la tradicional base decimal del Sistema Internacional (SI), y el gremio de los informáticos, más proclives a utilizar la base binaria originaria del bit.

El azar ha querido que los sucesivos exponentes ternarios de la base decimal ($10^3 = 1000$) que el SI utiliza para articular los prefijos Kilo-, Mega- y Giga-, ostenten valores muy similares a las potencias decimales de la base binaria ($2^{10} = 1024$) (ver [tabla 3.18](#)). Así se ha alentado un tácito redondeo en el que usamos 1 Mbyte para indicar 1.048.576 (2^{20}) bytes, cuando en realidad, según el SI, serían sólo 1.000.000 (10^6) bytes.

La organización interna de la memoria provoca que cueste lo mismo fabricar 1.048.576 que 1.000.000 de bytes, por lo que a su industria poco le preocupa que en realidad se ofrezca al cliente un 4.85 % de espacio de regalo. Pero en los discos, la cosa cambia: 40 Gbytes son para una marca 40.000.000.000 bytes, ni uno más, ya que comulgar con 1024 en lugar de con 1000 a la altura del Gigabyte supone ya un regalo del 7.37 %, que además, esta vez sí repercute en el coste de fabricación.

Las imprecisiones serán mayores a medida que el tiempo avance y las magnitudes se vayan haciendo más grandes, por lo que de cara a evitarlas en los textos de corte formal y científico, tanto la ISO (International Standardization Organization) como la IEC (International Electrotechnical Commission) definieron entre 1999 y 2000 nuevos prefijos para los múltiplos binarios ⁵, con objeto de poder distinguirlos de los decimales (ver [tabla 3.19](#)).

⁵ Pueden consultarse sus respectivas publicaciones, "ISO/TC 12, Quantities, units, symbols, conversion factors", y "IEC/TC 25, Quantities, units, and their letter symbols".