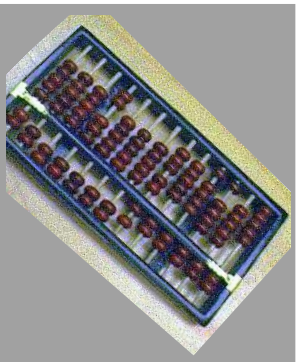
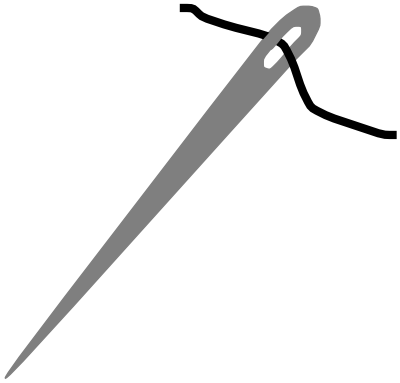


Arquitectura de Computadores



$$T = N \cdot CPI \cdot t_c$$

J.M. Llabería
E. Herrada
A. Olivé

© Copyright 2004, 2007 los autores, Universidad Politécnica de Cataluña

Contenidos

Capítulo 1 Computadores y figuras de mérito

1

COMPUTADORES Y FIGURAS DE MÉRITO

Los procesadores han experimentado cambios significativos en las últimas décadas. Sin embargo, el modelo básico de cálculo que implementan no se ha modificado ya que sigue siendo la máquina von Neumann. Un programa consiste de instrucciones y datos. Las instrucciones se codifican en un conjunto de instrucciones específico y el modelo de cálculo es todavía un único flujo de instrucciones que se ejecutan en secuencia y actualizan el estado de la arquitectura (memoria y registros).

En este capítulo se describe en primer lugar la máquina von Neumann y se muestra el cuello de botella que representa la memoria. Seguidamente se describe el comportamiento de los programas al acceder a memoria y su explotación a nivel hardware para reducir el tiempo medio de acceso a memoria.

Posteriormente se describen las tecnologías que, en las últimas décadas, han permitido el incremento de rendimiento de los procesadores y las figuras de mérito utilizadas para caracterizar un procesador. En particular se describen las técnicas de concurrencia que han permitido un espectacular incremento del rendimiento. Finalmente se describe la tecnología de fabricación que sustenta el incremento de rendimiento de los procesadores y las imbricaciones entre tecnología de fabricación y microarquitectura, haciendo hincapié en el consumo de potencia.

Contenido

MÁQUINA VON NEUMANN

Una máquina von Neumann dispone de posiciones de almacenamiento direccionables y sus valores pueden modificarse las veces que sea necesario mediante un conjunto de instrucciones. La secuencia de control se implementa almacenando las instrucciones en memoria y utilizando un puntero, denominado contador de programa (CP), el cual indica la instrucción que debe interpretarse.

En la Figura 1.1 se muestra un esquema simplificado de una máquina von Neumann en el que no se muestra la comunicación con el exterior. Se distinguen dos componentes: memoria y unidad de procesado (UP). Esta última incluye el control y las unidades funcionales (UF) que se utilizan para manipular los valores almacenados en memoria.

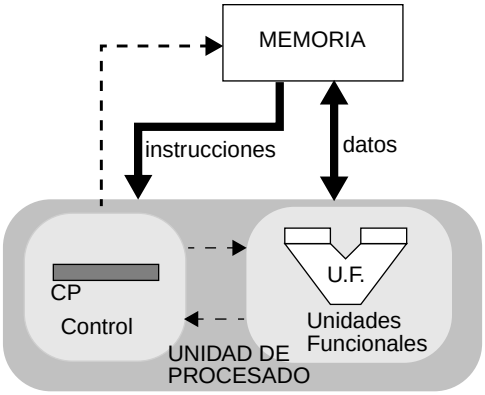


Figura 1.1 *Esquema simplificado de una máquina von Neumann. Memoria y unidad de procesado.*

Las instrucciones que manipulan datos o valores especifican, explícitamente o implícitamente, las posiciones de almacenamiento de los datos y el resultado y la operación que debe efectuarse con los datos para determinar el resultado. Además se dispone de instrucciones que permiten explicitar la secuencia de instrucciones que se interpreta.

El conjunto de instrucciones y su codificación se denomina arquitectura del conjunto de instrucciones (ACI) o lenguaje máquina y especifica la arquitectura de un computador. En el mercado existen máquinas con diferentes ACI. Un programa escrito en una ACI no se puede interpretar en una máquina que interpreta otro ACI. El

lenguaje máquina es de tipo imperativo como el lenguaje C y es la porción de la máquina visible al programador o al compilador. Este último se utiliza para traducir un lenguaje de alto nivel como C a lenguaje máquina.

El conjunto de modelos de máquinas que interpreta un mismo lenguaje máquina (LM) se denomina familia. Cada modelo puede utilizar una implementación distinta en función de la tecnología disponible o de su destino comercial, lo cual determina su complejidad y prestaciones (Figura 1.2).

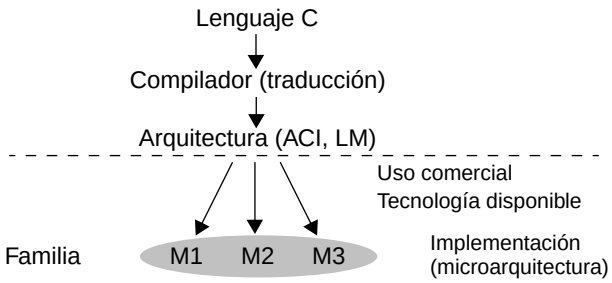


Figura 1.2 *Esquema de relaciones entre lenguaje de alto nivel, compilador, arquitectura y microarquitectura.*

La ACI suministra un nivel conceptual de abstracción a los que implementan compiladores, los cuales necesitan conocer las instrucciones permitidas y como se codifican, y a los diseñadores de UP (microarquitectos), los cuales construyen máquinas que interpretan las instrucciones.

Mientras que la semántica del lenguaje máquina puede ser potente y su codificación engorrosa, el núcleo de las UP actuales interpreta un conjunto de instrucciones de fácil codificación y ejecución. La brecha semántica entre un lenguaje máquina con un elevado nivel semántico y las instrucciones que interpreta el núcleo de una UP se rellena mediante un proceso de traducción dinámico. Una instrucción del lenguaje máquina se traduce en una o varias instrucciones que interpreta el núcleo de la UP.

Las instrucciones que interpreta el núcleo de una UP leen los datos fuente del banco de registros y almacenan el resultado en un registro. Para actualizar memoria se utilizan instrucciones exclusivamente con este propósito. En la Figura 1.3 se muestra la traducción dinámica de una instrucción de lenguaje máquina que suma el contenido de un registro con el contenido de una posición de memoria y almacena el resultado en memoria. La instrucción de

lenguaje máquina se traduce dinámicamente en tres instrucciones: la primera instrucción lee la posición de memoria, la siguiente instrucción efectúa la suma y la tercera instrucción almacena el resultado de la suma en memoria.

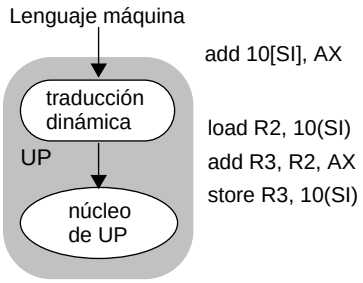


Figura 1.3 *Traducción dinámica de una instrucción de lenguaje máquina en instrucciones que interpreta una UP.*

Prestaciones

Las prestaciones de una máquina von Neumann dependen en gran medida de la interfaz entre la UP y memoria. La UP accede a memoria para obtener datos e instrucciones y esta demanda debe ser satisfecha para explotar en plenitud el potencial de la UP.

Dos parámetros importantes de la memoria son:

Latencia. Es el tiempo entre la solicitud de un dato a memoria y la disponibilidad del dato en la UP.

Ancho de banda. Es el número de bytes que se transmiten por unidad de tiempo.

Una operación típica que se efectúa en un procesador es la suma de dos valores: C= A + B. En la Figura 1.4 se muestran, en un seudolenguaje, las instrucciones que interpreta un procesador para efectuar la operación descrita. Las dos primeras instrucciones copian en registros los valores almacenados en las posiciones de memoria que identificamos como A y B. La siguiente instrucción efectúa la suma de los valores leídos y la última instrucción almacena el resultado de la suma en la posición de memoria que identificamos como C.

```
load R1, A
load R2, B
add R3, R2, R1
store R3, C
```

Figura 1.4 *Ejemplo de una secuencia de instrucciones.*

La siguiente expresión evalúa el número de operaciones por unidad de tiempo.

$$OP = \frac{\text{número de operaciones}}{\text{tiempo empleado}}$$

En el tiempo distinguimos el tiempo de acceso a memoria para obtener las instrucciones, leer y escribir los datos y el tiempo para efectuar la suma. Ahora bien, para observar la influencia de la memoria supondremos que el tiempo de las operaciones en la UP es igual a cero. Esto es, suponemos que en el tiempo de ejecución sólo influyen los accesos a memoria.

En el código de la Figura 1.4 el número de accesos a memoria es siete, cuatro para leer las instrucciones y tres para leer y escribir valores. Entonces, si queremos que la máquina interprete 500 millones de operaciones por segundo (MOP), el tiempo de acceso a memoria (TAM) debe ser

$$TAM = \frac{1}{7x500 \times 10^6} = 286 \text{ ps}$$

Si suponemos que los datos que se manipulan se representan en 64 bits y todos los bits se transmiten en paralelo, el ancho de banda necesario (bytes transmitidos por unidad de tiempo) entre memoria y el procesador son

$$AB = \frac{8 \text{ bytes}}{0.286 \text{ ns}} = 28 \text{ Gbytes/s}$$

En la Figura 1.5 se muestra la tendencia de las prestaciones de la UP y de la memoria DRAM en los últimos años. El incremento de prestaciones de la memoria DRAM no tiene comparación con el incremento, bastante mayor, de las prestaciones de la UP, debido a la significativa reducción en el tiempo de ciclo de la UP. Por tanto, hay que pensar en mejorar la organización del sistema de memoria, ya que la tecnología no es suficiente.

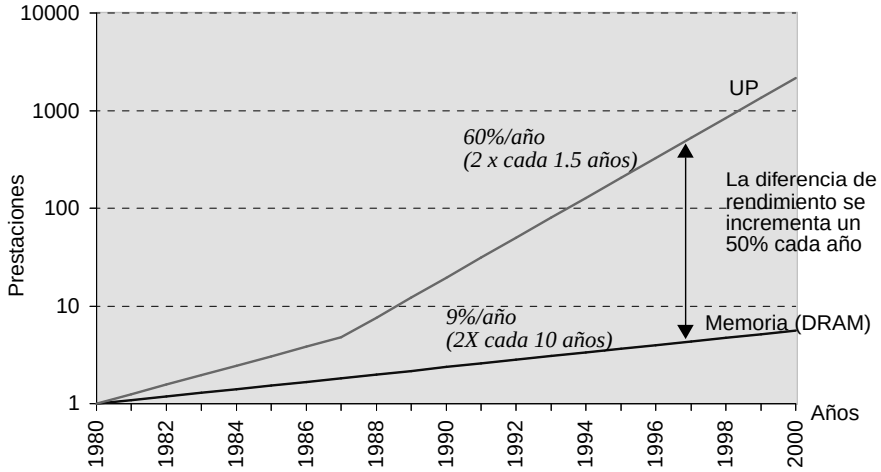


Figura 1.5 *Prestaciones normalizadas de la UP y memoria en los últimos años.*

PROPIEDAD DE LOCALIDAD Y JERARQUÍA DE MEMORIA

En esta sección se presenta una característica observada en los accesos a memoria de un programa. Posteriormente se describe como esta característica se utiliza a nivel hardware para reducir la latencia media de acceso a memoria.

Propiedad de localidad

Una propiedad importante de los programas es la localidad de los accesos a memoria o referencias. Esto es, los programas tienden a interpretar las instrucciones y acceder a datos que se han utilizado recientemente o que están almacenados en posiciones cercanas de memoria.

Localidad temporal. Existe una alta probabilidad de que cuando se accede a una posición de memoria esta sea accedida en un futuro cercano.

Localidad espacial. Existe una alta probabilidad de que cuando se accede a una posición de memoria sean accedidas, en un futuro cercano, posiciones de memoria cuya dirección es cercana.

Explotación de la localidad A la vista de la propiedad de localidad, las posiciones de memoria accedidas en un intervalo de tiempo son un conjunto limitado y predecible del espacio de direcciones (lógico) del programa. El conjunto de posiciones de memoria accedidas se denomina conjunto de trabajo. Entonces, la idea es explotar la propiedad de localidad de los accesos a memoria y su predecibilidad mediante una organización específica del sistema de memoria.

Ejercicio

Describa las propiedades de localidad de las variables s y A del siguiente código.

```
do I =1, N
   s = s + A(I)
enddo
```

Respuesta

La variable s muestra localidad temporal ya que en cada iteración se accede a ella tanto en lectura como escritura. La variable A muestra localidad espacial ya que los elementos de un vector se almacenan de forma contigua en posiciones de memoria y en iteraciones consecutivas se accede a elementos consecutivos del vector A.

Jerarquía de memoria

En la Figura 1.6 se muestra una organización jerárquica actual de memoria, siendo su objetivo reducir el tiempo medio de acceso, explotando la localidad de los accesos a memoria. En la parte inferior de la figura se observa el nivel de registros, seguidamente dos niveles de memoria cache y finalmente la memoria.

Cerca de las UF se ubican memorias con poca capacidad, reducido tiempo de acceso y gran ancho de banda. A medida que nos alejamos de las UF se incrementa la capacidad, se incrementa el tiempo de acceso y se reduce el ancho de banda.

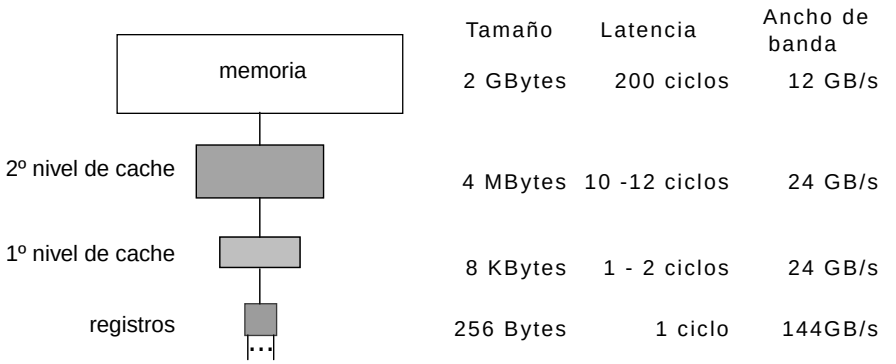


Figura 1.6 *Jerarquía de memoria. Los ciclos se corresponden con ciclos de UP. Se supone un procesador cuya frecuencia de funcionamiento es 1.5 Ghz y que en cada ciclo puede efectuar 8 lecturas y 4 escrituras al banco de registros. El primer nivel de cache tiene dos caminos de acceso de 8 bytes. El segundo nivel de cache tiene un camino de acceso y puede transmitir 32 bytes cada 2 ciclos. Para acceder a memoria se dispone de un camino de acceso que permite transmitir 64 bytes cada 8 ciclos.*

El coste de la memoria también es un factor importante en el diseño de la jerarquía de memoria. Una memoria con latencia reducida y elevado ancho de banda tiene un coste elevado. Por otro lado, aunque el coste no fuera un problema, restricciones tecnológicas de fabricación y organización de una memoria de gran capacidad limitan la latencia y el ancho de banda que puede conseguirse.

El primer nivel de la jerarquía se denomina banco de registros y la gestión del contenido de los registros se efectúa de forma explícita mediante instrucciones de lenguaje máquina. El nivel de registros explota la localidad temporal de las referencias a datos.

Los siguientes niveles se denominan memoria cache y la gestión de contenidos es usualmente transparente al lenguaje máquina. Cuando se efectúa una referencia a memoria desde la UP hay hardware específico que comprueba si la información está almacenada en el primer nivel de cache. Si no está almacenada se comprueba el siguiente nivel y así de forma sucesiva hasta que se encuentra un nivel donde está almacenada la información. Ultima-

mente la mayoría de procesadores también disponen de instrucciones, denominadas de prebúsqueda, para traer anticipadamente información desde memoria a alguna cache cercana al procesador.

El nivel de memoria, aunque tiene una gran capacidad de almacenamiento, tampoco almacena el programa en su totalidad. El programa se almacena en su totalidad en un dispositivo denominado disco y la gestión de contenidos se efectúa de forma explícita por el Sistema Operativo, mediante la técnica denominada memoria virtual que requiere un soporte hardware.

Los niveles de cache y memoria explotan la localidad temporal y espacial de las referencias a datos e instrucciones mediante un algoritmo de reemplazo y un tamaño de bloque, el cual es usualmente la unidad de transferencia entre dos niveles de la jerarquía y además es la unidad de mapeo.

El tiempo de acceso a memoria es variable, ya que depende del nivel de cache donde se encuentre almacenada la referencia. Entonces, el tiempo medio de acceso a memoria se expresa en función de los fallos de cada nivel de cache y la penalización que representa tener que acceder al siguiente nivel de la jerarquía, para obtener el dato o la instrucción. Supondremos que el programa y los datos están cargados en memoria, ya que el tiempo de servicio de un fallo de memoria es varios órdenes de magnitud mayor que la penalización que representa acceder a memoria. Es decir, que no consideramos el acceso a disco.

En una jerarquía con un nivel de cache, el tiempo medio para efectuar un acceso, medido en ciclos de UP (CMA), se evalúa mediante la siguiente expresión.

$$CMA = c_{acierto} + f_R \times P_f$$

donde $c_{acierto}$ es la latencia en ciclos de UP cuando se acierta en cache, f_R mide los fallos de cache por referencia a memoria y P_f la penalización en ciclos de UP en cada fallo de cache.

Ejercicio

Supongamos una cache de primer nivel de 8 KBytes, de mapeo directo y tamaño de bloque de 32 bytes y que el siguiente nivel de la jerarquía es memoria. Calcule el tiempo medio de acceso a dato (CMA) en el siguiente código, suponiendo que cuando se acierta en cache el número de ciclos es 1 y cuando se falla el número de ciclos adicionales es 30.

```
do I =1, N
   s = s + A(I)
enddo
```

Además, cuando se inicia la ejecución del bucle, suponga que la variable A no está almacenada en cache, que la variable s se ha asignado a un registro antes de iniciar las iteraciones del bucle y se almacena en memoria después de las N iteraciones del bucle, siendo un acierto en cache. También, suponga que la variable A está alineada en memoria a bloque de 32 bytes y que el tamaño de un elemento de A y la variable s es de 64 bits.

Respuesta

El tamaño de un dato son 64 bits. Por tanto, un bloque contiene $32\ \text{bytes} / 8\ \text{bytes} = 4$ elementos del vector A. El vector A está almacenado en posiciones contiguas de memoria y alineado a tamaño de bloque. Entonces, cada 4 iteraciones del bucle se produce un fallo de cache.

El número de iteraciones del bucle es N. Por tanto, el número de fallos de cache debidos a la variable A es $\lceil N/4 \rceil$ y la frecuencia de fallo por referencia es $f_R = \lceil N/4 \rceil / N$. Entonces, el número medido de ciclos de un acceso a datos es

$$CMA = c_{acierto} + f_R \times P_f = 1 + \left(\left\lceil \frac{N}{4} \right\rceil / N \right) \times 30$$

$$CMA \sim 1 + 0.25 \times 30 = 8.5$$

PROGRESO DE LOS PROCESADORES Y MÉTRICAS

En las décadas pasadas los computadores y en especial los microprocesadores (UP y primeros niveles de la jerarquía de memoria integrados en un dado de silicio) han experimentado avances significativos gracias a la tecnología CMOS y a las herramientas de diseño. El coste de los computadores se ha reducido a la vez que se ha visto incrementado el rendimiento.

La tecnología CMOS ha permitido implementar funciones lógicas, las cuales a su vez permiten diseñar módulos complejos y mediante el interconexionado de estos últimos se implementa un microprocesador o procesador. Así mismo, el espectacular progreso de la tecnología CMOS ha permitido incrementar las prestaciones de los procesadores.

Por otro lado, las herramientas de diseño facilitan el diseño de los microprocesadores y su verificación y reducen el tiempo necesario para crear un nuevo microprocesador con mejores prestaciones.

Evolución en el diseño de los procesadores

El progreso experimentado en el diseño de los procesadores es debido a una combinación de esfuerzos en:

- Tecnología de fabricación. Es el sustrato del que se parte para implementar procesadores. Determina los elementos de un procesador que pueden integrarse en un dado de silicio.
- Técnicas de diseño lógico y de circuitos. Innovaciones en el diseño de circuitos dan lugar a nuevos métodos para realizar funciones lógicas más eficientemente.
- Microarquitectura. La organización e implementación de la UP y jerarquía de memoria permiten reducir el tiempo que se tarda en interpretar cada instrucción.
- Arquitectura (lenguaje máquina) y compiladores que optimizan el código generado. Intentan reducir el número de instrucciones que debe ejecutarse para efectuar un cálculo.

Métricas para caracterizar un procesador

Algunas figuras de mérito importantes que se utilizan para caracterizar un procesador son:

- Rendimiento. Mide la inversa del tiempo que tarda en completarse una tarea o varias tareas. Depende entre otros factores de parámetros tales como el procesador, la carga de trabajo (conjunto de programas que se ejecutan), configuración (jerarquía de memoria, entrada/salida), el compilador y el sistema operativo.

- Potencia. Es la energía consumida por unidad de tiempo y se mide en vatios. Los procesadores de elevado rendimiento requieren mayor consumo.
- Coste. Está determinado básicamente por el tamaño físico del dado manufacturado en silicio. Areas grandes del dado determinan un coste de manufacturación superior al crecimiento lineal.
- Complejidad. Refleja el esfuerzo requerido para el diseño, validación y manufacturación del procesador. La complejidad está afectada por el número de dispositivos en el dado de silicio, el nivel de agresividad utilizado en el diseño para obtener rendimiento y la potencia máxima y área del dado que se han establecido previamente.

MÉTRICAS DE RENDIMIENTO

En esta sección se presentan métricas para evaluar el rendimiento de un procesador y para determinar el incremento o ganancia que representa una mejora en una parte del procesador.

Rendimiento de un procesador

La siguiente expresión describe el rendimiento de un procesador y caracteriza los factores más relevantes que contribuyen a su valor.

$$\frac{1}{\text{Rendimiento}} = T = N \times CPI \times t_c$$

Cuanto menor es el tiempo de ejecución (T) mayor es el rendimiento. Los factores utilizados para calcular el tiempo de ejecución de un programa son: a) el número de instrucciones ejecutadas (N), b) el número medio de ciclos por instrucción (CPI) y c) el tiempo de ciclo (t_c).

Teniendo en cuenta las unidades de los factores, la expresión anterior se expresa de la siguiente forma:

$$\frac{1}{\text{Rendimiento}} = \frac{\text{Tiempo}}{\text{Programa}} = \frac{\text{Instrucciones}}{\text{Programa}} \times \frac{\text{Ciclos}}{\text{Instruccion}} \times \frac{\text{Tiempo}}{\text{Ciclo}}$$

El tiempo de ejecución de un programa puede reducirse actuando sobre cualquiera de los tres factores de la expresión. Es suficiente con reducir cualquiera de ellos.

Sin embargo, los tres factores no son totalmente independientes y las interacciones entre ellos no son simples y tampoco es sencillo caracterizarlas. Entonces, mejorar el rendimiento requiere tomar compromisos entre varias opciones. Los factores de la expresión que calcula el tiempo de ejecución dependen del compilador, lenguaje máquina, organización y tecnología. En la Figura 1.7 se muestran estas relaciones de dependencia.

T =	N	x	CPI	x	t_c
	compilador		lenguaje máquina		organización
	lenguaje máquina		organización		tecnología

Figura 1.7 *Dependencias de los factores de la expresión del tiempo de ejecución con el compilador, el lenguaje máquina, la organización y la tecnología.*

El número de instrucciones de un programa depende del lenguaje máquina y del compilador. El CPI depende del lenguaje máquina y de la organización (UP y jerarquía de memoria). El tiempo de ciclo depende de la organización y de la tecnología (diseño de circuitos y fabricación) utilizada en la construcción.

Ahora bien, existen técnicas que afectan casi exclusivamente a uno de los factores. Una de ellas es que el compilador efectúe optimizaciones que reduzcan el número de instrucciones sin incrementar el CPI. Otra son los avances tecnológicos en el diseño de circuitos o en el proceso de fabricación, que reducen el retardo de propagación de las señales y por tanto puede reducirse el tiempo de ciclo sin incrementar el CPI. También una organización jerárquica de memoria, que explote la localidad de las referencias a memoria, reduce el CPI.

La técnica clave para reducir el CPI es la segmentación del proceso de interpretación de instrucciones y la interpretación solapada de varias instrucciones (procesador segmentado). La segmentación es semejante a una cadena de montaje, cada paso o etapa en la segmentación finaliza una parte de la interpretación de una instrucción.

Otra técnica utilizada para reducir el CPI, o incrementar el número de instrucciones por ciclo (IPC = 1 /CPI), es el paralelismo. Al aplicar esta técnica se interpretan de forma paralela varias instrucciones (procesador superescalar). Adicionalmente, estos últimos procesadores utilizan la técnica de segmentación.

La técnica de segmentación requiere usualmente añadir poco hardware a un diseño del procesador que interpreta las instrucciones en serie. Básicamente son los elementos que permiten que las etapas no interfieran entre sí. La técnica de paralelismo requiere replicar o añadir de forma significativa hardware al procesador serie.

La segmentación del camino de datos (microarquitectura) también permite reducir el tiempo de ciclo. En los últimos años, una vez se utilizan extensamente las técnicas de segmentación y paralelismo, la mayor parte del incremento de rendimiento ha sido debido al incremento de la frecuencia. La frecuencia se mide en hercios (Hz y su inversa es el tiempo de ciclo ($f = 1/t_c$).

En la Figura 1.8 y en la Figura 1.9 se muestra el rendimiento de computadores que utilizan procesadores de las familias Alpha y Pentium con distintas frecuencias de funcionamiento. El rendimiento se ha medido ejecutando los programas catalogados como enteros del conjunto de programas de prueba SPEC95. En las dos familias de procesadores se observa que al incrementar la frecuencia se incrementa el rendimiento.

En la familia de procesadores Alpha la influencia de la frecuencia del procesador en el rendimiento del computador se ha incrementado en cada nueva generación de procesador. Notemos que, en cada procesador de la familia, la pendiente de la aproximación lineal del rendimiento es distinta.

El incremento de rendimiento en cada nueva generación de los procesadores Alpha es debido a que se incrementa el número de instrucciones que se interpretan concurrentemente, a mejoras de la microarquitectura como la planificación dinámica de instrucciones (21264) y a mejoras en la jerarquía de memoria.

En la familia Pentium el incremento de prestaciones en cada nueva generación del procesador es pequeño. Un caso a destacar es el procesador Pentium III E en el que se incluyó el 2º nivel de

cache (256 Kbytes) en el chip y además funcionando a la misma frecuencia que el procesador. Adicionalmente se incluyeron mecanismos para mejorar la comunicación con memoria.

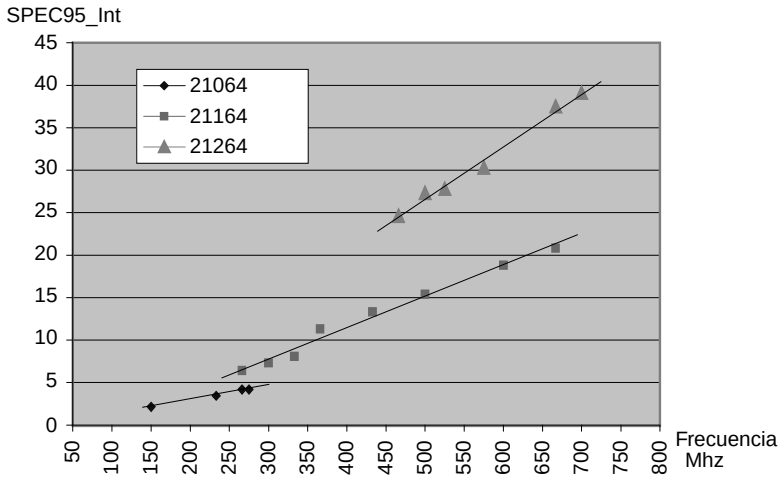


Figura 1.8 *Rendimiento de computadores con procesadores de la familia Alpha al ejecutar los programas enteros del conjunto de programas de prueba SPEC95.*

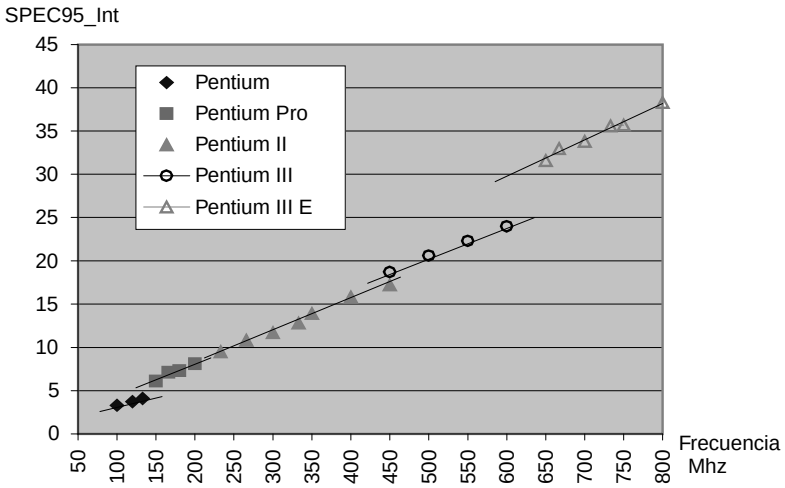


Figura 1.9 *Rendimiento de computadores con procesadores de la familia Pentium al ejecutar los programas enteros del conjunto de programas de prueba SPEC95.*

Ejercicio

Actualmente el incremento de frecuencia lleva aparejado un incremento del CPI. Para que el rendimiento del procesador no disminuya cuando se dobla la frecuencia, evalúe cuál es el máximo incremento en el CPI que se puede permitir.

Respuesta

El tiempo de ejecución en el procesador con mayor frecuencia debe ser menor igual que el tiempo de ejecución en el procesador con una frecuencia mitad.

$$T_{2f} \leq T_f$$

$$N \times CPI_{2f} \times [1/(2 \times f)] \leq N \times CPI_f \times [1/f]$$

$$CPI_{2f} \leq 2 \times CPI_f$$

Entonces, se puede multiplicar por dos el CPI del procesador a frecuencia mitad.

Ejercicio

El producto de los tres términos de la expresión que calcula el tiempo de ejecución es más importante que el valor de los términos individuales.

En la siguiente tabla se muestran características de dos procesadores, construidos por distinto fabricante, que interpretan el mismo lenguaje máquina.

Procesador	CPI	Frecuencia
A	1.2	800 Mhz
B	1.5	1 Ghz

Calcule el tiempo de ejecución de un mismo programa ejecutable en cada uno de los procesadores.

Para el procesador B se dispone de dos compiladores. El mejor de los compiladores genera programas con un 5% menos de instrucciones y al ejecutarse se mide un CPI un 10% menor. Calcule la ganancia cuando se utiliza el mejor compilador. La ganancia se define como el cociente entre tiempos de ejecución, siendo el denominador el tiempo de ejecución cuando se utiliza el mejor compilador.

Respuesta

Como se utiliza el mismo ejecutable el número de instrucciones que finalizan la interpretación es el mismo.

$$T_A = N \times CPI \times [1/f] = N \times 1.2 \times 1.25 = 1.5 \times N \text{ nseg}$$

$$T_B = N \times 1.5 \times 1 = 1.5 \times N \text{ nseg}$$

Entonces, los dos procesadores tardan el mismo tiempo en ejecutar el programa.

La ganancia en el procesador B cuando se utiliza el mejor compilador es

$$G = \frac{T_{original}}{T_{mejor}} = \frac{N \times 1.5 \times 1}{0.95 \times N \times 0.9 \times 1.5 \times 1} = 1.17$$

Componentes en el CPI

En primer lugar se desglosa el CPI en función del CPI de cada tipo de instrucción. Posteriormente se distingue en el CPI los ciclos por instrucción debidos a la UP y los ciclos debidos a que hay que esperar a que la jerarquía de memoria suministre el dato o la instrucción.

DISTINCIÓN POR TIPO DE INSTRUCCIÓN

El número medio de ciclos por instrucción se puede expresar en función del tipo de instrucción de la siguiente forma:

$$CPI = \sum_{i=1}^{n} \frac{N_i \times CPI_i}{N} = \sum_{i=1}^{n} f_i \times CPI_i$$

donde N_i representa el número de veces que la instrucción i es ejecutada en un programa, N es el número total de instrucciones que se ejecutan, CPI_i representa el número medio de ciclos necesarios para ejecutar la instrucción i y f_i es la fracción de instrucciones de tipo i.

Ejercicio

Un computador utiliza un procesador con una frecuencia de reloj de 15 Mhz y al ejecutar un conjunto de programas de prueba se mide una velocidad de ejecución de 10 millones de instrucciones por segundo (MIPS, MIPS = N / T). El tiempo de acceso a cache en caso de acierto es 1 ciclo. ¿Cuál es el CPI efectivo del computador?.

Suponga que con una mejor tecnología de fabricación se consigue que el procesador funcione con una frecuencia de reloj de 30 Mhz. Ahora bien, el sistema de memoria no se modifica y por tanto son necesarios 2 ciclos para acceder a la cache de datos. Algunas instrucciones de este procesador tienen los datos fuente

almacenados en memoria y/o almacenan el resultado en memoria. En concreto, un 30% de las instrucciones efectúa un acceso a memoria y otro 5% efectúa dos accesos a memoria. ¿Cuál es el rendimiento en MIPS del computador cuyo procesador funciona con una frecuencia de 30 Mhz?.

Respuesta

Utilizando la expresión que calcula los MIPS

$$MIPS = \frac{N}{T} = \frac{N}{N \times CPI \times t_c} = \frac{f}{CPI}$$

tenemos que el CPI efectivo a 15Mhz es

$$CPI_{15Mhz} = \frac{f}{MIPS} = \frac{15}{10} = 1.5$$

Cuando se incrementa la frecuencia a 30 Mhz, el CPI de un 30% de las instrucciones se incrementa en una unidad, ya que se efectúa un acceso a memoria. En consonancia, el CPI de las instrucciones que acceden dos veces a memoria (5%) se incrementa en 2 unidades. Entonces, el CPI efectivo a 30 Mhz es

$$CPI_{30Mhz} = CPI_{15Mhz} + 0.3 \times 1 + 0.05 \times 2 = 1.9$$

donde al CPI_{15Mhz} se le suman los ciclos por instrucción adicionales de las instrucciones que efectúan referencias a posiciones de memoria.

Los MIPS efectivos del procesador que funciona a 30 Mhz son

$$MIPS = \frac{f}{CPI} = \frac{30}{1.9} = 15.8$$

Entonces, al doblar la frecuencia del procesador y no modificar la jerarquía de memoria, el incremento en MIPS es menor que 2, ya que las instrucciones que efectúan referencias a posiciones de memoria tienen un CPI mayor.

Ejercicio

Actualmente el incremento de frecuencia lleva aparejado un incremento del CPI. Supongamos que las únicas instrucciones que ven incrementado su CPI son las instrucciones de secuenciamiento. Para que el rendimiento del procesador no disminuya cuando se dobla la frecuencia, evalúe cuál es el máximo incremento permitido en el CPI de las instrucciones de secuenciamiento, suponiendo que estas representan un 30% de las instrucciones ejecutadas.

Respuesta

El tiempo de ejecución en el procesador con mayor frecuencia debe ser menor igual que el tiempo de ejecución en el procesador con una frecuencia mitad.

$$T_{2f} \leq T_f$$

$$N \times (CPI + 0.3 \times \Delta CPI_{secu}) \times [1/(2 \times f)] \leq N \times CPI \times (1/f)$$

donde ΔCPI_{secu} es el incremento de CPI de las instrucciones de secuenciamiento cuando el procesador funciona con una frecuencia dos veces superior (2 x f). Los ciclos por instrucción del procesador cuya frecuencia es f se identifican como CPI.

Efectuando manipulaciones algebraicas en la expresión anterior tenemos

$$CPI + 0.3 \times \Delta CPI_{secu} \leq 2 \times CPI$$

$$0.3 \times \Delta CPI_{secu} \leq CPI$$

$$\Delta CPI_{secu} \leq 3.33 \times CPI$$

El CPI de las instrucciones de secuenciamiento se puede incrementar un 233%.

PENALIZACIÓN POR FALLO DE CACHE

El tiempo de ejecución de un programa puede dividirse en los ciclos que utiliza la UP ($N \times CPI|_{UP}$) y los ciclos que la UP está esperando para completar los accesos a memoria (CM). En los ciclos imputados a la UP incluimos los ciclos necesarios para acceder al primer nivel de cache.

$$T = (N \times CPI|_{up} + CM) \times t_c$$

El número de ciclos de espera para completar todos los accesos a memoria depende del número de instrucciones (N), del número de fallos por instrucción (f_I) y del incremento de tiempo por cada fallo, que denominaremos penalización por fallo (P_f).

$$CM = N \times f_I \times P_f$$

donde $f_I \times P_f$ es el número medio de ciclos de espera correspondiente a los accesos a memoria efectuados por una instrucción.

Teniendo en cuenta las unidades de los factores, la expresión anterior se expresa de la siguiente forma:

$$\frac{\text{Ciclos}}{\text{Programa}} = \frac{\text{Instrucciones}}{\text{Programa}} \times \frac{\text{Fallos}}{\text{Instrucion}} \times \frac{\text{Ciclos}}{\text{Fallo}}$$

Si en lugar de utilizar los fallos por instrucción se utilizan los fallos por referencia a memoria (f_R) obtenemos la siguiente expresión

$$CM = N \times R_I \times f_R \times P_f$$

donde R_I son los accesos a memoria por instrucción. Teniendo en cuenta las unidades de los factores, la expresión anterior se expresa de la siguiente forma:

$$\frac{\text{Ciclos}}{\text{Programa}} = \frac{\text{Instrucciones}}{\text{Programa}} \times \frac{\text{Accesos}}{\text{Instruccion}} \times \frac{\text{Fallos}}{\text{Acceso}} \times \frac{\text{Ciclos}}{\text{Fallo}}$$

Sustituyendo la penúltima expresión en la expresión del tiempo de ejecución obtenemos:

$$T = N \times (CPI|_{UP} + R_I \times f_R \times P_f) \times t_c$$

En la Figura 1.10 se relacionan dependencias de los factores de la expresión anterior con el compilador, lenguaje máquina, organización, jerarquía de memoria y tecnología.

T =	N	x	($CPI\|_{UP}$	+	R_I	x	f_R x P_f)	x	t_c
	compilador		lenguaje máquina		lenguaje máquina		jerarquía de memoria		organización
	lenguaje máquina		organización		compilador				tecnología

Figura 1.10 *Dependencias de los factores de la expresión del tiempo de ejecución con el compilador, el lenguaje máquina, la organización, la jerarquía de memoria y la tecnología.*

También puede tener interés distinguir entre lecturas y escrituras a memoria. Entonces, el tiempo de espera por fallos de cache se expresa como

$$CM = N \times (L_I \times f_L \times P_L + E_I \times f_E \times P_E)$$

donde L_I y f_L son respectivamente las lecturas y los fallos de lectura por instrucción y P_L es la penalización por un fallo de lectura. Así mismo, E_I son las escrituras por instrucción, f_E son los fallos de escritura por instrucción y P_E es la penalización por fallos de escritura.

Ejercicio

Un procesador que funciona a una frecuencia de 40Mhz interpreta 200.000 instrucciones de un programa y el programa utiliza 4 tipos de instrucciones. En la siguiente tabla, que se ha obtenido de una traza experimental del programa, se muestra la frecuencia de los tipos de instrucciones y el número de ciclos por instrucción.

Tipo de instrucción	CPI	frecuencia
aritméticas y lógicas	1	60%
load/store con acierto en cache	2	18%
saltos	4	12%
load/store con fallo de cache	8	10%

Calcule el CPI medio cuando el programa se interpreta en el procesador.

Respuesta

Utilizando la expresión del CPI en función del tipo de instrucción tenemos

$$CPI = \sum_{i=1}^{n} f_i \times CPI_i = 0.6 \times 1 + 0.18 \times 2 + 0.12 \times 4 + 0.1 \times 8 = 2.24$$

Ejercicio

Supongamos un computador donde los ciclos por instrucción es igual a 1.5 si siempre se acierta en cache. Los accesos a datos se efectúan mediante instrucciones load y store, las cuales representan un 35% de las instrucciones. La penalización por fallo de cache son 30 ciclos y los fallos por referencia son un 2%. Calcule los fallos por instrucción y el CPI efectivo cuando se consideran los fallos de cache.

Respuesta

El número medio de referencias a memoria por instrucción es 1 + 0.35 = 1.35, ya que hay que ir a buscar cada instrucción a memoria y un 35% de ellas accede a memoria para leer o escribir un dato. Entonces, los fallos por instrucción son:

$$f_I = (1 + 0.35) \times 0.02 = 0.027$$

El CPI efectivo se calcula como

$$CPI_{efectivo} = CPI_{base} + f_I \times P_f = 1.5 + 0.027 \times 30 = 2.31$$

donde CPI_{base} es el CPI cuando se acierta siempre en cache y la penalización por fallos de cache es f_I x P_f.

Ley de Amdahl

Una expresión formal del sentido común es la ley de Amdahl, que establece que la ganancia por añadir una mejora en un diseño está limitada por la fracción del tiempo original en que se utiliza la mejora.

La ganancia o aceleración que puede obtenerse al añadir una mejora en un diseño se define como el cociente:

$$\text{Ganancia} = \frac{\text{Rendimiento con la mejora}}{\text{Rendimiento original}} = \frac{\text{Tiempo original}}{\text{Tiempo con la mejora}}$$

La ganancia nos indica cuantas veces es más rápido el procesador con la mejora que el procesador original al ejecutar un programa.

En la Figura 1.11 se muestra un diagrama donde, en el tiempo de ejecución de una programa en el procesador original, se distingue la proporción de tiempo en la cual se puede utilizar la mejora introducida. Para simplificar el dibujo los instante de tiempos en los cuales se utiliza la mejora se han agrupado de forma contigua (t_2). La ganancia depende de dos factores: a) la fracción de tiempo en el procesador original donde puede utilizarse la mejora ($f_m = t_2/T_o$) y b) la ganancia cuando se utiliza la mejora el 100% ($g_m = t_2/t_3$).

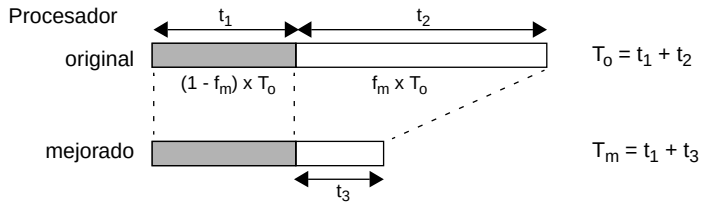


Figura 1.11 *Relación entre el tiempo de ejecución en el procesador original y el tiempo de ejecución en el procesador con la mejora.*

Teniendo en cuenta los tiempos de ejecución, la ganancia se calcula como

$$\text{Ganancia} = \frac{T_o}{T_m} = \frac{T_o}{t_1 + t_3}$$

Como $f_m = t_2/T_o$, tenemos que $t_1 = (1 - f_m) \times T_0$. Entonces, el tiempo con la mejora se puede expresar como $T_m = (1- f_m) \times T_o + t_3$.

Sustituyendo en la expresión que calcula la ganancia obtenemos

$$\text{Ganancia} = \frac{T_o}{t_1 + t_3} = \frac{T_o}{(1 - f_m) \times T_o + t_3}$$

Como $g_m = t_2/t_3$ y $t_2 = f_m \times T_o$, obtenemos $t_3 = f_m \times (T_o/g_m)$. Efectuando sustituciones en la expresión que calcula la ganancia obtenemos:

$$\text{Ganancia} = \frac{T_o}{t_1 + t_3} = \frac{T_o}{(1 - f_m) \times T_o + \frac{f_m \times T_o}{g_m}} = \frac{1}{(1 - f_m) + \frac{f_m}{g_m}}$$

En la Figura 1.12 se muestra el valor de la ganancia en función de f_m y g_m. Se observa que para que la ganancia sea significativa (próxima a g_m) el valor de f_m debe ser cercano a uno.

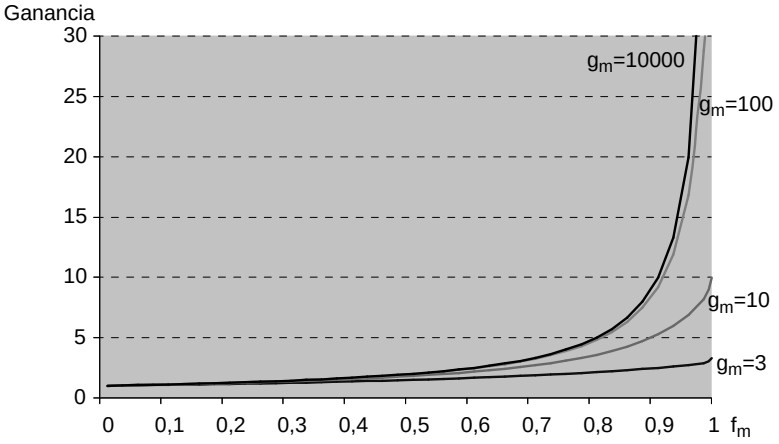


Figura 1.12 *Ley de Amdahl. Ganancia en función de f_m y g_m. Para $f_m = 0$ la ganancia es 1.*

Para constatar la observación sobre el valor de f_m calculamos el valor de f_m necesario para obtener un valor de la ganancia igual a $g_m/2$.

$$\text{Ganancia} = \frac{g_m}{2} = \frac{1}{(1 - f_m) + \frac{f_m}{g_m}}$$

Efectuando manipulaciones algebraicas y sustituyendo g_m por 100 obtenemos

$$f_m = \left.\frac{g_m - 2}{g_m - 1}\right|_{g_m = 100} = 0.99$$

Regla de diseño. Hacer que las instrucciones más frecuentes se ejecuten de forma rápida, puesto que el impacto en el rendimiento es directamente proporcional a la frecuencia de estas instrucciones.

Ejercicio

Analice el interesante caso de suponer que g_m tiende a infinito. Evalúe la expresión resultante cuando la mejora se utiliza el 70% del tiempo original y comente el resultado.

Respuesta

Cuando g_m tiende a infinito obtenemos la siguiente expresión

$$\text{Ganancia}|_{g_m \to \infty} = \frac{1}{1 - f_m}$$

Cuando la mejora se utiliza el 70% del tiempo original, la ganancia neta es únicamente 3.33. Entonces, la ganancia que se puede obtener está limitada por la fracción de tiempo en que no se utiliza la mejora (30%). Por tanto, desde un punto de vista coste/rendimiento no hay que pretender reducir excesivamente el caso común. Esto es, incrementos modestos de f_m mejoran más la ganancia que grandes incrementos de g_m.

Ejercicio

Suponga que la latencia de acceso a una cache es 10 veces inferior a la latencia de acceso a memoria y que la cache se utiliza el 90% del tiempo de ejecución. ¿Cuál es la ganancia que se obtiene por añadir la cache en la jerarquía de memoria de un procesador?.

Respuesta

Utilizando la expresión que cuantifica la ganancia en función de la utilización de la mejora

$$\text{Ganancia} = \frac{1}{(1 - f_m) + \frac{f_m}{g_m}} = \frac{1}{1 - 0.9 + 0.9/10} = \frac{1}{0.19} = 5.26$$

obtenemos que al añadir la cache el procesador es más de 5 veces más rápido.

COMPARACIÓN DE RENDIMIENTOS

Para comparar el rendimiento de dos o más computadores nos centraremos en el tiempo de ejecución de los programas, aunque se pueden utilizar otras métricas como el número de programas ejecutados por unidad de tiempo. En esta última métrica se tienen en cuenta factores tales como la multiprogramación.

La métrica que utilizaremos para comparar el rendimiento de dos máquinas A y B al ejecutar un programa es el cociente entre los tiempos de ejecución.

$$\frac{T_A}{T_B}$$

Cuando este cociente es mayor que uno existe una mejora y se dice que la máquina B es más rápida que la máquina A. Utilizaremos el sufijo 'X' para indicar la relación de mejora y un factor de mejora 2.14X se expresa verbalmente como 2.14 veces.

Otra forma de expresar la modificación del rendimiento es mediante un porcentaje. La proporción de ganancia, en porcentaje, se evalúa como:

$$\left(\frac{T_A}{T_B} - 1\right) \times 100$$

Esta última expresión expresa de forma menos evidente el cambio en el rendimiento cuando este es superior al 100%. Por ejemplo, una ganancia del 114% de un diseño B respecto a un diseño A indica que el diseño B es 2.14 veces mejor.

Comparación con cargas de trabajo

Un computador se utiliza usualmente para ejecutar más de un programa. Entonces, en una comparación de varios computadores hay que tener en cuenta los distintos tipos de programas que se ejecutan, ya que el rendimiento de un computador depende de características del programa.

Cuando se utiliza una carga de trabajo el rendimiento de un computador se evalúa utilizando el tiempo medio ponderado o media aritmética ponderada del tiempo de ejecución de los programas. Para ello utilizaremos la siguiente expresión:

$$T = \sum_{\text{programas}} f_i \times T_i$$

donde f_i es la frecuencia con que se ejecuta el programa i al considerar la carga de trabajo y T_i es el tiempo de ejecución del programa i.

Ejercicio

En la tabla izquierda se muestran los tiempos de ejecución en tres computadores (A, B, C) de 3 programas (P1, P2, P3) que representan una carga de trabajo. En la tabla derecha se indican tres posibles valores de peso relativo de cada programa en la carga de trabajo. Calcule el tiempo ponderado de ejecución.

		Programas		
		P1	P2	P3
Computador	A	1	10	1000
	B	10	100	100
	C	15	15	20

		Programas		
		P1	P2	P3
Ponderación	PO1	0.5	0.3	0.2
	PO2	0.9	0.08	0.02
	PO3	0.98	0.01	0.01

Para cada ponderación de los programas seleccione el mejor computador. Comente los resultados obtenidos en función de la ponderación de los programas en la carga de trabajo.

Comente alguna posible causa en los elementos del computador que produzca las disparidades de tiempo de ejecución de un programa, por ejemplo el programa P3, en los tres computadores.

Respuesta

En la siguiente tabla se muestran el tiempo de ejecución ponderado por programa y el tiempo medio ponderado.

		PO1				PO2				PO3			
		P1	P2	P3	Tiempo medio	P1	P2	P3	Tiempo medio	P1	P2	P3	Tiempo medio
Computador	A	0.5	3	200	203.5	0.9	0.8	20	21.7	0.98	0.1	10	11.1
	B	5	30	20	55.0	9	8	2	19.0	9.8	1	1	11.8
	C	7.5	4.5	4	16.0	13.5	1.20	0.4	15.1	14.7	0.15	0.2	15.0

Con las ponderaciones PO1 y PO2 el computador seleccionado sería el C. En cambio, con la ponderación PO3 el computador seleccionado sería el A.

En función de la ponderación el computador A sería el último seleccionado (PO1) o el primero (PO3). Con el computador C se produce el caso contrario.

Las disparidades en tiempo de ejecución pueden ser debidas a la jerarquía de memoria de los computadores. El tiempo medio de servicio de un acceso a memoria puede ser mucho mayor en el computador A, lo cual puede deberse a que el computador A tiene una cache mucho más pequeña que la que tiene el computador C.

Ejercicio

Dado el CPI de cada uno de los programas de una carga de trabajo y la ponderación de cada uno de ellos en la carga de trabajo, calcule el CPI medio de la carga de trabajo.

Respuesta

El tiempo medio ponderado de ejecución por programa de una carga de trabajo es

$$T = \sum_{\text{programas}} f_i \times T_i = \sum_{\text{programas}} f_i \times (N_i \times CPI_i \times t_c)$$

donde f_i es la ponderación del programa i en la carga de trabajo, N_i es el número de instrucciones, CPI_i es el CPI medio y t_c es el tiempo de ciclo.

El número ponderado de instrucciones de la carga de trabajo es $N = \sum_{\text{programas}} f_i \times N_i$.

Efectuando manipulaciones algebraicas obtenemos

$$T = \sum_{\text{programas}} f_i \times (N_i \times CPI_i \times t_c) = \sum_{\text{programas}} N \times f_i \times \frac{N_i}{N} \times CPI_i \times t_c$$

Como

$$T = N \times CPI \times t_c$$

el CPI medio es igual a

$$CPI = \sum_{\text{programas}} f_i \times \frac{N_i}{N} \times CPI_i$$

Esto es, hay que calcular la media aritmética del CPI de cada programa, ponderada por la proporción de instrucciones de cada programa respecto del número ponderado de instrucciones y la ponderación de cada programa en la carga de trabajo.

CONCURRENCIA

En esta sección se describe la utilización de las técnicas de paralelismo y segmentación para incrementar el rendimiento cuando se interpretan instrucciones en la UP y cuando se accede al sistema de memoria. El objetivo de estas técnicas es reducir el CPI o lo que es lo mismo incrementar el IPC en un modelo de procesador. También se presenta una técnica cuyo objetivo es incrementar la utilización de los recursos de un procesador interpretando concurrentemente varios programas.

Interpretación de instrucciones

Seguidamente se describen varias técnicas aplicadas en el diseño de la UP y se evalúa el incremento de rendimiento potencial que pueden aportar.

Las técnicas de segmentación y paralelismo aplicadas al diseño de un procesador tienen como objetivo reducir el tiempo de ejecución de un programa de forma transparente al programador. Por transparente se entiende que el programador no necesita modificar el programa. La transparencia puede ser a nivel de lenguaje de alto nivel o a nivel de lenguaje máquina.

En el primer caso el encargado de adecuarse al lenguaje máquina del procesador es el compilador o un traductor de lenguaje máquina a lenguaje máquina. En este caso se incluye a los procesadores VLIW.

Cuando se produce el segundo caso se dice que el procesador es binario compatible. Esto es, un código en lenguaje máquina que se ejecuta en un procesador que interpreta las instrucciones en secuencia se puede ejecutar en un procesador en cuya implementación se han utilizado las técnicas de concurrencia siendo el resultados de los cálculos el mismo. En este caso se incluye los procesadores segmentados y a los procesadores superescalares.

La utilización de la técnica multihilo en el diseño de un procesador es ortogonal a la utilización de las técnicas de segmentación y paralelismo. El objetivo de la técnica multihilo es incrementar el número de programas ejecutados por unidad de tiempo.

Segmentación

La segmentación es semejante a una cadena de montaje, cada paso o etapa de la segmentación realiza una parte de la interpretación de una instrucción. En la parte izquierda de la Figura 1.13 se muestra una interpretación serie de varias instrucciones y en la parte derecha se muestra una interpretación segmentada y solapada. La interpretación de una instrucción se ha seccionado en 4 etapas (a, b, c, d), en cada ciclo se inicia la interpretación de una nueva instrucción y las instrucciones concurrentes están en etapas distintas. Los procesadores que interpretan de esta forma las instrucciones se denominan procesadores segmentados escalares.

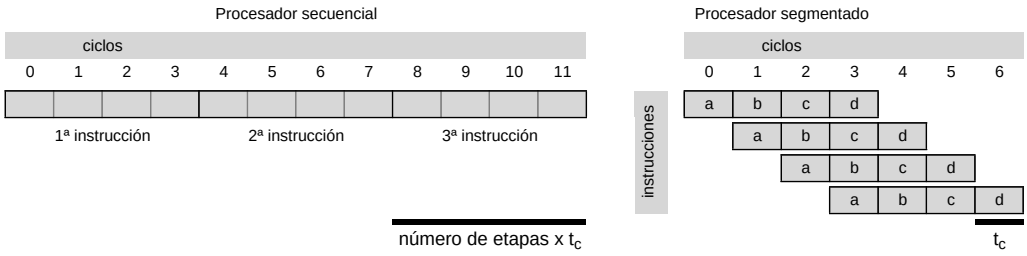


Figura 1.13 *Interpretación serie y segmentada con solapamiento de varias instrucciones.*

La ganancia ideal que se puede obtener cuando se utiliza la segmentación en el diseño de un procesador es

$$\text{Ganancia} = \frac{\text{número de etapas} \times t_c}{t_c} = \text{número de etapas}$$

donde el numerador es el tiempo de ejecución de una instrucción en un procesador serie y el denominador es el tiempo de ejecución de una instrucción en un procesador segmentado. Notemos que en este último procesador, debido al solapamiento en la interpretación de instrucciones, finaliza una instrucción cada ciclo de reloj (t_c).

Paralelismo

La utilización de paralelismo en la implementación de un procesador da lugar a que varias instrucciones estén en la misma etapa de interpretación. En la parte izquierda de la Figura 1.14 se muestra un procesador que interpreta dos instrucciones en paralelo y además el proceso de interpretación está segmentado. Los procesadores que funcionan del modo descrito se denominan procesadores superescalares.

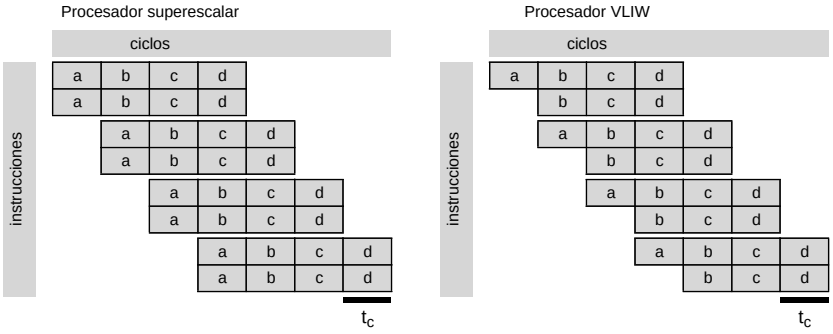


Figura 1.14 *Procesador superescalar (parte izquierda) y procesador VLIW (parte derecha).*

La ganancia ideal que puede obtenerse cuando se utiliza paralelismo y segmentación en el diseño de un procesador es

$$\text{Ganancia} = \frac{\text{número de etapas} \times t_c}{t_c / \text{número de instrucciones por ciclo}}$$

donde el numerador es el tiempo de ejecución de una instrucción en un procesador serie y el denominador es el tiempo de ejecución de una instrucción en el procesador superescalar, en el cual se interpretan varias instrucciones en paralelo.

Simplificando la expresión de la ganancia tenemos

$$\text{Ganancia} = \text{número de instrucciones por ciclo} \times \text{número de etapas}$$

En la parte derecha de la Figura 1.14 se muestra el funcionamiento de un procesador “Very Long Instruction Word” (VLIW). Las instrucciones de estos procesadores especifican la ejecución de varias operaciones. Las operaciones agrupadas en una instrucción son independientes entre sí y los tipos de operaciones que se pueden agrupar en una instrucción están restringidos por las UF disponibles. Estos procesadores requieren que el compilador efectúe un importante trabajo de ordenación y agrupación de las

operaciones. Sin embargo, se simplifica parte de la microarquitectura de la UP, lo cual lo distingue, entre otras peculiaridades, de un procesador superescalar.

Mermas en la ganancia

En el cálculo de las ganancias ideales, tanto cuando se utiliza segmentación como paralelismo, se ha supuesto que la segmentación o paralelismo no incrementa el tiempo de interpretación de una instrucción. Esto es, el denominador de las expresiones anteriores debería ser mayor que el utilizado o en otras palabras, el denominador está infravalorado. También se ha supuesto que los recursos del procesador no están limitados y que las instrucciones que se están interpretando concurrentemente son independientes entre sí.

En los próximos capítulos analizaremos la disminución de la ganancia ideal cuando las circunstancias no son ideales. La merma respecto a la ganancia ideal es debido a limitaciones de recursos y a que hay que respetar la semántica del lenguaje máquina. Por respetar la semántica se entiende que el resultado, al ejecutar un programa en un procesador segmentado, superescalar o VLIW, debe ser el mismo que en un procesador que interpreta las instrucciones de forma serie.

Cuando se produce una situación denominada de riesgo, ya sea por falta de recursos o porque no se respetaría la semántica, se inhibe la segmentación y/o el paralelismo y se siguen interpretando las instrucciones (emulación) reduciendo el grado de concurrencia y en casos extremos inhibiendolo e interpretando las instrucciones de forma (modo) serie. Por grado de concurrencia entendemos el número de instrucciones que se están interpretando concurrentemente. Cuando desaparece el riesgo se vuelven a interpretar las instrucciones de forma segmentada y/o paralela.

Los ciclos en que se reduce el grado de concurrencia en la interpretación de instrucciones se denominan ciclos perdidos. Por tanto, al ejecutar un programa en un procesador que interpreta instrucciones concurrentemente se pueden distinguir varios modos de funcionamiento en función de la concurrencia existente en un instante determinado.

Multihilo

Los procesadores segmentados y superescalares pueden perder bastantes ciclos por situaciones de riesgo debido a que hay que mantener la semántica del lenguaje máquina que se interpreta. En lugar de que estos ciclos se desaprovechen, un procesador multihilo los utiliza para interpretar instrucciones de otros hilos. Por hilo entendemos otro programa o un segmento de código del mismo programa que se puede ejecutar concurrentemente. En este último caso el programador o el compilador determinan los segmentos de código que se pueden ejecutar concurrentemente.

En la Figura 1.15 se muestra mediante un gráfico la reducción del tiempo total de ejecución de dos programas cuando se utiliza la técnica multihilo. En la parte superior se muestra una ejecución serie de los programas. En el programa A después de ejecutarse el trozo A1 se produce un fallo de cache y el número de ciclos que hay que esperarse hasta que la memoria suministre el dato es bastante significativo (E1). Una vez se dispone del dato se reanuda la interpretación de instrucciones (A2). El programa B se comporta de forma similar, con un tiempo de espera E2 entre el trozo B1 y B2.

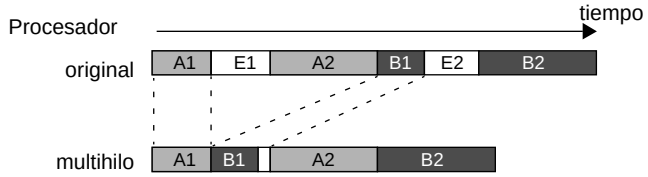


Figura 1.15 *Procesador que interpreta las instrucciones de forma serie y procesador multihilo.*

En la parte inferior de la Figura 1.15 se muestra la ejecución entrelazada de los programas A y B utilizando la técnica multihilo. Cuando se produce el fallo de cache al ejecutar el programa A, en lugar de que el procesador entre en estado ocioso (E1), el procesador multihilo ejecuta el programa B. Cuando se produce un fallo de cache al ejecutar el programa B, el procesador entra en estado ocioso, ya que aún no ha sido servido el fallo de cache del programa A ($t_{E1} > t_{B1}$). Cuando se sirve el fallo de cache del programa A se reanuda la ejecución del programa (A2). Una vez finaliza el programa A se reanuda inmediatamente la ejecución del programa B ya que el fallo de cache ha sido servido ($t_{A2} > t_{E2}$).

Notemos que cuando se utiliza la técnica multihilo, los recursos del procesador están menos tiempo ociosos y además el tiempo total que se tarda en ejecutar los programas es menor.

Un procesador que utiliza la técnica multihilo dispone de hardware para interpretar de forma entrelazada o concurrente instrucciones de varios programas. La inversión hardware que se requiere es pequeña, aunque el esfuerzo de diseño para incluir esta prestación en procesadores con planificación dinámica de instrucciones puede ser significativo, debido a que es necesario considerar un mayor número de posibles interrelaciones entre los flujos de instrucciones concurrentes.

El beneficio de incluir la técnica multihilo en un procesador es incrementar la utilización de los recursos y por tanto el número de programas ejecutados por unidad de tiempo.

Podemos distinguir tres tipos de técnicas multihilo en función del número de hilos concurrentes en un instante determinado:

- Grueso: En un intervalo de tiempo determinado sólo se están interpretando instrucciones de un hilo.
- Fino: Existen varios hilos activos, pero en un ciclo determinado sólo se inicia la ejecución de instrucciones de un único hilo
- Multihilo simultáneo: Existen varios hilos activos y en cada ciclo se pueden estar ejecutando instrucciones de varios hilos.

En la técnica multihilo también se producen pérdidas de rendimiento. Por ejemplo, un programa puede ver incrementado su tiempo de ejecución aunque el número de programas ejecutados por unidad de tiempo sea mayor.

Aunque no se ha comentado explícitamente, la técnica multihilo requiere que la jerarquía de memoria soporte accesos concurrentes. Seguidamente se describe con mayor detalle esta característica de funcionamiento de la jerarquía de memoria.

Acceso a la jerarquía de memoria

La segmentación de la UP y la interpretación de instrucciones en paralelo ha requerido mejorar el diseño de la jerarquía de memoria para que no estuviera limitando las prestaciones. En particular la demanda de ancho de banda se ha incrementado ya que en menor tiempo deben efectuarse el mismo número de accesos a memoria.

En la Figura 1.16 se muestra una tabla donde se calcula el número de instrucciones que podrían ejecutarse cuando hay que acceder a memoria debido a un fallo de cache en 3 computadores construidos con 3 procesadores de la familia Alpha.

	Ciclos	Instrucciones
Alpha (21064)	340 ns / 5.0 ns = 68	68 x 2 = 136
Alpha (21164)	266 ns / 3.3 ns = 80	80 x 4 = 320
Alpha (21264)	180 ns / 1.7 ns = 108	108 x 4 = 432

Figura 1.16 *Número de instrucciones que podrían ejecutarse cuando hay que acceder a memoria. Los ciclos se calculan como el cociente del tiempo de acceso a memoria y el tiempo de ciclo. Las instrucciones se calculan como el número de ciclos por el máximo número de instrucciones que pueden decodificarse en cada ciclo.*

En diseños sencillos de la jerarquía de memoria un fallo de cache inhibe el servicio de otros accesos a memoria (cache bloqueante). También, cuando al interpretar una instrucción load se produce un fallo de cache la interpretación de instrucciones se bloquea (load bloqueante).

Adicionalmente, el acceso a cada nivel de la jerarquía se efectúa en secuencia. Esto es, no se inicia un acceso a un nivel de cache hasta que no se ha detectado un fallo en el nivel previo.

Estas decisiones de diseño se han ido relajando para incrementar la concurrencia siendo el objetivo reducir o soportar la latencia e incrementar el ancho de banda.

Cache no bloqueante

Una mejora en la jerarquía de memoria son las cache no bloqueantes que permiten servir solicitudes de acceso que son aciertos mientras se está sirviendo un fallo de cache.

Este tipo de cache con la inclusión de instrucciones de prebúsqueda en el lenguaje máquina permite soportar la latencia de fallo de cache. El objetivo de una instrucción de prebúsqueda de datos es traer un dato, almacenado en niveles lejanos de la jerarquía, al primer nivel de cache, antes de que una instrucción load quiera cargar el dato en un registro. La ejecución de una instrucción de prebúsqueda es similar a la ejecución de una instrucción load con

la salvedad de que no se carga ningún dato en un registro. Si el bloque que busca la instrucción de prebúsqueda ya está almacenado en el primer nivel de cache finaliza la ejecución de la instrucción. En caso contrario se prosigue la búsqueda en los siguientes niveles de la jerarquía hasta que se encuentra el bloque.

Entre las instrucciones que se siguen interpretando después de una instrucción de prebúsqueda, que ha fallado al acceder a cache, puede haber instrucciones de prebúsqueda que también fallen al acceder a cache. En este caso, los fallos se pueden servir concurrentemente o en secuencia, pero siempre existe un número máximo permitido de fallos de cache pendientes de servicio. El servicio de fallos concurrentes es una característica de la jerarquía de memoria. Cuando se supera el límite de fallos permitidos, en algunos procesadores, la instrucción de prebúsqueda se convierte en una instrucción nop. Esto es, la instrucción finaliza la interpretación después de detectar el fallo.

Ejercicio

Supongamos una cache con un tamaño de bloque de 16 bytes y que la UP tarda 1 ciclo en interpretar cualquier instrucción. Cuando se produce un fallo de cache en el primer nivel se supone acierto en el segundo nivel, siendo la latencia de acceso 5 ciclos. Además, suponemos que los fallos se sirven en secuencia.

Seguidamente se muestra el código que se interpreta en el procesador. En la parte izquierda se muestra el código especificado en un lenguaje de alto nivel y en la parte derecha en lenguaje ensamblador. La variable s está almacenada en el registro r3 y cuando se empieza a ejecutar el bucle se producen fallos de carga al acceder a los elementos del vector A.

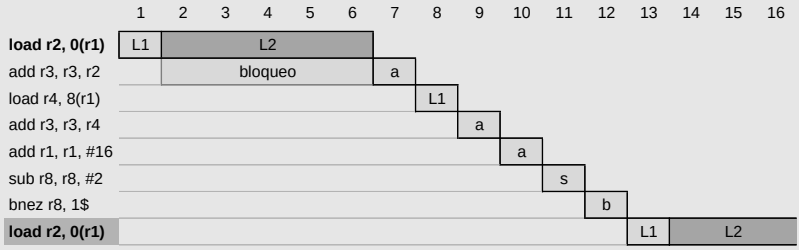
```
prebúsqueda A(1)                    PF 0 (r1)
do I =1, N, 2                 1$:   PF 16 (r1)
  prebúsqueda A(I+2)                load r2, 0 (r1)
  s = s + A(I)                      add r3, r3, r2
  s = s + A(I+1)                    load r4, 8(r1)
enddo                               add r3, r3, r4
                                    add r1, r1, #16
                                    sub r8, r8, #2
                                    bne r8, 1$
```

El símbolo PF representa una instrucción de prebúsqueda y el registro r1 almacena la dirección base del vector A cuando se empieza a ejecutar el código. El tamaño de los elementos del vector A es de 8 bytes.

Calcule el tiempo de ejecución del código anterior con instrucciones de prebúsqueda y sin ellas. En este último caso suponga que el código es el mismo pero sin las instrucciones de prebúsqueda.

Respuesta

Como el tamaño de bloque es 16 bytes y el tamaño de un elemento del vector A es 8 bytes se produce un fallo de cache cada dos referencias al vector A. En el siguiente diagrama temporal se muestra la ejecución de algunas instrucciones cuando no se utilizan las instrucciones de prebúsqueda, marcándose en negrita las instrucciones de acceso a memoria que fallan en cache.

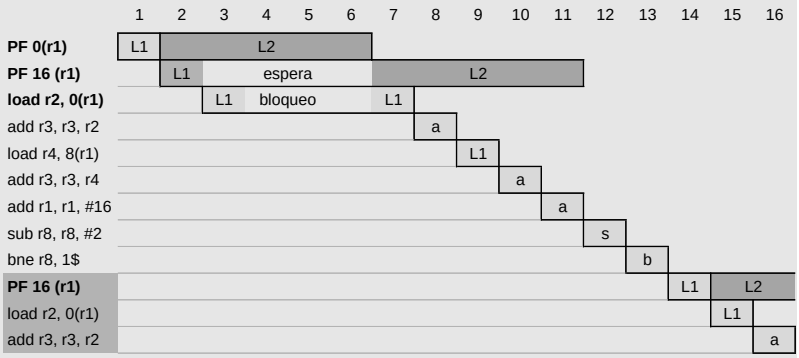


La instrucción add r3, r3, r2 debe esperarse a que se cargue el dato en el registro r2 antes de efectuar el cálculo. Se produce un fallo de cache en cada iteración del bucle y el número de instrucciones ejecutadas por iteración es 7. Entonces,

$$T_{\text{sin prebúsqueda}} = (7 + 5) \times (N/2) = 6 \times N$$

donde el valor 5 son los ciclos necesarios para acceder al siguiente nivel de la jerarquía de memoria.

En el siguiente diagrama temporal se muestra la ejecución cuando se utilizan las instrucciones de prebúsqueda.



La instrucción PF 16(r1) debe esperarse a que finalice el acceso a L2 de la instrucción previa, ya que los fallos de cache se gestionan en serie. La instrucción load r2, 0(r1) debe esperar a que el bloque que contiene el elemento al que quiere acceder esté almacenado en cache. Por tanto, se bloquea la interpretación de instrucciones hasta que finaliza la instrucción load.

Los datos del bloque que trae al primer nivel de cache la instrucción PF 16 (r1) son accedidos por las instrucciones load r2, 0(r1) y load r3, 8(r1) en la siguiente iteración.

Por iteración se ejecutan 8 instrucciones y la latencia de acceso al siguiente nivel de la jerarquía de memoria (PF 16 (r1)) se solapa en su totalidad con la interpretación de instrucciones de la misma iteración. Entonces,

$$T_{\text{con prebúsqueda}} = 5 + 8 \times (N/2) \sim 4 \times N$$

donde el valor 5 son los ciclos de la primera instrucción de prebúsqueda que no se solapan con la ejecución de la instrucción PF 16(r1).

Load no bloqueante

Otra mejora para incrementar el rendimiento es que las instrucciones load sean no bloqueantes. Esto es, la UP sigue interpretando instrucciones después de un fallo de cache, mientras no se interprete una instrucción que utilice como dato fuente el dato que se está esperando cargar en un registro.

Entre las instrucciones que se siguen interpretando, después de un fallo de cache, puede haber instrucciones load que fallen al acceder a cache. En este caso, proseguir la interpretación de

instrucciones es una característica del procesador, pero siempre existe un número máximo permitido de fallos de cache pendientes de servicio.

Ejercicio

En este ejercicio se utiliza la característica de load no bloqueante en lugar de las instrucciones de prebúsqueda utilizadas en el ejercicio previo. El código especificado en lenguaje de alto nivel y en ensamblador es el siguiente.

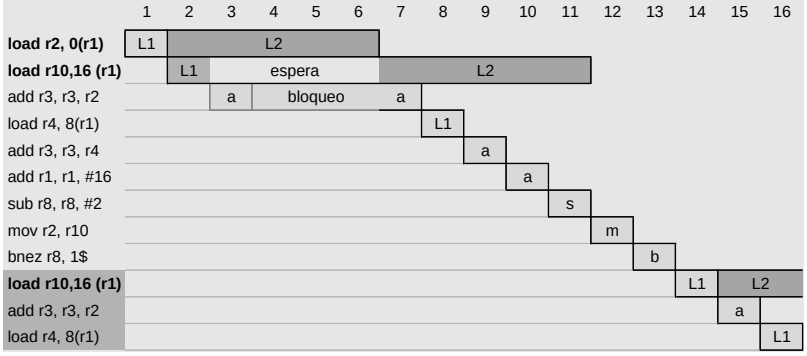
```
tmp = A(1)                          load r2, 0 (r1)
do I =1, N, 2                  1$:  load r10, 16 (r1)
   temp1 = A(I+2)                   add r3, r3, r2
   s = s + tmp                      load r4, 8(r1)
   s = s + A(I+1)                   add r3, r3, r4
   tmp = tmp1                       add r1, r1, #16
enddo                               sub r8, r8, #2
                                    mov r2, r10
                                    bne r8, 1$
```

Suponiendo las mismas condiciones que en el ejercicio previo calcule el tiempo de ejecución.

Respuesta

El siguiente diagrama temporal muestra la ejecución de algunas instrucciones, marcándose en negrita los casos en que hay fallo de cache. La instrucción load r10, 16(r1) debe esperar para acceder al segundo nivel de cache, ya que los fallos se serializan.

Como el load es no bloqueante el procesador sigue interpretando instrucciones. La siguiente instrucción es add r3, r3, r2, la cual debe esperarse a que se cargue el dato en el registro r2 para ejecutarse. Una vez se ha cargado el registro r2, se ejecuta la instrucción y se prosigue la interpretación de instrucciones.



La instrucción load r10, 16(r1) almacena un valor en el registro r10 que se utiliza al final del bucle (mov r2, r10). El otro dato del bloque que ha cargado en cache se lee por la instrucción load r4, 8(r1) en la siguiente iteración. En las siguientes iteraciones el fallo de cache de la instrucción load r10, 16(r1) se solapa con la ejecución de las instrucciones del bucle.

Por iteración se ejecutan 8 instrucciones y la latencia de acceso al siguiente nivel de la jerarquía de memoria (load r10, 16(r1)) se solapa en su totalidad con la interpretación de las 5 siguientes instrucciones de la misma iteración. Entonces,

$$T = 5 + 8 \times (N/2) \sim 4 \times N$$

donde el valor 5 son los ciclos de la primera instrucción load que no se solapan con la ejecución de la instrucción load r10, 16(r1).

Ancho de banda

Actualmente, cuando se requiere que el primer nivel de cache soporte varios accesos paralelos se utilizan memorias denominadas multipuerto. Esto es, en paralelo se pueden estar efectuando varios accesos a la misma posición de memoria.

Cuando en un nivel de la jerarquía se permite más de un fallo de cache pendiente de servicio, el servicio de los fallos es serie a menos que los siguientes niveles de la jerarquía de memoria y los buses que los interconectan soporten concurrencia.

Una posibilidad para soportar accesos concurrentes a la información almacenada en un nivel de la jerarquía de memoria es utilizar varios bancos de memoria con un único puerto de acceso por banco y acceso independiente a cada banco. Esta organización se denomina multibanco y permite servir de forma solapada accesos concurrentes si acceden a bancos distintos. En caso contrario deben serializarse y se dice que se ha producido un conflicto.

Para determinar el banco en el cual se almacena un dato se utiliza entrelazado por dirección. Usualmente se utilizan los bits menos significativos de la dirección para determinar el banco y el tamaño de la información que se almacena en un banco suele ser la palabra o el bloque de cache.

Ejercicio

Suponga una cache de segundo nivel con mapeo directo que se implementa utilizando 4 bancos de memoria. Para determinar la ubicación de los datos en los bancos se utilizan los bits menos significativos de la dirección y el tamaño de la información que se almacena en un banco es el bloque, el cual es de 32 bytes.

Determine el banco en que se almacena el contenido de las siguientes direcciones de memoria de tamaño 8 bytes: 0, 40, 72, 224.

Respuesta

En la siguiente figura se muestra la ubicación en los bancos del contenido de algunas direcciones de memoria alineadas a 8 bytes. En la figura se especifica la dirección de memoria y las direcciones correspondientes a un bloque se muestran en la misma fila.

banco 0	banco 1	banco 2	banco 3
0 8 16 24	32 40 48 56	64 72 80 88	96 104 112 120
128 136 144 152	160 168 176 184	192 200 208 216	224 232 240 248

El bloque al que pertenece una dirección se calcula mediante la siguiente expresión

$$\text{bloque} = \left\lfloor \frac{\text{dirección}}{\text{tamaño del bloque}} \right\rfloor$$

El banco donde se almacena un bloque se calcula mediante la siguiente expresión

$$\text{banco} = \text{bloque} \ \textbf{mod} \ (\text{número de bancos})$$

Entonces,

Dirección	Bloque	Banco
0	0	0
40	1	1
72	2	2
224	7	3

Supongamos una jerarquía de memoria con dos niveles de cache y memoria. En la Figura 1.17 se muestra un diagrama donde se muestra la interpretación de una secuencia de instrucciones load.

En la parte superior de la Figura 1.17 la jerarquía de memoria sirve los accesos en secuencia, ya que las cache son bloqueantes. La primera referencia no se encuentra en ningún nivel de cache y la quinta instrucción referencia un dato que se encuentra en el segundo nivel de cache. En la parte inferior de la Figura 1.17 se observa la concurrencia cuando se utilizan cache no bloqueantes y las instrucciones load son no bloqueante. Mediante la concurrencia en el acceso a los niveles de la jerarquía de memoria se incrementa el número de accesos por unidad de tiempo o ancho de banda, lo cual reduce la latencia efectiva observada.

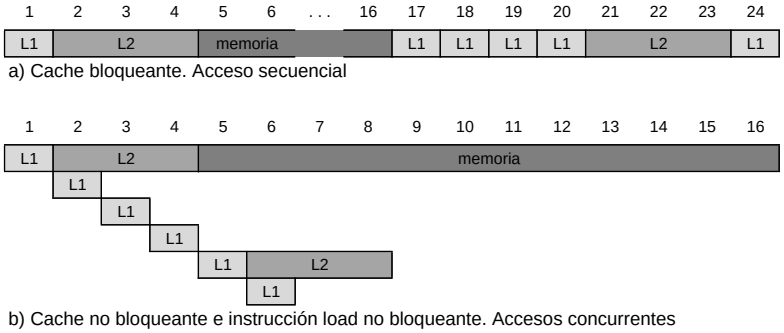


Figura 1.17 *Acceso en secuencia (a) y acceso concurrente (b) a los niveles de la jerarquía de memoria.*

Tendencia en la jerarquía de memoria

El aumento de la capacidad de integración ha permitido incluir en el chip niveles de la jerarquía y controladores de los niveles de la jerarquía externos al chip. Ello reduce la latencia media por referencia a memoria y facilita la implementación de accesos concurrentes a niveles distintos o al mismo nivel de la jerarquía de memoria. También, en algunos procesadores el acceso a los dos primeros niveles de la jerarquía de memoria se efectúa en paralelo. Este último diseño reduce la latencia efectiva al segundo nivel de cache en caso de fallo en el primer nivel de cache y acierto en el segundo nivel de cache.

Como ejemplo en la Figura 1.18 se muestra la evolución de los procesadores Intel en relación con la jerarquía de memoria y comunicación con el exterior del chip. Se ha pasado de una organización donde las caches son externas al chip y se accede en secuencia a los elementos de la jerarquía a una organización con los dos primeros niveles de la jerarquía incluidos dentro del chip y

un elevado ancho de banda para comunicarse con los elementos externos al chip (memoria, otros procesadores, E/S). La disponibilidad de caminos de acceso independientes y específicos para algunos niveles de la jerarquía de memoria permite acceder de forma concurrente a varios niveles de la jerarquía de memoria reduciéndose la latencia media de acceso observada (Pentium II y posteriores).

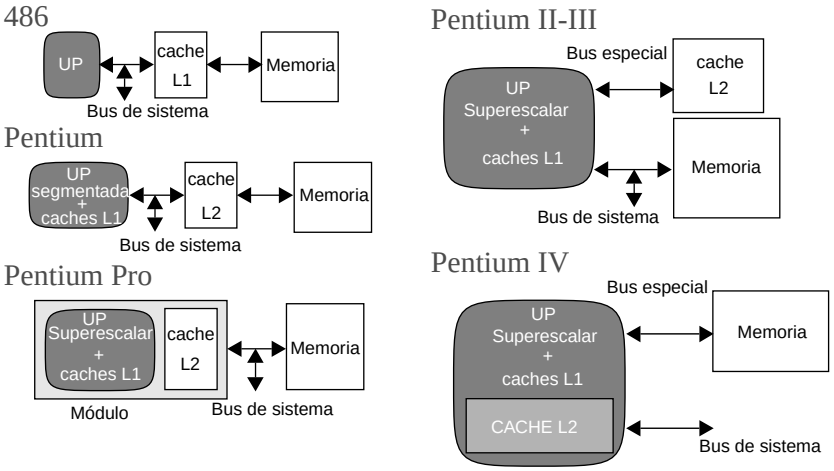


Figura 1.18 *Evolución de la jerarquía de memoria en la familia de procesadores Intel (1985 - 2000). En tramado oscuro se muestran los elementos incluidos en el chip, junto con el procesador. Caches L1 indica una cache de instrucciones y una cache de datos con caminos de acceso independientes. La palabra cache en singular indica una cache en la que se almacenan tanto instrucciones como datos.*

También como ejemplo, en la Figura 1.19 se muestra una tabla con la frecuencia de funcionamiento del bus de sistema que se utiliza para acceder a memoria en varios procesadores de Intel.

	frecuencia	año
Pentium II	66 Mhz	1998
Pentium III	133 Mhz	1999

	frecuencia	año
Pentium 4	400 Mhz	2000
Pentium 4 HT	533 Mhz	2002
Pentium 4 HT	800 Mhz	2003

Figura 1.19 *Frecuencia de transmisión de información en el bus de sistema que conecta el procesador con memoria.*

A partir del Pentium 4 se utiliza conexión punto a punto con memoria y se utilizan los dos flancos de la señal de reloj para transmitir información. En la frecuencia especificada en la tabla se tiene en cuenta esta característica.

TECNOLOGÍA DE FABRICACIÓN

En esta sección se describen algunas características que sustentan el progreso en el rendimiento de los procesadores.

La tecnología de fabricación utilizada actualmente es CMOS y el elemento básico es el transistor. Un proceso tecnológico de fabricación se identifica por la longitud de la puerta del transistor medida en micrómetros (10^{-6} m y denotada como µm) o nanómetros (10^{-9} m y denotado como nm). En la Figura 1.20 se muestra una tabla con las últimas generaciones tecnológicas utilizadas en el diseño de tres procesadores.

Año	89	90	91	92	93	94	95	96	97	98	99	00	01	02	03	04
Intel	1,0		0,8			0,6	0,35			0,25	0,18			0,13		0,09
Alpha				0,75			0,5			0,35			0,18			
UltraSPARC							0,5	0,35	0,28			0,18	0,15	0,13		

Figura 1.20 *Generaciones tecnológicas utilizadas en el diseño de los procesadores Intel, Alpha y UltraSPARC. La tecnología esta expresada en µm.*

Cada nueva generación tecnológica introduce mejoras significativas. Idealmente una nueva tecnología de fabricación escala por un factor de ~0.7 (30% de reducción) las dimensiones de los dispositivos (transistores) y cables (interconexiones). Estos escalados aportan beneficios que influyen en la frecuencia de funcionamiento, la tensión de alimentación, el tamaño del dado (chip) y la potencia consumida. La frecuencia de funcionamiento puede ser un 43% mayor y la capacidad efectiva equivalente total se reduce un 30%.

Potencia y energía

La energía se mide en unidades de trabajo (julios) y la potencia es la energía consumida por unidad de tiempo, expresada en vatios (julio/seg. o W). En la Figura 1.21 se muestra en un gráfico la relación entre energía y potencia. La potencia es el nivel de consumo y la energía es el área (nivel x tiempo).

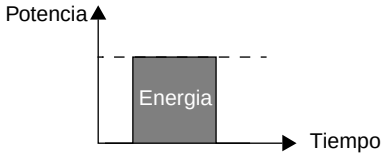


Figura 1.21 *Relación entre energía y potencia.*

La potencia consumida es importante por razones de disipación térmica. La energía consumida es importante por razones de coste de la energía o para incrementar el tiempo de vida de una batería.

Potencia

La potencia consumida en circuitos CMOS tiene tres componentes: conmutación, corriente de fugas y corriente de cortocircuito. El consumo de conmutación es debido a la conmutación entre niveles de tensión en la carga capacitiva efectiva de todo el chip. La corriente de fugas es debida a que los transistores implementados no son ideales. La corriente de cortocircuito es debida a que los dos transistores de un inversor CMOS están activos al mismo tiempo cuando la entrada cambia de nivel de tensión.

Un análisis simple del consumo de potencia en circuitos CMOS se basa en la aproximación de que la potencia consumida es debida enteramente a la carga y descarga de los nodos capacitivos del circuito (conmutación).

La potencia de conmutación se incrementa con la tensión de alimentación (V) y la frecuencia (f) de la siguiente forma:

$$\text{Potencia} = \text{Energia} \times f = C \times V^2 \times f$$

donde C es la capacidad efectiva equivalente que representan todos los dispositivos y cables del chip (dado) en un ciclo, y se mide en faradios.

En la Figura 1.22 se relacionan dependencias de los factores de la expresión que calcula la potencia consumida con la organización (microarquitectura), diseño de circuitos y proceso tecnológico.

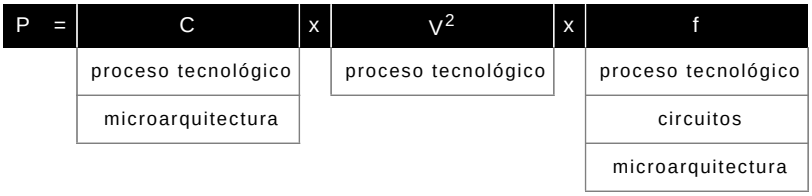


Figura 1.22 *Relación de dependencia de los factores implicados en el cálculo de la potencia.*

Teoría del escalado

En una transición entre generaciones tecnológicas la teoría de escalado ideal indica que las dimensiones de los dispositivos (transistores) y cables (interconexiones) se escalan por un factor de ~0.7 (30% de reducción).

Las variables t', C', V', f' y P' son respectivamente los valores de capacidad, tensión, frecuencia y potencia previos al escalado.

Como las dimensiones se han reducido un 30% (Figura 1.23):

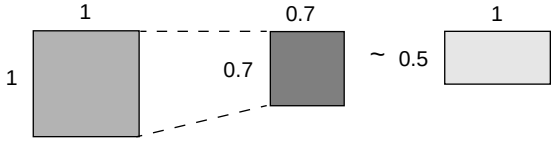


Figura 1.23 *Relación entre las dimensiones de un dispositivo en un cambio de generación tecnológica.*

- El retardo de puerta se reduce un 30% (t = 0.7 x t'). Por tanto, la frecuencia de operación puede ser un 43% mayor (1.43X ~ 1.5X).
- La capacidad efectiva equivalente se reduce un 30% (C = 0.7 x C').
- El tamaño del transistor es del orden de 2 veces menor en área (0.7 x 0.7). Por tanto, el número de transistores por unidad de área (densidad de transistores) se duplica.
- El área del dado (chip) se reduce a la mitad: 0.7 x 0.7 = 0.7^2, (A = 0.49 x A' ~ 0.5X A').

- Densidad de capacidad efectiva equivalente: C/Area = (0.7 x C') / (0.7 x 0.7 x A') = (1/ 0.7) x (C'/A') = 1.43 x (C'/A') ~ 1.5 x (C'/A'), donde C es la capacidad efectiva equivalente que representan todos los dispositivos y cables del microprocesador.

Consideramos dos situaciones distintas en lo que respecta a la tensión de alimentación.

- Escalado de campo eléctrico constante. Se escala la tensión de alimentación (se reduce un 30%, V = 0.7x V').

  La energía total consumida en un ciclo disminuye un 65% (E=P/f) y la potencia consumida se reduce del orden del 50%

$$\text{Potencia} = C \times V^2 \times f = 0.7 \times C' \times 0.7^2 \times V'^2 \times \frac{1}{0.7} \times f' \sim 0.5 \times P'$$

- Escalado de tensión constante. La tensión de alimentación permanece constante (V = 1 x V').

  La potencia permanece constante.

$$\text{Potencia} = C \times V^2 \times f = 0.7 \times C' \times 1 \times V'^2 \times \frac{1}{0.7} \times f' = 1 \times P'$$

Una nueva generación de fabricación se empieza a utilizar cada 2 o 3 años. Entonces, en cada nueva generación se esperan las siguientes mejoras potenciales (Figura 1.25):

- Contracción ideal (misma arquitectura): Utilizando el mismo número de transistores se incrementa la frecuencia y el rendimiento 1.5 veces, se reduce 2 veces el tamaño del dado y se reduce 2 veces la potencia consumida (Figura 1.23).
- Nueva generación ideal (nueva arquitectura y mismo tamaño del dado de silicio): Utilizando el doble de transistores se gana 3 (1.5 x 2) veces en rendimiento y 1.5 veces en frecuencia sin incrementar el tamaño del dado y la potencia consumida (Figura 1.24).

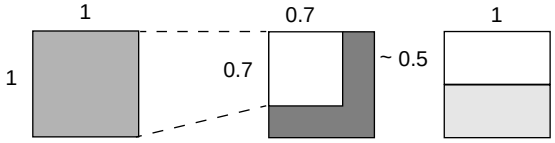


Figura 1.24 *Nueva generación ideal. Relación de tamaños.*

Características	Mejoras ideales	
	Contracción	Nueva generación
Número de transistores	1X	2X
Tamaño del dado	0.5X	1x
Frecuencia	1.5X	1.5X
Consumo de potencia	0.5X	1X
Rendimiento	1.5X	3X

Figura 1.25 *Mejoras ideales utilizando la teoría del escalado suponiendo campo eléctrico constante.*

En la práctica es necesario más de una generación tecnológica para los incrementos de rendimientos descritos y usualmente a mayor coste. Por ejemplo, se ha evaluado experimentalmente que la densidad de capacidad efectiva equivalente se incrementa del orden del 30% - 35%. Por tanto, en una nueva generación del procesador no se duplica por 2 la densidad de transistores.

Ejercicio

En la construcción de microprocesadores se han observado las siguientes tendencias cuando se pasa de una generación tecnológica a la siguiente.

	Tendencia observada	
Frecuencia	se dobla	2X
Tensión de alimentación	reduce un 30%	0.7X
Densidad de capacidad equivalente (capacidad/área)	incrementa un 30%	1.3X
Tamaño del lado del dado	incrementa un 25%	1.25X

En 1999 un procesador consume una potencia de 100W. ¿Cuál es la potencia consumida después de 4 generaciones tecnológicas?. Si en 1999 se consumen 70 amperios. ¿Cuántos amperios se consumen después de 4 generaciones tecnológicas?.

Respuesta

La expresión que calcula la potencia es

$$\text{Potencia} = C \times V^2 \times f = A \times \frac{C}{A} \times V^2 \times f$$

siendo A el área del dado.

Teniendo en cuenta los escalados y suponiendo que el dado es cuadrado, tenemos que en la siguiente generación

	generación de partida (f_0, V_0, C_0, A_0)
Frecuencia	$f = 2 \times f_0$
Tensión de alimentación	$V = 0.7 \times V_0$
Densidad de capacidad equivalente (capacidad/área)	$C/A = 1.3 \times (C_0/A_0)$
Area del dado	$A = 1.25^2 \times A_0$

Entonces, en la generación k la potencia consumida es

$$P_k = 1.25^{2\times k} \times A_0 \times \left(1.3^k \times \frac{C_0}{A_0}\right) \times (0.7^{2\times k} \times V_0{}^2) \times (2^k \times f_0)$$

Efectuando manipulaciones algebraicas

$$P_k = 1.99^k \times P_0$$

siendo $P_0 = C_0 \times V_0{}^2 \times f_0$ la potencia en 1999. Por tanto, después de 4 generaciones tecnológicas (k=4) la potencia consumida son 1568 W.

La intensidad de corriente se calcula como

$$I = \frac{P}{V} = \frac{1.99^4 \times P_0}{0.7^4 \times V_0} = 65.32 \times I_0$$

siendo I_0 = 70 amperios la corriente consumida en 1999. Sustituyendo tenemos que se consumen 4572 amperios.

Energía

Una batería suministra amperios por unidad de tiempo a una tensión preestablecida. Esto es, almacena un carga (Q = I x t) que suministra cuando se solicita. La carga máxima de una batería se identifica mediante la tensión y los amperios por unidad de tiempo.

$$\text{bateria} = \text{energía}$$

$$\text{bateria} = \text{amperios} \times \text{segundo} \times \text{voltios}$$

$$\text{bateria} = \frac{\text{coulombios}}{\text{segundo}} \times \text{segundo} \times \text{voltios}$$

$$\text{bateria} = \text{coulombios} \times \text{voltios}$$

Ejercicio

Un programa tiene un bucle que en cada iteración ejecuta 1 millón de instrucciones. El computador donde se ejecuta el programa funciona a una frecuencia de 700 Mhz y consume una potencia de 70 W (vatios). La batería que alimenta al procesador suministra 2.5 AH (amperios-hora) a 10 voltios.

a) ¿Cuántas iteraciones del bucle se pueden ejecutar antes de que la batería se quede sin carga, suponiendo que se ejecuta 1 instrucción en cada ciclo de procesador (CPI=1)?.

Supongamos que se reduce la frecuencia de funcionamiento del computador a 600 Mhz y el consumo de potencia se reduce a 60W.

b) ¿Cuántos ciclos tarda en descargarse la batería?.

c) ¿Cuántas iteraciones del bucle se pueden ejecutar?.

d) Compare los resultados obtenidos en las preguntas a) y c) y coméntelos.

Respuesta

a) La energía almacenada en la batería es

$$\text{Energía}|_{\text{bateria}} = \text{amperios} \times \text{segundo} \times \text{voltios}$$

$$\text{Energía}|_{\text{bateria}} = 2.5 \times 3600 \times 10 = 9 \times 10^{4} \text{ julios}$$

La energía consumida por el procesador en cada ciclo es

$$\text{Energía}|_{\text{procesador}} = \text{potencia} \times \text{tiempo en segundos}$$

$$\text{Energía}|_{\text{procesador}} = 70 \times \frac{1}{700 \times 10^{6}} = 10^{-7} \text{ julios}$$

Entonces, si se ejecuta 1 millón de instrucciones por iteración y cada instrucción tarda 1 ciclo en ejecutarse, el número de iteraciones del bucle que se pueden ejecutar antes de que se consuma la carga de la batería es

$$\text{Iteraciones} = \frac{\text{Energía}|_{\text{bateria}}}{\text{Energía}|_{\text{procesador}} \times \text{instrucciones}}$$

$$\text{Iteraciones} = \frac{9 \times 10^{4}}{10^{-7} \times 10^{6}} = 9 \times 10^{5}$$

b) Los ciclos que tarda en descargarse la batería es el cociente entra la carga de la batería y la energía que consume por ciclo el procesador.

$$\text{ciclos} = \frac{\text{Energía}|_{\text{bateria}}}{\text{Energía}|_{\text{procesador}}}$$

$$\text{ciclos} = \frac{9 \times 10^4}{60 \times \frac{1}{600 \times 10^6}} = 9 \times 10^{11}$$

c) El número de iteraciones del bucle que se pueden ejecutar es el cociente del tiempo que tarda en descargarse la batería y el número de instrucciones por iteración, ya que una instrucción tarda 1 ciclo en ejecutarse.

$$\text{Iteraciones} = \frac{\text{tiempo}}{\text{instrucciones}} = \frac{9 \times 10^{11}}{10^6} = 9 \times 10^5$$

d) En los dos casos el número de iteraciones que se puede ejecutar es el mismo. Ello es debido a que se incrementa el tiempo de ejecución del programa en el caso b)

$$\text{tiempo}|_{\text{iteración}} = \text{instrucciones} \times \text{CPI} \times \frac{1}{\text{frecuencia}}$$

$$\text{tiempo}|^{700\text{Mhz}}_{\text{iteración}} = 10^6 \times 1 \times \frac{1}{700 \times 10^6} = 1.43 \text{ milisegundos}$$

$$\text{tiempo}|^{600\text{Mhz}}_{\text{iteración}} = 10^6 \times 1 \times \frac{1}{600 \times 10^6} = 1.66 \text{ milisegundos}$$

La diferencia de tiempo es 0.23 milisegundos, durante los cuales se sigue consumiendo energía.

TENDENCIAS EN LA TECNOLOGÍA DE FABRICACIÓN

En este apartado se muestra la evolución de parámetros tecnológicos en familias de procesadores.

Frecuencia

En la Figura 1.26 se muestra la tendencia en frecuencia de tres familias de procesadores. Podemos decir que la tendencia es multiplicar por 2 la frecuencia de la generación tecnológica previa.

La superación de la previsión de la teoría del escalado (1.43) se debe a: a) la reducción del número de puertas empleadas en cada ciclo de reloj con un posible incremento de la segmentación y b) la utilización de técnicas de diseño de circuitos mejoradas que reducen el retardo medio de puerta por encima del 30% por generación tecnológica previsto por la teoría de escalado.

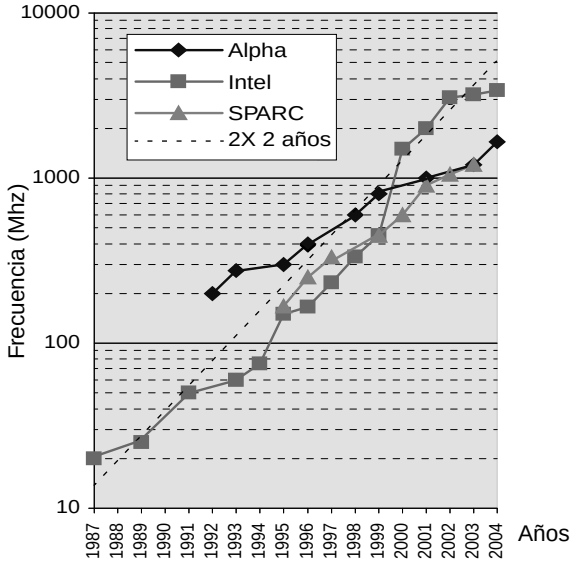


Figura 1.26 *Tendencia de la frecuencia en los procesadores Intel, Alpha y UltraSPARC.*

En la Figura 1.27 se muestra una tabla con los niveles de puerta por ciclo en los procesadores Alpha. Se observa una reducción del orden del 15% por generación en las tres primera generaciones.

Año	92	95	98	03
procesador	21064	21164	21264	21364
µm	0,75	0,5	0,35	0,18
frecuencia (Mhz)	200	300	600	1200
niveles de puertas	16	14	12	12

Figura 1.27 *Procesador Alpha. Niveles de puertas en un ciclo de reloj.*

En los procesadores Intel se ha pasado de 50 puertas por ciclo en el procesador 486 a 18 puertas por ciclo es el Pentium II. Las generaciones intermedias del procesador, Pentium y PentiumPro, utilizan 38 y 23 puertas por ciclo respectivamente.

Potencia

La potencia consumida por un chip depende de la tecnología de fabricación y del circuito que contiene y como se implementa. Esto es, entre otras cosas depende del tamaño del chip, del estilo de diseño de los circuitos lógicos, de la microarquitectura y de la frecuencia de operación.

Las generaciones tecnológicas hasta 0.8 µm han mantenido la tensión de alimentación (escalado de tensión constante). Posteriormente se ha escalado la tensión de alimentación (escalado del campo eléctrico constante) y con ello se ha reducido la potencia consumida. Actualmente por razones tecnológicas de fabricación (corriente de fugas) la tendencia de disminución se ha reducido. En Figura 1.28 se muestra la tendencia en la tensión de alimentación utilizada en los procesadores durante la última década.

Año	< 93	93	95	97	99	01	03
micras	0.7	0.5	0.35	0.25	0.18	0.13	0.09
voltios	5	3.3	2.3	1.8	1.5	1.2	1.05
tendencia de disminución	30%				15%		

Figura 1.28 *Tendencia de la tensión de alimentación.*

En la Figura 1.29 se muestra el consumo de potencia para varias familias de procesadores. Se observa que el consumo de potencia en cada familia de procesadores se ha incrementado en cada nueva generación del procesador. La línea discontinua conecta el primer procesador de cada generación (líder).

Las líneas continuas de la Figura 1.29 muestran la potencia consumida por nuevas versiones del procesador en un nuevo proceso tecnológico, con geometrías menores que el procesador líder de la generación (compactación).

En resumen, la reducción geométrica en conjunción con menores tensiones de alimentación dan lugar a consumos menores de potencia en las nuevas versiones del procesador. Sin embargo, cuando se utiliza un proceso tecnológico en una nueva generación del procesador se produce un incremento de la potencia consumida.

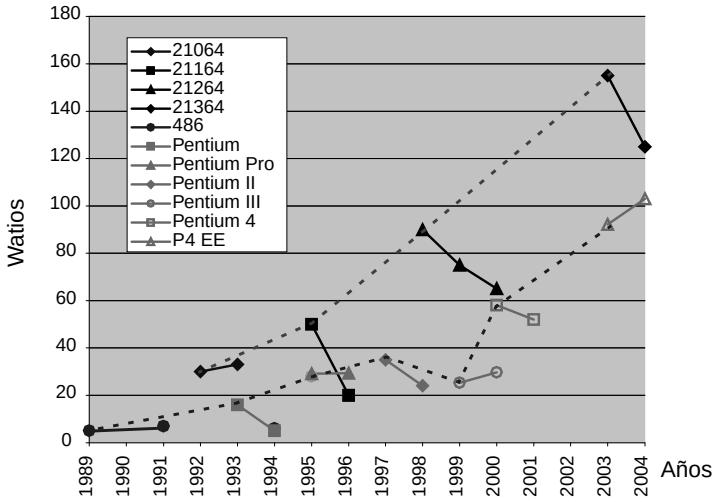


Figura 1.29 *Evolución de la potencia consumida en procesadores de las familias Alpha e Intel.*

La carga capacitiva que observa la señal de reloj es actualmente la componente mayor de consumo. Una celda o un registro representan una carga capacitiva y esta capacidad conmuta en cada

ciclo de reloj, consumiendo potencia aunque los datos de entrada no se modifiquen. En la Figura 1.30 se muestra la distribución del consumo de potencia en el procesador Alpha 21264.

	Reloj	Caches	U.F.	Control	E/S
Alpha 21264	40%	20%	15%	15%	10%

Figura 1.30 *Distribución del consumo de potencia.*

Tamaño del dado

Los diseñadores, además de aprovechar el incremento de densidad de los transistores, también incrementan el tamaño del dado. En la Figura 1.31 se muestran dos gráficas. En la gráfica de la izquierda se muestra la evolución en el número de transistores y en el tamaño del dado en los procesadores de las familias Alpha, Intel y SPARC. En la gráfica de la derecha se muestra el número total de transistores incluidos en el dado y la distribución de los mismos entre lógica y memoria. Observe que en los últimos años el incremento de capacidad de integración se utiliza para incluir en el chip niveles de la jerarquía de memoria.

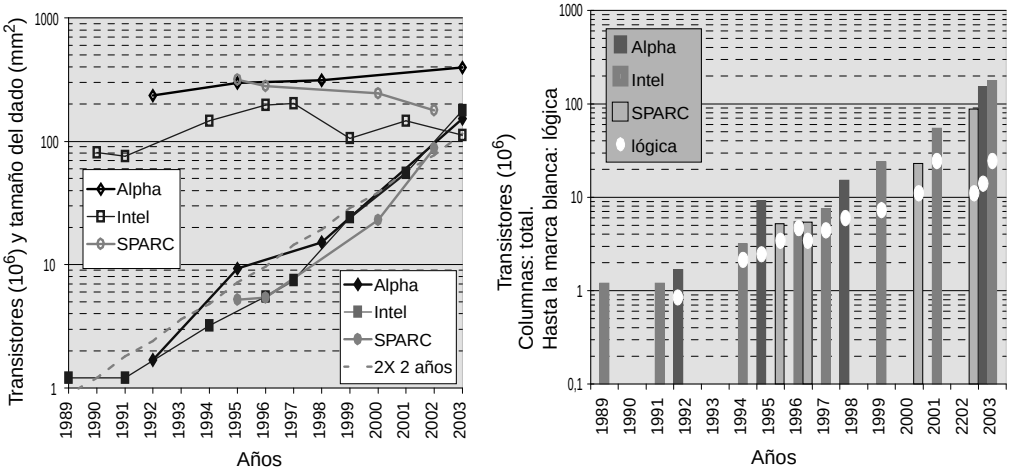


Figura 1.31 *Número de transistores, tamaño del dado y distribución de los transistores entre lógica y memoria en procesadores de las familias Alpha, Intel y SPARC.*

MÉTRICAS QUE RELACIONAN RENDIMIENTO Y POTENCIA

En la Figura 1.32 se muestra el rendimiento, la potencia consumida y el tamaño del dado de procesadores de la familia Intel relativo al procesador i486. Se observa que para obtener un incremento de rendimiento de 6X se ha incrementado la potencia consumida en 18X y el tamaño del dado en 15X.

En la década de los 80 la potencia consumida no se tomaba en consideración (< 10 W). En la década de los 90 se diseñaba optimizando el rendimiento y posteriormente se optimizaba la potencia consumida. A partir de 2000 la potencia consumida es el punto de partida y entonces se optimiza el rendimiento. Seguidamente se presenta un análisis que relaciona la potencia consumida con el rendimiento.

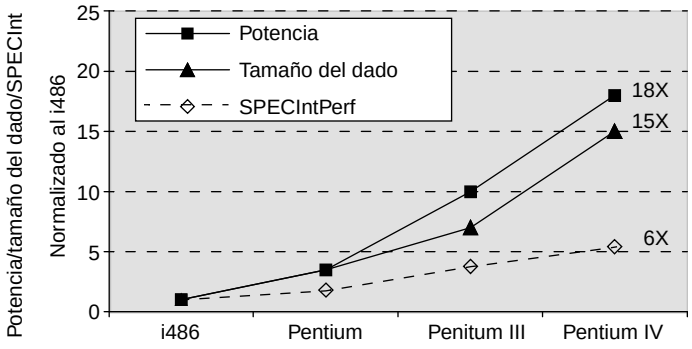


Figura 1.32 *Potencia, tamaño del dado y rendimiento de procesadores de la familia Intel relativos al procesador i486.*

La potencia consumida tiene dos costes. Uno de ellos es la disipación térmica y el otro el suministro de la potencia, mediante una tensión de alimentación y una intensidad de corriente.

- El coste de disipación térmica esta asociado con mantener los dispositivos dentro del rango de temperatura de funcionamiento especificado y mantener la integridad del envoltorio (package) que encapsula el chip. Además, el calor generado debe disiparse desde la fuente sin un coste excesivo. La potencia disipada por el chip por unidad de área se mide en vatios/cm^2 y se denomina densidad de potencia). La densidad de potencia puede limitar el progreso en rendimiento debido a limitaciones tecnológicas en la disipación térmica.

- El suministro de potencia y su distribución también tienen un coste significativo. La potencia requerida por un componente del chip debe ser suministrada mediante una tensión de alimentación preestablecida y con una intensidad de corriente suficiente para que funcione correctamente el dispositivo. Consumos elevados de potencia con tensiones de alimentación reducidas requieren niveles de intensidad de corriente elevados. Otro factor a tener en cuenta en el diseño es la variabilidad en la intensidad de corriente solicitada en ciclos cercanos. Todos estos factores combinados requieren bastante atención y representan un coste significativo.

En un computador portátil es preferible hablar de energía que de potencia, ya que las baterías tienen almacenada una cantidad fija de carga (E = V x Q). Entonces, decir que un procesador consume menos que otro puede dar lugar a interpretaciones erróneas. En otras palabras, aun tardando más tiempo en realizar un cálculo, la energía consumida puede ser la misma. Por tanto, la batería se descarga en la misma cantidad de energía en ambos casos. En estas condiciones es mejor una figura de mérito que relacione la energía y las operaciones efectuadas.

Métricas de eficiencia

La idea es que la figura de mérito evalúe la eficiencia entre la potencia consumida y el rendimiento. Una métrica sencilla es

$$\frac{\text{MIPS}}{\text{W}} = \frac{\text{millones de instrucciones por segundo}}{\text{vatios}}$$

Con esta métrica, cuanto mayor es el valor del cociente mejor es el computador.

Dado que actualmente existe un amplio espectro en la utilización de los procesadores, la métrica MIPS/W no es suficiente para caracterizar los objetivos establecidos en el diseño de un procesador.

En el segmento de procesadores denominado de gama alta el objetivo es incrementar el rendimiento dentro de unas restricciones de consumo de potencia. En contraposición, en el segmento denominado de bajo consumo el objetivo es la duración de la carga de la batería o tiempo de vida de la batería, teniendo en cuenta unas restricciones de rendimiento.

Entonces, en función de la utilización del procesador, interesa que en la métrica utilizada se dé más peso al rendimiento o a la potencia consumida. En la Figura 1.33 se muestran métricas en función de la utilización de un computador.

Gama de bajo consumo (portátiles)	Gama media (estaciones de trabajo)	Gama de alto rendimiento (servidores)
MIPS/W	$MIPS^2/W$	$MIPS^3/W$

Figura 1.33 *Figuras de mérito en función del segmento de consumo del procesador*

La inversa de las métricas de la Figura 1.33 también se utiliza. La métrica W/MIPS es una forma de evaluar el producto potencia-retardo (power-delay product, PDP) y evalúa la energía consumida por ciclo o conmutación. Esta métrica tiene las mismas unidades que la energía ya que representa la energía media consumida por ciclo.

$$\frac{W}{MIPS} = \frac{julios}{seg} \times \frac{seg}{instrucciones} = \frac{julios}{instrucción}$$

La métrica $W/MIPS^2$ es una forma de evaluar el producto energía-retardo (energy-delay product, EDP). EDP es la energía media consumida multiplicada por el tiempo de cálculo requerido (PDP x T) y tiene en cuenta que se puede incrementar el retardo reduciendo la energía consumida por operación.

De forma similar, la métrica $W/MIPS^3$ evalúa el producto energía-$retardo^2$ (ED^2P). La expresión CPI^3 x W también pertenece a este último tipo de métricas.

Ejercicio

Deduzca que la métrica $rendimiento^3$/potencia es invariante a la tensión de alimentación. Para ello suponga que la frecuencia es proporcional de forma lineal a la tensión de alimentación.

Respuesta

Como la frecuencia es proporcional a la tensión de alimentación

$$f \sim K \times V$$

el rendimiento también es proporcional a la tensión de alimentación

$$R \sim IPC \times f \sim IPC \times K \times V$$

La potencia se evalúa como

$$P \sim C \times V^2 \times f = C \times K \times V^3$$

Entonces, la potencia es proporcional al cubo de la tensión de alimentación. Por tanto, la métrica rendimiento^3/potencia es independiente de la tensión de alimentación.

$$\frac{R^3}{P} \sim \frac{IPC^3 \times f^3}{C \times V^2 \times f} = \frac{IPC^3 \times K^3 \times V^3}{C \times K \times V^3} \sim IPC^3$$

Ejercicio

En un cambio de generación tecnológica calcule el producto energía-retardo para los casos: a) escalado de tensión constante y b) escalado de campo eléctrico constante.

Respuesta

a) La energía y el retardo son

$$E = C \times V^2 = 0.7 \times C' \times V'^2 = 0.7 \times E' \qquad \text{retardo} = 0.7 \times r'$$

Entonces el producto energía-retardo es

$$E \times \text{retardo} = 0.7^2 \times E' \times r' \sim 0.5 \times E' \times r'$$

b) La energía y el retardo son

$$E = C \times V^2 = 0.7 \times C' \times 0.7^2 \times V'^2 = 0.7^3 \times E' \qquad \text{retardo} = 0.7 \times r'$$

Entonces el producto energía-retardo es

$$E \times \text{retardo} = 0.7^4 \times E' \times r' \sim 0.25 \times E' \times r'$$

El escalado de campo eléctrico constante da el menor producto energía-retardo.

Consumo reducido

El objetivo es maximizar el tiempo de vida de una batería ya que la carga acumulada es fija. La energía se puede expresar mediante la siguiente relación.

$$E = T \times P = \frac{1}{R} \times P$$

donde E es la energía, P es la potencia, T el tiempo de ejecución y R el rendimiento.

Entonces, para mejorar el tiempo de vida de una batería, el beneficio de rendimiento debe ser mayor que la potencia adicional consumida. Por ejemplo, si el rendimiento se incrementa un 10%, pero la potencia consumida es un 10% superior, el tiempo de vida de la batería es el mismo.

Una ejecución lenta consume menos potencia en un periodo largo de tiempo, mientras que un paralelismo agresivo reduce el tiempo de actividad pero incrementa la potencia consumida (Figura 1.34).

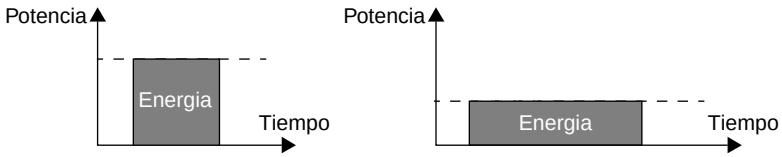


Figura 1.34 *La energía consumida depende del tiempo y de la potencia. En la gráfica la altura representa la potencia y el área tramada en oscuro es la energía.*

EJEMPLOS

En los siguientes ejemplos se calculan métricas relativas a una carga de trabajo y se generaliza la ley de Amdahl para el caso de varios modos de funcionamiento. También se muestra la tendencia de consumo de potencia en los procesadores. Así mismo, se analizan técnicas de reducción de consumo de potencia y se relaciona el consumo de potencia de un propuesta arquitectónica con otra posibilidad tecnológica que permite obtener el mismo incremento de rendimiento.

Rendimiento medio de una carga de trabajo

En ocasiones en lugar de disponer del tiempo de ejecución como métrica se dispone del rendimiento (1 / T).

Pregunta 1: *Desarrolle una expresión que permita evaluar el rendimiento de una carga de trabajo a partir de los rendimientos individuales. Esta expresión calcula la media armónica de los rendimientos.*

Respuesta: La siguiente expresión evalúa el rendimiento de una carga de trabajo

$$R = 1/T = 1/\left(\sum_{\text{programas}} f_i \times T_i\right)$$

donde f_i es el peso del programa i en la carga de trabajo y T_i es el tiempo de ejecución.

Efectuando manipulaciones algebraicas obtenemos:

$$R = 1/T = 1/\left[\sum_{\text{programas}} f_i \times \left(\frac{1}{(1/T_i)}\right)\right] = \frac{1}{\sum_{\text{programas}} f_i \times \frac{1}{R_i}}$$

donde R_i es el rendimiento del programa i.

Número medio de instrucciones por ciclo de una carga de trabajo

De los programas incluidos en una carga de trabajo se conoce el número de ciclos por instrucción (IPC = 1 /CPI) de cada uno de ellos y el peso relativo de cada uno de ellos en la carga de trabajo.

Pregunta 1: *Calcule el IPC medio de la carga de trabajo.*

Respuesta: El tiempo medio de ejecución de una carga de trabajo es

$$T = \sum_{\text{programas}} f_i \times T_i = \sum_{\text{programas}} f_i \times \left(N_i \times \frac{1}{IPC_i} \times t_c \right)$$

donde f_i es el peso del programa i en la carga de trabajo, N_i es el número de instrucciones, IPC_i es el IPC medio y t_c es el tiempo de ciclo. Efectuando manipulaciones algebraicas obtenemos

$$T = \sum_{\text{programas}} f_i \times \left(N_i \times \frac{1}{IPC_i} \times t_c \right) = \sum_{\text{programas}} N \times f_i \times \left(\frac{N_i}{N} \times \frac{1}{IPC_i} \times t_c \right)$$

El número ponderado de instrucciones de la carga de trabajo es

$$N = \sum_{\text{programas}} f_i \times N_i$$

Como

$$T = N \times \frac{1}{IPC} \times t_c$$

tenemos que el IPC medio es igual a

$$IPC = 1 / \left(\sum_{\text{programas}} f_i \times \frac{N_i}{N} \times \frac{1}{IPC_i} \right)$$

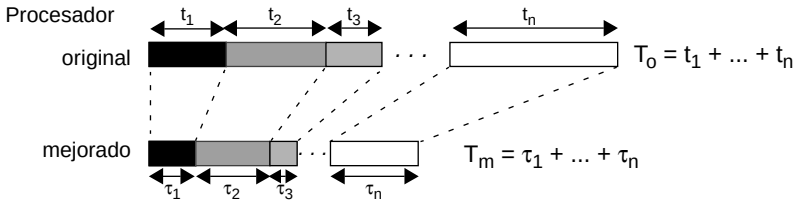
Esto es, hay que calcular la media armónica de los IPC de cada programa ponderado por la proporción de instrucciones de cada programa y el peso de cada programa en la carga de trabajo.

Generalización de la ley de Amdahl

Al mejorar un procesador se pueden plantear distintas mejoras que se utilizan individualmente en intervalos de tiempo disjuntos al ejecutar un programa.

Pregunta 1: *Generalice la ley de Amdahl para el caso de que existan distintas mejoras que se utilizan individualmente en intervalos de tiempo disjuntos al ejecutar un programa. Suponga que todas las fracciones de tiempo experimentan una mejora.*

Respuesta: En la siguiente figura se muestra la proporción de tiempo que en un programa se puede utilizar la mejora i. Los instantes de tiempo en que se utiliza cada mejora se han agrupado de forma contigua (t_i).



Sea $f_i = t_i/T_o$ la fracción de tiempo, en el procesador original, donde se utiliza la mejora i, cumpliéndose que $\sum_i f_i = 1$.

Sea $g_i=t_i/\tau_i$ la ganancia cuando se utiliza la mejora i.

Entonces,

$$\text{Ganancia} = \frac{T_o}{\sum_i \tau_i} = \frac{1}{\sum_i \frac{\tau_i}{T_o}} = \frac{1}{\sum_i \frac{t_i}{g_i \times T_o}} = \frac{1}{\sum_i \frac{f_i}{g_i}}$$

La ley de Amdahl generalizada se puede utilizar para evaluar el rendimiento de un procesador segmentado. En este tipo de procesadores cuando se produce un riesgo se suspende la segmentación en algún grado para que finalice la instrucción que produce el riesgo.

La ganancia ideal cuando se segmenta la interpretación de instrucciones es el número de etapas. Sin embargo, cuando se produce un riesgo la ganancia ideal queda mermada ya que se inhibe parcialmente o totalmente el solapamiento en la interpretación de instrucciones. Esta merma se denomina penalización respecto a la segmentación ideal. Entonces, si N es la concurrencia ideal, N - K es la concurrencia cuando se produce un riesgo, siendo K la penalización de la concurrencia.

Supongamos que un procesador se ha segmentado en 5 etapas y las instrucciones que producen riesgos son de tipo secuenciamiento y de tipo load. Las instrucciones de secuenciamiento dan lugar a que se suspenda totalmente la interpretación segmentada (N - K= 1) y las instrucciones load tienen una penalización de K = 1 ciclo.

Pregunta 2: *Suponga que en un programa las instrucciones de secuenciamiento representan un 10% de las instrucciones interpretadas y las instrucciones load representan un 20%. Calcule la ganancia del procesador segmentado respecto una interpretación serie de las instrucciones. Para ello, suponga que en el procesador serie todas las instrucciones tardan el mismo tiempo en interpretarse.*

Respuesta: En la ley de Amdahl generalizada f_i es la proporción de veces que se produce el riesgo i y g_i es la ganancia respecto a una interpretación serie de la instrucción. Un 70% de las instrucciones no producen riesgos y la ganancia es N = 5. Un 10% de las instrucciones requiere una interpretación secuencial (5 - 4 = 1). El restante 20% se interpreta con una ganancia de 5 - 1 = 4.

$$\text{Ganancia} = \frac{1}{\frac{0.1}{5-4} + \frac{0.2}{5-1} + \frac{0.7}{5}} = 3.45$$

Entonces, la ganancia es 3.45X en lugar de 5X que sería el caso ideal sin riesgos.

Jerarquía de memoria

El cociente entre el tiempo de acceso a memoria y tiempo de ciclo de la UP tiende a incrementarse cada vez más. Por tanto, la penalización en un fallo de cache también aumenta.

Una técnica para reducir la penalización por fallo de cache es la utilización de varios niveles de cache. La idea es que el primer nivel de cache se acomode al tiempo de ciclo de la UP y al ancho de banda requerido. Sin embargo, ello limita el tamaño del primer nivel de cache e incrementa los fallos.

En este contexto, el objetivo del segundo nivel de cache es reducir el número de acceso a memoria o en otras palabras, reducir los fallos que deben servirse desde memoria. Por tanto, se reduce la penalización en un fallo de cache de primer nivel, ya que la latencia de acceso al segundo nivel de cache es bastante menor que la latencia de acceso a memoria.

Pregunta 1: *Desarrolle una expresión que calcule el tiempo medio de acceso en ciclos de UP (t_c) en una jerarquía de memoria con dos niveles de cache. Suponga que el acceso a la cache de segundo nivel se inicia después de detectar un fallo en el primer nivel de cache, lo cual requiere que haya transcurrido la latencia de acceso a la cache de primer nivel.*

Respuesta: El número medio de ciclos de UP de un acceso a memoria se evalúa mediante la siguiente expresión.

$$CMA = c_{L1} + f_{L1} \times P_{L1}$$

donde c_{L1} es la latencia en ciclos de UP cuando se acierta en la cache de primer nivel, f_{L1} mide los fallos de cache por acceso a memoria de la UP en el primer nivel de cache y P_{L1} es la penalización en ciclos de UP en cada fallo de cache de primer nivel.

La penalización P_{L1} de los fallos en el primer nivel depende de los aciertos y fallos en el segundo nivel de cache y de la penalización en caso de fallo en el segundo nivel, lo cual requiere acceder a memoria.

$$P_{L1} = c_{L2} + f_{L1\text{-}L2} \times P_{L2}$$

donde c_{L2} es la latencia en ciclos de UP cuando se acierta en la cache de segundo nivel, $f_{L1\text{-}L2}$ mide los fallos de cache de segundo nivel por fallo de cache en la cache de primer nivel y P_{L2} es la penalización en ciclos de UP en cada fallo de cache de segundo nivel.

Sustituyendo,

$$CMA = c_{L1} + f_{L1} \times (c_{L2} + f_{L1\text{-}L2} \times P_{L2})$$

$$CMA = c_{L1} + f_{L1} \times c_{L2} + f_{L1} \times f_{L1\text{-}L2} \times P_{L2}$$

Sean

$$f_{L2} = f_{L1} \times f_{L1\text{-}L2}$$

los fallos de cache de segundo nivel por referencia a memoria de la UP. Entonces,

$$CMA = c_{L1} + f_{L1} \times c_{L2} + f_{L2} \times P_{L2}$$

donde en el cálculo de las frecuencias de fallo f_{L1} y f_{L2} se utiliza el mismo denominador, las referencias a memoria de la UP.

Pregunta 2: *Calcule la penalización por acceso a memoria.*

Respuesta: La penalización por acceso a memoria son los ciclos adicionales a la latencia de acierto en la cache de primer nivel en el número medio de ciclos por acceso (CMA).

$$\text{penalización}|_{\text{acceso}} = f_{L1} \times c_{L2} + f_{L2} \times P_{L2}$$

Pregunta 3: *Suponiendo que R_I son los accesos a memoria por instrucción, calcule la penalización por instrucción.*

Respuesta: Multiplicando la penalización por acceso por los accesos por instrucción obtenemos la penalización por instrucción.

$$\text{penalización}|_{\text{instrucción}} = \text{penalización}|_{\text{acceso}} \times R_I$$

$$\text{penalización}|_{\text{acceso}} \times R_I = R_I \times f_{L1} \times c_{L2} + R_I \times f_{L2} \times P_{L2}$$

Sean f_I los fallos por instrucción y f_R los fallos por acceso a memoria en un determinado nivel de cache. Entonces,

$$f_I = R_I \times f_R$$

Sustituyendo en la expresión que calcula la penalización obtenemos:

$$\text{penalización}|_{\text{instrucción}} = f_{IL1} \times c_{L2} + f_{IL2} \times P_{L2}$$

La jerarquía de memoria de un computador tiene las siguientes características:

	ciclos de UP
penalización por fallo en L2	200
latencia de acierto en L2	12
latencia de acierto en L1	1

En el computador se ha ejecutado un conjunto de programas de pruebas y se ha medido que se efectúan 1.3 accesos a memoria por instrucción, siendo 0.07 y 0.02 respectivamente las frecuencias de fallo por instrucción en L1 y L2.

Pregunta 4: *Calcule la penalización por instrucción y por acceso a memoria.*

Respuesta: Sustituyendo en la expresión que calcula la penalización por instrucción obtenemos:

$$\text{penalización}|_{\text{instrucción}} = 0.07 \times 12 + 0.02 \times 200 = 4.84 \text{ ciclos}$$

Utilizando la relación entre penalizaciones obtenemos:

$$\text{penalización}|_{\text{acceso}} = \text{penalización}|_{\text{instrucción}} / R_I$$

$$\text{penalización}|_{\text{acceso}} = 4.84 / 1.3 = 3.72 \text{ ciclos}$$

Recursos en un procesador

Un procesador superescalar dispone de una unidad funcional para efectuar sumas en coma flotante y la frecuencia de reloj es 250Mhz. Este procesador puede iniciar en cada ciclo la ejecución de una instrucción de coma flotante y de una instrucción de acceso a memoria, además de una operación de aritmética entera y una instrucción de secuenciamiento.

La métrica que utilizamos para evaluar el rendimiento son millones de operaciones en coma flotante por segundo (MFLOPS).

Teniendo en cuenta la frecuencia de funcionamiento y que el procesador puede ejecutar una instrucción de coma flotante en cada ciclo, el número máximo de MFLOPS es: 250 MFLOPS. Un rendimiento próximo al anterior se mide cuando se ejecuta el siguiente código y los elementos del vector A están almacenados en cache antes de iniciar la ejecución del bucle.

```
do I =1, N, 2
   s = s + A(I)
enddo
```

En el procesador se han ejecutado los siguientes tres códigos, midiéndose los MFLOPS indicados en la siguiente tabla. Cuando se han tomado las medidas los vectores accedidos estaban almacenado en cache.

A)	B)	C)
do I =1, N, 2 C(I) = A(I) + B(I) enddo	do I =1, N, 2 A(I) = B(I) + B(I) enddo	do I =1, N, 2 A(I) = A(I) + B(I) enddo
80 MFLOPS	120 MFLOPS	80 MFLOPS

Como ayuda para contestar a la siguiente pregunta suponga que al ejecutar el bucle A) la traza dinámica de instrucciones que se ejecuta es la siguiente. Para simplificar la traza se han eliminando las instrucciones de control del bucle y de cálculo de direcciones que no influyen en los MFLOPS medidos.

ciclo	instrucción 1	instrucción 2
1	load A(1)	--
2	load B(1)	--
3	add C(1), A(1), B(1)	load A(2)
4	store C(1)	--
5	load B(2)	--
6	add C(2), A(2), B(2)	load A(3)
7	store C(2)	--

Pregunta 1: *Calcule, a partir de los recursos disponibles y el código que se ejecuta, un valor de MFLOPS y compruebe si el valor medido se corresponde con el valor calculado.*

Respuesta:

A) Una iteración del bucle requiere 3 accesos a memoria y sólo se puede efectuar un acceso a memoria por ciclo. Por tanto, el rendimiento que puede obtenerse es

250 MFLOPS / 3 accesos = 83.3 MFLOPS

el cual es aproximadamente el valor medido.

B) En cada iteración del bucle se efectúan 2 accesos a memoria, ya que los elementos de la variable B sólo se leen una vez. Entonces el rendimiento que puede obtenerse es

250 MFLOPS / 2 accesos = 125 MFLOPS

el cual es aproximadamente el valor medido.

C) En cada iteración del bucle se efectúan 3 accesos a memoria. Entonces el rendimiento que puede obtenerse es

250 MFLOPS / 3 accesos = 83.3 MFLOPS

el cual es aproximadamente el valor medido.

Selección entre opciones de diseño

En un procesador las instrucciones load y store representan un 26% de las instrucciones y tienen un CPI de 2.3 y el CPI medio de todas las otras instrucciones es 1.5. En la próxima generación del procesador, debido a un cambio de proceso tecnológico, se puede disponer de más transistores y se plantean dos opciones en la forma de utilizar estos transistores adicionales.

- A) Utilizar los transistores en mejorar todos los componentes del procesador en el chip, lo cual permite reducir el periodo del reloj en un 10%.
- B) Incrementar el tamaño de la cache de 1-nivel, lo cual reduce el CPI de las instrucciones load y store un 20%, ya que se supone que no se incrementa la latencia de acceso a cache.

Pregunta 1: *¿Qué opción permite obtener un procesador con mayor rendimiento?*

Respuesta: La opción de diseño A disminuye el tiempo de ciclo en un 10%. Entonces, el tiempo de ejecución se evalúa como

$$T_A = 0.9 \times T_o$$

El CPI del procesador original es

$$CPI|_o = CPI|_{LS} \times f_{LS} + CPI|_{resto} \times (1 - f_{LS})$$

donde $CPI|_{LS}$ y f_{LS} son respectivamente el CPI y la frecuencia de las instrucciones load y store, $CPI|_{resto}$ y $(1\text{-}f_{LS})$ son respectivamente el CPI y la frecuencia del resto de instrucciones. Sustituyendo tenemos

$$CPI|_o = 2.3 \times 0.26 + 1.5 \times 0.74 = 1.71$$

Entonces

$$T_A = 0.9 \times T_o = 0.9 \times N \times CPI_o \times t_o = 0.9 \times N \times 1.71 \times t_o$$
$$= 1.54 \times N \times t_o$$

La opción de diseño B disminuye el CPI de las instrucciones de acceso a memoria en un 20%. Entonces,

$$CPI_B = CPI|_{LS} \times f_{LS} \times FD_{LS} + CPI|_{resto} \times (1 - f_{LS})$$

donde FD_{LS} es el factor de reducción. Sustituyendo tenemos

$$CPI_B = 2.3 \times 0.26 \times 0.8 + 1.5 \times 0.74 = 1.59$$

Entonces

$$T_B = N \times CPI_B \times t_o = 1.59 \times N \times t_o$$

La mejor opción es la que da lugar a un menor tiempo de ejecución. Por tanto, la opción A es la mejor.

Pregunta 2: *¿Cuál es la ganancia de la opción con mayor rendimiento?.*

Respuesta: La ganancia se calcula mediante la siguiente expresión

$$\text{Ganancia} = \frac{T_o}{T_A} = \frac{T_o}{0.9 \times T_o} = \frac{1}{0.9} = 1.11$$

Entonces, la opción A es 1.11X mejor o un 11% mejor.

Tendencias de potencia y área del dado

En la construcción de microprocesadores se han observado las siguientes tendencias cuando se pasa de una generación tecnológica a la siguiente.

	Tendencia observada	Teoría del escalado
Frecuencia	2X	1.5X
Tensión de alimentación	0.8X	0.7X
Capacidad equivalente	0.7X	0.7X
Potencia	0.9X	0.5X
Densidad de potencia	1.3X - 1.8X	1X
Area	0.5X	0.5X

Como generación de partida supondremos 0.25 μm con un tamaño del lado del dado de 15 mm (cuadrado) y una potencia de 66 W. Las siguientes generaciones tecnológicas son 0.18 μm, 0.13 μm y 0.1 μm.

Suponemos que el incremento de frecuencia sigue la teoría del escalado y la tensión de alimentación la tendencia observada.

Pregunta 1: *¿Cuál es la potencia de conmutación y la densidad de potencia (W/cm^2) si el tamaño (o área) del dado se mantiene constante a lo largo de 4 generaciones?.*

Respuesta: La expresión que calcula la potencia es

$$\text{Potencia} = C \times V^2 \times f$$

Se supone que el tamaño del dado se mantiene constante. Esta hipótesis sólo afecta a la capacidad en la expresión que evalúa la potencia. La capacidad es directamente proporcional al área del dado.

$$C \sim A$$

La teoría del escalado supone una reducción del 50% en área y una reducción de capacidad del 30%. Entonces, como se mantiene constante el área del dado, la capacidad en la generación k es

$$C_k = 2^k \times 0.7^k \times C_0$$

ya que el área del dado es el doble del que indica la teoría del escalado.

En estas condiciones, la potencia consumida en la generación k es

$$P_k = (2^k \times 0.7^k \times C_0) \times (0.8^{2\times k} \times V_0^{\,2}) \times (1.5^k \times f_0)$$

ya que el incremento de frecuencia sigue la teoría del escalado y la tensión de alimentación la tendencia observada.

Agrupando factores y sustituyendo la potencia consumida en la generación de partida tenemos (P_0 = 66).

$$P_k = 1.344^k \times C_0 \times V_0^{\,2} \times f_0 = 1.344^k \times P_0 = 1.344^k \times 66$$

Efectuando cálculos para valores de k en el rango [0..3], siendo k=0 la generación de partida obtenemos los siguientes resultados. Como el área del dado se mantiene constante, para evaluar la densidad de potencia se utiliza el área del dado de la generación de partida (15 x 15 = 225 mm^2).

Generación	k	Potencia (W)	Densidad de potencia (W/cm^2)
0.25	0	66	29.33
0.18	1	88.70	39.42
0.13	2	119.22	52.99
0.1	3	160.23	71.21

Pregunta 2: *¿Cuál es el tamaño del dado si la potencia consumida se mantiene constante a lo largo de 4 generaciones?. ¿Cuál es la densidad de potencia?.*

Respuesta: Para el cálculo utilizaremos como referencia el área de la generación de partida (A_0). Entonces, sea FA_k la fracción del área original en la generación k.

$$A_k = FA_k \times A_0$$

Teniendo en cuenta esta expresión y que la capacidad es directamente proporcional al área del dado, la potencia consumida en la generación k es

$$P_k = FA_k \times P'_k$$

donde P'_k es la potencia consumida en la generación k, suponiendo que, entre generaciones, se mantiene constante el tamaño del dado. Este valor se ha calculado en la pregunta anterior.

$$P'_k = 1.344^k \times P_0$$

Sustituyendo tenemos

$$P_k = FA_k \times 1.344^k \times P_0$$

Como se pide mantener la potencia constante

$$P_0 = P_k = FA_k \times 1.344^k \times P_0$$

Despejando

$$FA_k = \frac{1}{1.344^k} = 0.744^k$$

Efectuando cálculos para valores de k en el rango [0..3], siendo k=0 la generación de partida, obtenemos los siguientes resultados.

Generación	FA_k	Area (cm^2)	Lado (mm)	Densidad de potencia (W/cm^2)
0.25	1	2.25	15	29.33
0.18	0.744	1.67	12.9	39.52
0.13	0.553	1.25	11.2	52.80
0.1	0.412	0.93	9.6	70.97

El número de transistores es proporcional al área del dado. La teoría del escalado indica que en cada generación tecnológica el número de transistores por unidad de área se duplica. En cambio la tendencia histórica observada es que el área del dado se incrementa un 14%.

Pregunta 3: *¿Cuál es el incremento en el número de transistores de un chip: A) cuando se sigue la tendencia observada de incremento del dado, y: B) cuando se mantiene constante la potencia consumida?.*

Respuesta: El número de transistores es proporcional al área del dado

$$TR \sim A$$

La teoría del escalado indica que el área del dado se reduce un 50% por generación tecnológica.

A) Se sigue la tendencia observada: el área del dado se incrementa un 14% en cada generación.

$$A_k = 1.14^k \times A_0$$

siendo A_0 el área de la generación tecnológica de partida.

Entonces, el número de transistores en la generación k cumple la siguiente relación

$$TR_k|_{observada} \sim 1.14^k \times 2^k \times TR_0 = 2.28^k \times TR_0$$

donde TR_0 es el número de transistores en la generación de partida y el factor 2^k es debido a que al pasar de una generación a otra la teoría del escalado indica que el área se reduce un 50%.

Por tanto, el incremento de transistores es del 128% por generación.

B) Se mantiene la potencia constante

Utilizando el resultado de la pregunta anterior tenemos que la fracción FA_k del área de partida (A_0) en la generación k es

$$FA_k = 0.744^k \sim 0.75^k$$

lo cual representa una reducción de área de un 25% por generación

$$A_k = FA_k \times A_0 = 0.75 \times A_{k-1}$$

Entonces, el número de transistores en la generación k cumple la siguiente relación

$$TR_k|_{constante} \sim 0.75^k \times 2^k \times TR_0 = 1.5^k \times TR_0$$

Por tanto, el incremento de transistores es del 50% por generación.

En resumen, cuando se mantiene constante la potencia, el incremento de transistores es de un 50% por generación, en lugar del más de 100% que representaría la tendencia histórica.

Escalado dinámico tensión/frecuencia

En un chip la potencia consumida debida a la conmutación de los transistores se determina mediante la siguiente expresión.

$$\text{Potencia} = C \times V^2 \times f$$

Además, en un rango limitado de valores de la tensión de alimentación, ésta y la frecuencia están relacionadas por la siguiente expresión lineal.

$$f \sim K \times V$$

Esto es, dado un diseño, si se escala la tensión de alimentación, la carga capacitiva es la misma y la frecuencia máxima de funcionamiento es proporcional a la tensión de alimentación.

Pregunta 1: *Desarrolle expresiones que relacionen la potencia exclusivamente con la tensión de alimentación o con la frecuencia.*

Respuesta: Sustituyendo la expresión que relaciona la tensión de alimentación con la frecuencia en la expresión de la potencia obtenemos

$$\text{Potencia} \sim C \times K \times V^3 \sim \frac{C}{K^2} \times f^3$$

La potencia varia de forma cúbica con la tensión de alimentación o la frecuencia.

Las expresiones desarrolladas en la pregunta anterior indican el método más simple y eficiente de reducir la potencia disipada en un procesador diseñado para funcionar a una frecuencia elevada.

Esto es, variando la tensión y/o la frecuencia mediante control software se pueden ajustar dinámicamente los requisitos de rendimiento requeridos (R = f / [N x CPI]). De esta forma se ahorra energía cuando no se necesita el máximo rendimiento. Esto es, se incrementa el tiempo de cálculo pero se reduce la energía consumida.

Pregunta 2: *Calcule la reducción del consumo de potencia cuando se reduce la tensión de alimentación un 10%.*

Respuesta: Una reducción del 10% en la tensión de alimentación representa una reducción de potencia de un 27% (0.73X).

$$\frac{C \times K \times 0.9^3 \times V^3}{C \times K \times V^3} = 0.73$$

El rendimiento de un procesador medido en MIPS (millones de instrucciones por segundo) se evalúa mediante la siguiente expresión.

$$R = \frac{N}{T} = IPC \times f$$

Pregunta 3: *Relacione el rendimiento con la tensión de alimentación y calcule el impacto que tiene en el rendimiento una reducción de un 10% en la tensión de alimentación.*

Respuesta: Utilizando la expresión que relaciona la tensión de alimentación y la frecuencia tenemos que el rendimiento es proporcional a la tensión de alimentación.

$$R \sim IPC \times f \sim IPC \times V$$

Entonces, una reducción del 10% en la tensión de alimentación representa una reducción en rendimiento del 10% (0.9X).

$$\frac{IPC \times 0.9 \times V}{IPC \times V} = 0.9$$

Pregunta 4: *Desarrolle una expresión que calcule la energía relacionando la potencia y el rendimiento. Calcule la reducción de energía cuando se reduce la tensión de alimentación un 10%.*

Respuesta: La energía consumida es la potencia consumida durante un periodo de tiempo.

$$E \sim P \times \frac{1}{R}$$

Sustituyendo la potencia y el rendimiento por las expresiones que las relacionan con la tensión de alimentación obtenemos la siguiente expresión.

$$E \sim \frac{C \times K \times V^3}{IPC \times K \times V} = C \times \frac{V^2}{IPC} \sim V^2$$

La reducción de energía es del 19%.

$$\frac{0.9^2 \times V^2}{V^2} = 0.81$$

Alternativas de microarquitectura y potencia

El objetivo en un procesador de elevado rendimiento es maximizar el rendimiento teniendo en cuenta restricciones de potencia consumida debido a cuestiones de disipación térmica.

En un rango limitado de valores de la tensión de alimentación se cumplen las siguientes expresiones, que relacionan la tensión de alimentación (V), la potencia (P) y el rendimiento con la frecuencia (f).

$$f \sim V \qquad P \sim f^3 \qquad R \sim IPC \times f$$

donde IPC son instrucciones por ciclo.

Pregunta 1: *Calcule el incremento de potencia consumida, relativo a la potencia original, cuando se incrementa la tensión de alimentación* $V + \Delta V$.

Respuesta: Supongamos un incremento de la tensión de alimentación $V + \Delta V$. Esto representa un incremento de la frecuencia $f + \Delta f$ ya que $f \sim V$ y por tanto un incremento de la potencia.

$$\frac{(P + \Delta P) - P}{P} \sim \frac{(f + \Delta f)^3 - f^3}{f^3}$$

Aproximando el primer término del numerador por

$$(f + \Delta f)^3 \sim f^3 + 3 \times f^2 \times \Delta f$$

y después de manipulaciones algebraicas, tenemos

$$\frac{\Delta P}{P} \sim 3 \times \frac{\Delta f}{f}$$

Pregunta 2: *Calcule, de forma relativa, el impacto en el rendimiento cuando se incrementa la tensión de alimentación* $V + \Delta V$.

Respuesta: Un incremento en la tensión de alimentación representa un incremento lineal en la frecuencia. Entonces,

$$\frac{(R + \Delta R) - R}{R} \sim \frac{IPC \times (f + \Delta f) - IPC \times f}{IPC \times f}$$

$$\frac{\Delta R}{R} \sim \frac{\Delta f}{f}$$

Pregunta 3: *Calcule la relación entre el incremento de potencia y el incremento de rendimiento cuando se utiliza escalado tensión-frecuencia.*

Respuesta: Tenemos que

$$\frac{\Delta P}{P} \sim 3 \times \frac{\Delta f}{f} \qquad \text{y} \qquad \frac{\Delta R}{R} \sim \frac{\Delta f}{f}$$

Entonces,

$$\left(\frac{\Delta P}{P}\right) \Big/ \left(\frac{\Delta R}{R}\right) \sim 3$$

Una alternativa de microarquitectura que gana rendimiento y reduce el consumo será mejor que simplemente utilizar un escalado de tensión/frecuencia.

Supongamos una alternativa a nivel de microarquitectura (IPC) que incrementa el rendimiento.

$$\frac{\Delta R}{R} \sim \frac{\Delta IPC}{IPC}$$

El razonamiento que se solicita en la siguiente pregunta es en el mismo margen de valores en el cual se puede aplicar escalado tensión-frecuencia.

Pregunta 4: *Desde el punto de vista de consumo de potencia, deduzca cuándo una mejora arquitectónica debe descartarse, ya que incrementando la tensión de alimentación se obtiene el mismo rendimiento y el consumo de potencia es el mismo.*

Respuesta: De las expresiones previas que tienen en cuenta el escalado tensión/frecuencia, tenemos

$$\frac{\Delta P/P}{\Delta R/R} \sim 3$$

Esto es, la proporción entre incremento de potencia y rendimiento es 3.

Entonces, concluimos que una característica arquitectónica es interesante, desde el punto de vista del consumo de potencia, si la relación entre el incremento relativo de potencia y el incremento relativo de rendimiento es menor que 3. En otras palabras, si el incremento relativo de potencia es menor que 3 veces el incremento de rendimiento.

Por tanto, una alternativa arquitectónica debe descartarse si el incremento relativo de potencia es mayor que 3 veces el incremento de rendimiento.

EJERCICIOS

Ejercicio 1.1

El computador C1 con

- procesador de 500 MHz de frecuencia de reloj
- cache de datos de 32 KB, mapeo directo, bloque de 32B, 60 ns de penalización en caso de fallo
- cache de instrucciones ideal

ejecuta el programa

```
int i;                          //en registro
float s;                        //en registro, un operando float ocupa 8B
float X [10000];

for ( i=0; i<10000; i++)
      { s = s + X[i] }
```

que el compilador traduce en 5 instrucciones de lenguaje máquina.

El programa tarda 300 microsegundos en ejecutarse.

Pregunta 1: *Calcule: MFLOPS (millones de operaciones en coma flotante por segundo), MIPS, CPI.*

El tiempo de ejecución T_{ex} se puede expresar como

$$T_{ex} = T_{calc} + T_{mem}$$

donde T_{mem} representa el tiempo que el procesador espera para acceder a memoria cuando se falla en la cache de datos.

Pregunta 2: *Calcule la fracción (m) de accesos a datos que son fallo de cache y T_{mem}.*

Mejoras tecnológicas permiten implementar el procesador y las caches con un tiempo de ciclo de reloj de 1.5 ns. Ahora bien, la organización del procesador y del subsistema de memoria es idéntica a la del computador C1.

Pregunta 3: *Calcule el tiempo de ejecución T'_{ex} del programa en el nuevo computador y la ganancia.*

Ejercicio 1.2

En un procesador las instrucciones Load y Store representan un 26% de las instrucciones siendo 2.3 el CPI. Todas las demás instrucciones tienen 1.5 de CPI.

Pregunta 1: *Calcule el CPI del procesador original.*

En la próxima generación del procesador, debido a un cambio en el proceso tecnológico, se puede disponer de más transistores y se plantean dos opciones en la forma de utilizar estos transistores adicionales.

1. Incrementar el tamaño de la cache de primer nivel, lo cual reduce en un 20% el CPI de las instrucciones Load y Store, porque no se incrementa la latencia y la tasa de fallos es menor.
2. Utilizar los transistores en mejorar todos los componentes del procesador en el chip, lo cual permite reducir el periodo del reloj en un 10%.

Pregunta 2: *Calcule el CPI de la 1ª opción de diseño.*

Pregunta 3: *Calcule el tiempo de ejecución en la 2ª opción de diseño.*

Pregunta 4: *¿Cuál de las dos opciones permite obtener un procesador con mayor rendimiento?*

Pregunta 5: *¿Cuál es la ganancia, respecto del procesador original, de la mejor opción?*

Ejercicio 1.3

Deduzca 1) la cantidad de MIPS, 2) el CPI medio y 3) el tiempo total para cada una de las tres preguntas siguientes.

Pregunta 1: *Procesador con reloj de 900 MHz que ejecuta 1000 iteraciones de un bucle de 12 instrucciones y tarda 18 ciclos en cada iteración.*

Pregunta 2: *Procesador con reloj de 450 MHz que ejecuta 2000 iteraciones de un bucle de 12 instrucciones y tarda 18 ciclos en cada iteración.*

Pregunta 3: *Procesador con reloj de 900 MHz que ejecuta 2000 iteraciones de un bucle de 12 instrucciones y tarda 36 ciclos en cada iteración.*

Sabemos que un procesador emplea la mitad de su tiempo en hacer accesos a cache. Queremos reducir el tiempo total del procesador en un 20% del original.

Pregunta 4: *Deduzca cuanto más rápida debería ser la nueva cache para conseguirlo, expresando la respuesta como el valor del cociente: tiempo cache vieja / tiempo cache nueva*

Ejercicio 1.4

Un programa de prueba P contiene 20 millones de operaciones en coma flotante.

Una estación de trabajo E tiene un procesador a 90 MHz, incluye un coprocesador numérico y usa un compilador con optimizaciones. Este compilador permite que E efectúe los cálculos coma flotante bien en modo 1 (habilitando el coprocesador), o bien en modo 2 (inhabilitando el coprocesador y usando rutinas software).

La ejecución de P tarda 1.5 s cuando E funciona en modo 1, mientras que tarda 13.6 s en modo 2.

Se ha medido que CPI = 5 en modo 1, mientras que CPI = 3 en modo 2.

Pregunta 1: *Deduzca la velocidad en MIPS para cada uno de los dos modos.*

Pregunta 2: *Deduzca el número total de instrucciones ejecutadas en cada uno de los dos modos.*

Pregunta 3: *En el modo 2, deduzca el promedio de instrucciones necesarias para ejecutar una operación coma flotante.*

Pregunta 4: *En el modo 1, deduzca la velocidad en MFLOPS.*

Se puede modificar el modo 1, cambiando el coprocesador por otro que es capaz de acelerar F veces la ejecución de las operaciones responsables del 30% del tiempo total de ejecución de P.

Pregunta 5: *Deduzca el valor que debe tener F para que el tiempo de ejecución se reduzca a la mitad.*
Deduzca cuál sería el nuevo tiempo de ejecución si sabemos que F = 2.

Ejercicio 1.5

Un procesador A tiene 4 tipos de instrucciones: Enteras (ENT), accesos a memoria (MEM), coma flotante (CF) y de secuenciamiento (SEC). Para un programa determinado P, la siguiente tabla muestra la distribución de instrucciones y el CPI observado para cada tipo de instrucción. Conocemos que la probabilidad de ruptura del secuenciamiento implícito es del 60%.

Instrucción	sobre el total de instrucciones	CPI
ENT	30%	1
MEM	20%	10
CF	25%	8
SEC	25%	4 si salta; 1 si no salta

Pregunta 1: *Calcule el CPI medio del programa P al ejecutarse en el procesador A.*

Conocemos que la frecuencia de reloj del procesador A es de 400MHz y que el programa P ejecuta 2000 millones de instrucciones.

Pregunta 2: *¿Cuál es el tiempo de ciclo (tc) en nanosegundos?. ¿Cuál es el tiempo de ejecución (Texe) del programa P en segundos?.*

Pregunta 3: *¿Cuál es el porcentaje de tiempo que representa la ejecución de las instrucciones de CF?.*

La técnica de segmentación aplicada a la unidad funcional de coma flotante permite mejorar el CPI de las instrucciones CF.

Pregunta 4: *¿Cuál debería de ser la ganancia en el CPI de las instrucciones de CF para que el tiempo de ejecución total sea un 80% del tiempo original?.*

En un procesador B que incorpora la mejora anterior en la unidad funcional de coma flotante, el programa P tarda 16 segundos en ejecutarse.

Pregunta 5: *¿Cuál es la frecuencia de reloj del procesador B?.*

Ejercicio 1.6

Para evaluar las prestaciones de dos computadores se utiliza un banco de pruebas P, que es un conjunto de programas que ejecutan un total de 300 millones de operaciones que se reparten de este modo:

Operaciones	sobre el total de operaciones
suma o resta de enteros	20%
multiplicación o división de enteros	1%
suma o multiplicaciones de coma flotante	25%
divisiones de coma flotante	4%
otras que no son cálculos	50%

El computador A tiene un juego de instrucciones que son de tres clases:

clase	descripción	CPI
A1	Suma/Resta enteros	1
A2	Mult/Div enteros	2
A3	Mem, Branch, otras que no son de cálculo	3

Cuando tiene que hacer cualquier operación coma flotante, el computador A debe ejecutar 12 instrucciones de clase A1.

Se ha medido que el computador A obtiene, al ejecutar P, una velocidad de 3 MFlops.

El computador B tiene un juego de instrucciones que son de cuatro clases:

clase	descripción	CPI
B1	Cálculo enteros	3
B2	Suma/Mult coma flotante	4
B3	Div coma flotante	18
B4	Mem, Branch, otras que no son de cálculo	2

Se ha medido que el computador B obtiene, al ejecutar P, una velocidad de 29 MFlops.

Calcule, justificando todas las respuestas, los siguientes valores para cada uno de los dos computadores:

Pregunta 1: *Número de instrucciones ejecutadas (N).*

Pregunta 2: *CPI medio (CPI).*

Pregunta 3: *Tiempo de ejecución (T) en segundos.*

Pregunta 4: *Frecuencia de reloj ó inversa del tiempo de ciclo (F) en megahercios.*

Pregunta 5: *Velocidad medida en Mips (V).*

Pregunta 6: *Tiempo de ejecución empleado en: a) cálculo con enteros (TE), b) cálculos coma flotante (TR), c) no cálculos (TN).*

Ejercicio 1.7

Dos procesadores, PR1 y PR2, interpretan un mismo repertorio con 4 tipos de instrucciones: A, B,C y D

El PR1 tiene reloj de 0.9 GHz y se implementa de manera que consigue, para cada tipo de instrucción, IPC(A) = 1.0, IPC(B) = 0.5 IPC(C) =1/3 e IPC(D) = 0.25.

El PR2 tiene reloj de 0.75 GHz y se implementa de manera que consigue, para cada tipo de instrucción, IPC(A) = IPC(B) = 0.5 e IPC(C) = IPC(D) = 0.25.

Pregunta 1: *Calcule el IPC medio, y el CPI medio, del PR1 al interpretar un código que tiene equitativamente repartida la cantidad de instrucciones de cada tipo.*

Pregunta 2: *Calcule la velocidad en Mips del PR2 cuando interpreta un código con 50% de tipo A, 25% de tipo B, 20% de tipo C y 5% de tipo D.*

Se ha medido la velocidad relativa en Mips de ambos procesadores, cuando interpretan un código que no tiene ninguna instrucción de tipo D, y se ha encontrado que
Mips(PR1) / Mips(PR2) = 8 / 5.

Pregunta 3: *Calcule un reparto de cada tipo A, B y C de instrucciones en ese código.*

El PR1 tarda 10 s. en interpretar un programa Pa. Una versión mejorada del PR1 es el PR1++, que tarda la mitad de ese tiempo en interpretar el mismo Pa, gracias a que sus accesos a memoria son 4 veces mas rápidos que los del PR1.

Pregunta 4: *Calcule el tiempo que emplean PR1 y PR1++ en todo el proceso que no es de accesos a memoria.*

Pregunta 5: *El PR1 tarda 9 s. en interpretar otro programa Pb. Calcule el tiempo que emplea el PR1 en accesos a memoria, sabiendo que el PR1++ obtiene una ganancia sobre el PR1 que es el 75% de la ganancia ideal máxima que podría obtener, al interpretar el Pb.*

Sabemos que el PR2 está en un portátil alimentado por una batería que entrega 5 A. por hora, a una tensión de 8 V. y que interpreta ininterrumpidamente, una y otra vez, el código de la pregunta 2, agotando esa batería en 2 horas.

Pregunta 6: *Calcule cuantas instrucciones ha interpretado el PR2 hasta agotar su batería.*

Pregunta 7: *Calcule la potencia, en vatios, que consume el PR2. Calcule el valor de la capacidad efectiva equivalente del PR2.*

Ejercicio 1.8

Un procesador interpreta instrucciones de 3 tipos: enteras (ENT), accesos a memoria (MEM) y saltos (BR). La frecuencia de reloj del procesador es de 800 MHz. El primer nivel de la jerarquía de memoria está integrado en el mismo chip del procesador y consta de una cache de instrucciones y una cache de datos (CD). Para reducir la latencia media de accesos a memoria, el procesador dispone de una cache (segundo nivel) externa.

En este ejercicio consideraremos que la cache de instrucciones y la cache de segundo nivel son ideales (no hay fallos) y que las instrucciones MEM son bloqueantes. La penalización por fallo en la cache de datos es de 12 ciclos.

El procesador ejecuta un programa P. La siguiente tabla muestra la distribución de instrucciones y el CPI:

Tipo de instrucción	frecuencia	CPI
ENT	40%	1
MEM acierto en CD	20%	2
MEM fallo en CD	20%	14
BR	20%	2

Pregunta 1: *Calcule el IPC medio, CPI, la cantidad de MIPS, fallos en CD por instrucción, fallos en CD por acceso a memoria al ejecutar el programa P.*

Mejoras tecnológicas permiten integrar la cache de segundo nivel en el chip (reduciendo la latencia de acceso) y aumentar la frecuencia de reloj a 1 GHz. En el nuevo diseño se mantiene la microarquitectura original. La potencia consumida por el chip es de 50 w (vatios). La batería que alimenta el chip tiene una capacidad energética de 25 wh (vatios x hora).

Al ejecutar el programa P en el nuevo diseño se obtiene un CPI medio igual a 3.

Pregunta 2: *Calcule el número máximo de instrucciones de P para que el procesador pueda ejecutar completamente el programa sin recargar la batería.*

Pregunta 3: *Calcule el CPI de las instrucciones MEM que fallan en CD y deduzca la penalización (en ciclos) por fallo.*

Ejercicio 1.9

Un procesador con las siguientes características

- frecuencia de reloj de 1.5 GHz, que interpreta 1 instrucción por ciclo pero que bloquea la interpretación de instrucciones mientras se sirve un fallo de cache.
- tensión de alimentación a 2 voltios, consume 55 vatios de potencia cuando interpreta instrucciones y cuando el procesador está bloqueado consume 10 vatios.
- cache de datos con 30 ciclos de penalización en caso de fallo y cache de instrucciones ideal.

ejecuta las $3x10^9$ instrucciones de un programa, de las cuales 1/3 son instrucciones de acceso a memoria (M). Conocemos que el 5% de M son fallo en la cache de datos.

Pregunta 1: *Calcule el tiempo de ejecución del programa, el CPI medio y los MIPS obtenidos.*

Pregunta 2: *Calcule la energía total consumida por el procesador para ejecutar el programa y la potencia media.*
Calcule la carga mínima de la batería (Coulombios) que alimenta al procesador para poder ejecutar el programa.

Mejoras tecnológicas en el proceso de fabricación plantean dos alternativas para conseguir una ganancia del 20% en el tiempo de ejecución:

a. aumentar la capacidad de la cache de datos, manteniendo la frecuencia de reloj del procesador original.
b. mejorar todos los componentes del procesador para incrementar la frecuencia de reloj. La capacidad de la cache de datos no se modifica.

Pregunta 3: *Calcule la reducción de fallos de cache en la alternativa a).*

Pregunta 4: *Deduzca la frecuencia de reloj en la alternativa b)*

Ejercicio 1.10

Al interpretar un programa se mide la siguiente distribución de instrucciones y CPI asociado.

Tipo de instrucción	distribución	CPI
store	15%	1
load	25%	2
secuenciamiento	15%	4
aritméticas, enteros	35%	1
desplazamiento aritmético	5%	1
multiplicación	5%	10

Pregunta 1: *Calcule el CPI.*

Una optimización efectuada por el compilador reemplaza operaciones complejas o difíciles por operaciones más simples, con el objetivo de reducir el tiempo de ejecución. Un ejemplo es reemplazar una multiplicación por una constante por una secuencia de instrucciones de desplazamiento aritmético y de suma.

En el programa anterior se ha medido que el 50% de las multiplicaciones se pueden convertir en secuencias de instrucciones de desplazamiento aritmético y de suma, siendo el número medio de instrucciones de la secuencia igual a 3.

Pregunta 2: *Calcule el incremento relativo en el número de instrucciones.*

Pregunta 3: *Calcule el CPI cuando se interpreta el código optimizado. Previamente calcule la distribución de instrucciones.*

Pregunta 4: *Calcule la ganancia que aporta la optimización de código. Efectúe el cálculo utilizando la ley de Amdahl. Previamente calcule la fracción de tiempo en la cual se utiliza la mejora.*

Ejercicio 1.11

Un procesador con frecuencia de reloj de 2 GHz ejecuta el siguiente código:

```
for ( i = 0; i < 4096; i++)      // s ocupa 8 bytes y está asignada a un registro
    {                            // los vectores X e Y están alineados a bloque de
    s = s + X[i] * Y[i]          // memoria.
                                 // cada elemento de X e Y ocupa 8 bytes.
    }                            // los valores de s, X e Y se interpretan en coma
                                 // flotante
```

El procesador dispone de una cache de datos (L1) de 32 KBytes, 2 asociativa, tamaño de bloque de 64 Bytes, penalización de 128 ciclos cuando se ha de obtener el dato de memoria principal. Cuando se produce un fallo en la cache, el procesador bloquea la interpretación de las instrucciones hasta que se completa el servicio del fallo. La cache de instrucciones no produce fallos cuando se buscan las instrucciones del código. Suponga que los vectores X e Y no están cargados en la cache de datos L1 cuando se inicia la ejecución del código.

El compilador traduce el código en 8 instrucciones, de las cuales 2 son instrucciones de acceso a memoria. El CPI medio cuando se acierta en cache es 1.

Pregunta 1: *Calcule el número total de fallos.*

Pregunta 2: *Calcule la relación de fallos por instrucción.*

Pregunta 3: *Teniendo en cuenta los fallos en cache, calcule el CPI, MIPS y MFLOPS.*

Añadimos una cache de segundo nivel (L2) y suponemos que no falla ya que los vectores X e Y están cargados en L2 antes de iniciar la ejecución del código.

Pregunta 4: *Deduzca la penalización por fallo en L1 para doblar el rendimiento.*

Ejercicio 1.12

En un procesador se ejecuta el siguiente código.

```
do I =1, N
   s = max (s , A(I))
enddo
```

El procesador tiene una cache de datos de primer nivel de 8 KBytes, de mapeo directo y tamaño de bloque de 32 bytes y el siguiente nivel de la jerarquía es memoria. Cuando se falla en el primer nivel de cache, el número de ciclos adicionales para obtener el dato de memoria es 30.

Suponga que cuando se inicia la ejecución del bucle la variable A no está almacenada en cache, que la variable s se ha asignado a un registro antes de iniciar la ejecución de las iteraciones del bucle y se almacena en memoria después de ejecutar las N iteraciones del bucle, siendo un acierto en cache. También suponga que la variable A está alineada en memoria a bloque de 32 bytes y que el tamaño de un elemento de A y la variable s es de 64 bits.

Pregunta 1: *Calcule el número medio de ciclos de un acceso a datos (CMA), suponiendo que cuando se acierta en cache el número de ciclos es 1.*

Suponga que cuando se ejecuta el código no se producen fallos en la cache de instrucciones.

Cuando se detecta un fallo de cache el procesador suspende la interpretación de todas las instrucciones (se bloquea). Una vez se ha servido el fallo, se reanuda la interpretación de instrucciones.

El número de instrucciones del bucle es 4, solo una de ellas accede a memoria de datos y el CPI medio cuando se acierta en cache es 1.25.

Pregunta 2: *Calcule los fallos por instrucción.*

Pregunta 3: *Calcule el CPI cuando se consideran los fallos de cache.*

El computador donde se ejecuta el programa funciona a una frecuencia de 500 Mhz y consume una potencia de 30 W (vatios). La batería que alimenta al procesador suministra 1 A · H (amperios por hora) a 5 voltios.

Pregunta 4: *Calcule la energía de la batería.*
$\text{Energía}|_{\text{batería}} = \text{amperios} \times \text{segundo} \times \text{voltios}$

Pregunta 5: *Calcule la energía consumida por el procesador en un ciclo.* $\text{Energía}|_{\text{procesador}} = \text{potencia} \times \text{tiempo en segundos}$

Pregunta 6: *¿Cuántas iteraciones del bucle se pueden ejecutar antes de que la carga de la batería se reduzca a la mitad?.*

Ejercicio 1.13

El siguiente código, una vez traducido a LM, se ejecuta en un procesador.

```
do I =1, 750
   if (A(I) ≥ 0) then
      P = P + A(I)
   else
      N = N + A(I)
   endif
enddo
```

El procesador tiene una cache de datos cuyo tamaño de bloque es de 32 bytes y el siguiente nivel de la jerarquía es memoria.

El tamaño de un elemento del vector A es 8 bytes y las variables I, P, y N están almacenadas en registros mientras se ejecuta el bucle. Suponga que la variable A está alineada en memoria a bloque de 32 bytes. Esto es, la dirección del primer elemento (A(1)) del vector A es divisible por 32.

Pregunta 1: *Calcule los fallos de carga debidos al vector A (el vector A no está almacenado en cache).*

Pregunta 2: *Calcule los fallos por acceso a memoria debidos al vector A.*

Cuando se falla en el primer nivel de cache, el número de ciclos adicionales para obtener el dato de memoria es 30.

Pregunta 3: *Calcule el número medio de ciclos de un acceso a datos (CMA), suponiendo que cuando se acierta en cache el número de ciclos es 1.*

Suponga que cuando se ejecuta el código no se producen fallos en la cache de instrucciones y que cuando se detecta un fallo en la cache de datos, el procesador suspende la interpretación de todas las instrucciones (se bloquea). Una vez se ha servido el fallo, se reanuda la interpretación de instrucciones.

El bucle tiene 8 instrucciones en lenguaje máquina, sólo una de ellas accede a memoria de datos y el IPC medio cuando se acierta en cache es 0.6.

Pregunta 4: *Calcule los fallos por instrucción debidos a datos.*

Pregunta 5: *Calcule el CPI cuando se consideran los fallos de cache.*

En un nuevo diseño del procesador se reduce el tiempo de ejecución de 2 tipos de instrucciones (X e Y) y no se modifica el tiempo de ciclo del procesador. El tipo de instrucciones X representa un 10% del tiempo de ejecución y los ciclos de ejecución de este tipo de instrucciones se reducen en un 60%. El tipo de instrucciones Y representa un 50% del tiempo de ejecución y los ciclos de ejecución de este tipo de instrucciones se reducen en un 70%.

Pregunta 6: *Calcule la ganancia que se obtiene al ejecutar el programa en el nuevo diseño, suponiendo que el vector A está almacenado en cache antes de empezar la ejecución del bucle.*

Ejercicio 1.14

Un procesador con una frecuencia de reloj f MHz ejecuta una carga de trabajo de n programas. Sea N el número total de instrucciones ejecutadas y CIC el número total de ciclos de ejecución del conjunto de programas. De cada programa se conoce:

	descripción
p_i	fracción de tiempo (CIC_i / CIC), donde CIC_i es el número de ciclos de ejecución
CPI_i	CPI medio
$MIPS_i$	media de MIPS
W_i	potencia media (vatios) consumida por el procesador

Pregunta 1: *Deduzca una expresión para evaluar el CPI medio de la carga de trabajo en función del CPI de cada programa.*

Pregunta 2: *Deduzca una expresión para evaluar el rendimiento en MIPS de la carga de trabajo en función de los MIPS de cada programa.*

Pregunta 3: *Calcule la potencia media W consumida por el procesador (vatios) al ejecutar la carga de trabajo en función de la potencia en cada programa.*

Pregunta 4: *Calcule la energía (julios) por millón de instrucciones (EPMI) en función del rendimiento en MIPS.*

Ejercicio 1.15

Una especificación de lenguaje máquina se ha utilizado para construir 2 procesadores (A y B) utilizando dos generaciones tecnológicas. En un cambio de generación tecnológica se escalan las siguientes magnitudes.

	se reduce
Tamaño del lado del transistor (es cuadrado)	30%
Retardo de puerta	30%
Capacidad efectiva equivalente	30%
Tensión de alimentación	30%

Pregunta 1: *Calcule la ganancia potencial en rendimiento (MIPS) que aporta el cambio de generación tecnológica al construir el procesador B.*

Además, en el cambio de generación tecnológica, se ha reducido el número de niveles de puertas empleados en cada ciclo de reloj y se han utilizado técnicas de diseño de circuito que permiten incrementar la frecuencia de funcionamiento del procesador en 1.4X. Ahora bien, las mejoras descritas requieren que se utilice un 10% más de área en el procesador B que el que determina la teoría del escalado.

Pregunta 2: *Calcule la ganancia en rendimiento teniendo en cuenta el incremento descrito de frecuencia.*

Pregunta 3: *Calcule la relación (cociente) de potencia del procesador B respecto del procesador A.*

Debido a que en un cambio tecnológico el incremento de rendimiento de la memoria es menor, la penalización cuando se accede a memoria pasa de 60 ciclos de procesador en el procesador A a 80 ciclos de procesador en el procesador B. Esto es, aumenta la penalización por fallo en el 1º nivel de cache (L1). Por ello, en el procesador B se decide incluir el 2º nivel (L2) de la jerarquía de memoria en el chip. Ello permite que la cache L2 funcione a la frecuencia del procesador y que se pueda incrementar el ancho de banda entre L1 y L2.

El CPI de los procesadores A y B, cuando siempre se acierta en L1, es 1.05 y 1.10 respectivamente. La frecuencia de fallo de cache por instrucción en L1 es 0.15 y la frecuencia de fallo por instrucción en L2 es 0.02. Las penalizaciones por fallo en L1 y acierto en L2 son 10 ciclos en el procesador A y 5 ciclos en el procesador B. Los ciclos de penalización son relativos a cada procesador.

Pregunta 4: *Calcule el CPI de los procesadores A y B.*

Pregunta 5: *Calcule la ganancia en CPI del procesador B.*

Pregunta 6: *Calcule la ganancia en rendimiento del procesador B teniendo en cuenta tanto los aspectos tecnológicos como de arquitectura. Exprese la ganancia en tanto por ciento.*

Incluir la cache L2 en el chip del procesador B requiere incrementar el área del dado. El incremento de área, relativo al área después de incluir las mejoras para incrementar la frecuencia, es un 60%.

Considere que la densidad de potencia (Potencia/Area) de la cache L2 es 10 veces menor que la del núcleo del procesador con sus cache (L1) de instrucciones y datos.

Pregunta 7: *Calcule la relación de potencia del procesador B respecto del procesador A.*

Ejercicio 1.16

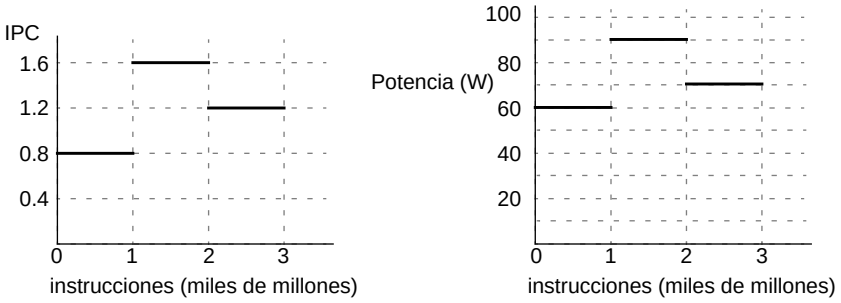
Un procesador puede funcionar con una tensión de alimentación (V) de 1.48 voltios y una frecuencia de 1.6 Ghz (modo A). Este mismo procesador, para reducir el consumo, puede también funcionar con una tensión de alimentación un 35% menor y a la vez la frecuencia se reduce un 62% (modo B).

Pregunta 1: *Al ejecutar un programa, calcule el incremento porcentual de tiempo de ejecución cuando el procesador funciona en el modo B en lugar del modo A.*

Pregunta 2: *Al ejecutar un programa, calcule la reducción porcentual de energía cuando el procesador funciona en el modo B en lugar del modo A.*

Ejercicio 1.17

En las siguientes figuras se muestra el IPC y la potencia consumida a medida que se interpretan instrucciones de un programa, en un procesador que funciona a una frecuencia de 2 Ghz.



Pregunta 1: *Calcule el tiempo de ejecución del programa.*

Pregunta 2: *Calcule la potencia de conmutación media. Recuerde que la potencia es la energía por unidad de tiempo.*

Suponga que por cuestiones de disipación térmica la potencia máxima consumida no puede exceder de los 80 W. Para controlar la potencia máxima se utiliza escalado tensión-frecuencia. Se reduce la tensión de alimentación y como la frecuencia, en un cierto rango de valores, es linealmente proporcional a la tensión de alimentación también se reduce la frecuencia. Finalmente, como la potencia es proporcional a la frecuencia y al cuadrado de la tensión de alimentación, se reduce la potencia.

Para determinar la relación potencia-rendimiento utilice la regla 3:1 que indica que por cada 3% de reducción (incremento) en la potencia consumida se reduce (incrementa) la frecuencia de reloj en un 1%. Esta regla es una aproximación de primer orden de la relación.

$$\left(\frac{f_1}{f_2}\right)^3 = \left(\frac{V_1}{V_2}\right)^3 = \frac{P_1}{P_2}$$

Suponiendo incrementos de frecuencia Δf y de potencia ΔP y efectuando manipulaciones algebraicas y aproximaciones tenemos

$$\left(\frac{f+\Delta f}{f}\right)^3 = \frac{P+\Delta P}{P} \longrightarrow 1+3\frac{\Delta f}{f} \sim 1+\frac{\Delta P}{P} \longrightarrow 3\frac{\Delta f}{f} \sim \frac{\Delta P}{P}$$

Pregunta 3: *Calcule el tiempo de ejecución cuando se utiliza escalado tensión-frecuencia para mantener la potencia consumida dentro del límite de 80 W en cada región.*

Pregunta 4: *Calcule la potencia de conmutación media en el escenario descrito.*

Suponga que la potencia consumida debido a corrientes de fugas es del orden del 20% en cada región de la figura mostrada previamente. La potencia debido a las corrientes de fugas se suma a la potencia de conmutación para obtener la potencia total consumida en cada región.

Suponga que dinámicamente se utiliza escalado tensión-frecuencia para mantener la potencia total dentro del límite de 80 W.

Pregunta 5: *Calcule el tiempo de ejecución.*

Pregunta 6: *Calcule la potencia media en el escenario descrito.*

Ejercicio 1.18

Un procesador convencional tiene un primer nivel de cache de datos de 32 Kbytes 2-asociativa, siendo el tamaño de bloque de 16 bytes. La cache de datos es bloqueante y la latencia de acceso a memoria son 4 ciclos. Por otro lado, supondremos que la cache de instrucciones es ideal; es decir, no se producen fallos de cache.

En este procesador se efectúa el cálculo especificado en el siguiente código.

Lenguaje de alto nivel	Lenguaje ensamblador
do I = 1, N	1$: 1. load **R1**, 0(R2)
s = s + A(I)	2. add **R3**, R4, R1
enddo	3. add **R2**, R2, # 8
	4. add **R5**, R5, # -1
	5. bne R5, 1$

Un elemento del vector A ocupa 8 bytes y el elemento A(1) está alineado a principio de bloque.

El vector A no está almacenado en cache cuando se inicia la ejecución del bucle.

Suponga que el tiempo de interpretación de cualquier instrucción, cuando se acierta en cache, es 1 ciclo.

La respuesta de algunas preguntas requiere la utilización de diagramas temporales. En estos diagramas se representan en vertical las instrucciones y en horizontal los ciclos. Así mismo, todas las ejecuciones de una misma instrucción (instancia) se representan en la misma fila del diagrama, en el ciclo que le corresponda.

Pregunta 1: *Calcule en el procesador convencional el número de ciclos que tarda en ejecutarse el bucle y el CPI.*

El consumo medio de potencia cuando se interpreta una instrucción, suponiendo acierto en cache, son P vatios. Durante el tiempo transcurrido en un fallo de cache el consumo se reduce un 50%, ya que el procesador está esperando el dato y no se interpretan más instrucciones (load bloqueante). El consumo de energía durante un fallo de cache lo imputaremos al núcleo del procesador. Estos es, supondremos que el acceso a memoria no consume energía.

Pregunta 2: *Calcule la energía consumida suponiendo que el procesador convencional funciona a 500Mhz.*

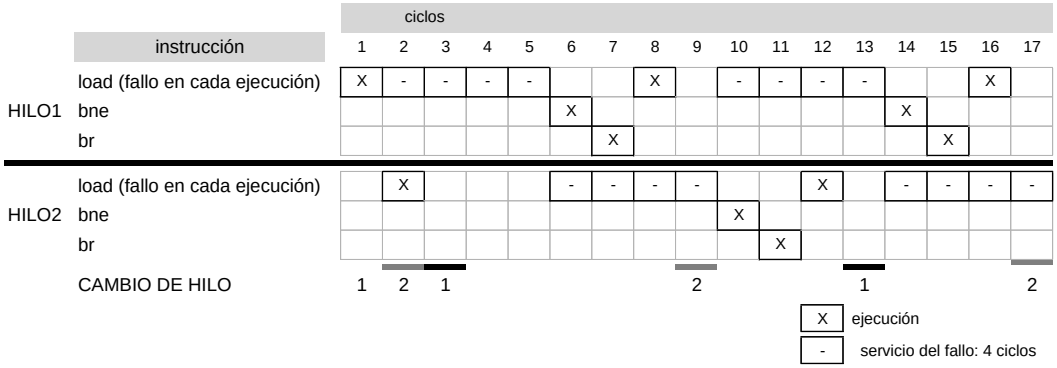
En otra versión del procesador se introduce la técnica multihilo con el objetivo de aumentar la productividad y la utilización de los recursos del procesador.

La técnica multihilo permite que se interpreten concurrentemente varias secuencias de código. La granularidad con que se aplica la técnica es gruesa. Esto es, el procesador interpreta instrucciones del mismo hilo mientras no se produzca un evento singular, que en nuestro caso será un fallo de cache.

Cuando se produce un fallo de cache, en el siguiente ciclo se inicia (o reanuda) la interpretación de instrucciones de otro hilo hasta que se produce un fallo de cache.

En el procesador multihilo se mejora la funcionalidad de la cache de datos haciendo que sea no bloqueante. En este caso se permite que mientras se sirve un fallo de cache se puedan estar sirviendo aciertos de cache.

Si mientras se está sirviendo un fallo de cache se produce otro fallo de cache, el servicio de este último se retarda hasta que ha finalizado el fallo previo de cache. Ahora bien, el procesador cambia de hilo, y si éste esta esperando el servicio de un fallo de cache el procesador se bloquea. En la siguiente figura se muestra un ejemplo.



Nosotros supondremos que el procesador puede estar interpretando concurrentemente las instrucciones correspondientes a 2 hilos. El hilo1 ejecuta el código especificado previamente y el hilo2 ejecuta el mismo código, pero el vector al que se accede es el vector B. El número de iteraciones en el hilo2 también es el mismo que en el hilo1.

Suponga que tanto el vector A como el vector B no están almacenados en la cache de datos al iniciar la ejecución de los códigos. Suponga también que en primer lugar se inicia la ejecución del hilo1.

En todas la preguntas que siguen indique los tiempos de ejecución en función de N. No es necesario considerar los ciclos en el inicio o el final.

Pregunta 3: *Calcule el tiempo de ejecución del hilo1 en el procesador multihilo. Cuantifique el incremento del tiempo de ejecución respecto del procesador convencional.*

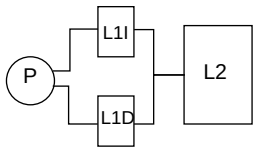
Pregunta 4: *Calcule el número de ciclos que tardan en ejecutarse los dos bucles en el procesador multihilo.*

Pregunta 5: *Calcule la ganancia en el tiempo de ejecución al ejecutar los dos bucles en el procesador multihilo respecto del procesador convencional.*

Pregunta 6: *Calcule la energía consumida cuando se ejecutan los dos bucles en el procesador multihilo. Cuantifique la disminución de energía consumida respecto del procesador convencional. El procesador multihilo funciona a 500Mhz.*

Ejercicio 1.19

Un procesador tiene una cache de instrucciones y una cache de datos como primer nivel de la jerarquía de memoria. Las dos cache tienen un tamaño de 8Kbytes y el mapeo es directo. El tamaño de bloque utilizado en las dos caches es de 32 bytes.



Supondremos que siempre se acierta en el segundo nivel de la jerarquía de memoria. La penalización en un fallo de cache, en el primer nivel de la jerarquía, son 10 ciclos.

En un fallo de cache, si el contenedor donde debe almacenarse el bloque está ocupado por otro bloque se efectúa una acción de reemplazo. El bloque que ocupa el contenedor se reemplaza por el bloque referenciado en el fallo. En este ejercicio no es necesario considerar actualizaciones de L2 en una acción de reemplazo. Además, suponemos que todos los bloques accedidos están en L2.

En este procesador, una vez traducido a lenguaje máquina, se ejecuta el siguiente código.

```
do I = 1, 2000
   s = s + A(I)
enddo
do I = 1, 2000
   p = p * A(I)
enddo
```

El vector A no está almacenado en la cache L1D antes de iniciar la ejecución del código, el tamaño de un elemento es de 8 bytes y la dirección del primer elemento (A(1)) del vector A es divisible por 32. Las variables s y p se han asignado a registros antes de iniciar las iteraciones de los dos bucles.

Pregunta 1: *¿Cuántos elementos del vector A pueden almacenarse concurrentemente en la cache de datos de primer nivel?.*

Pregunta 2: *Calcule el número de fallos que se producen en la cache de datos cuando se ejecutan los dos bucles.*

Cuando se acierta en el primer nivel de la jerarquía de memoria, el CPI al ejecutar el 1º bucle es 1.0 y el CPI al ejecutar el 2º bucle es 1.4. El número de instrucciones de LM al traducir los dos bucles es el mismo e igual a 5. El número de instrucciones de acceso a memoria en cada bucle es 1 y representa un 20% de las instrucciones en que se ha traducido cada bucle.

Pregunta 3: *Considerando exclusivamente fallos en la cache de datos, calcule el CPI al ejecutar los dos bucles.*

Una optimización, denominada fusión de bucles, efectuada por el compilador, con el objetivo de reducir el tiempo de ejecución, transforma el código anterior en el siguiente código.

```
do I = 1, 2000
   s = s + A(I)
   p = p * A(I)
enddo
```

Pregunta 4: *Calcule el número de fallos en la cache de datos que se producen al ejecutar el bucle.*

Cuando se acierta en el primer nivel de la jerarquía de memoria, el CPI al ejecutar el bucle es 7/6. El número de instrucciones de acceso a memoria representa 1/6 de las instrucciones en que se ha traducido el bucle. El número de instrucciones en LM es igual a 6.

Pregunta 5: *Considerando exclusivamente fallos en la cache de datos, calcule el CPI al ejecutar el bucle.*

Pregunta 6: *Calcule la ganancia que se obtiene al efectuar la transformación de código.*

Suponga que también se producen fallos en la cache de instrucciones y que el código no está almacenado en cache antes de iniciar la ejecución de los bucles. Una instrucción ocupa 32 bits.

Pregunta 7: *Calcule los fallos debidos a instrucciones en cada uno de los códigos.*

La potencia cuando se ejecutan instrucciones es 30 vatios y cuando el procesador se bloquea debido a un fallo de cache la potencia se reduce un 80%. El procesador funciona a una frecuencia de 500Mhz.

Pregunta 8: *Calcule la energía media consumida cuando se ejecuta cada uno de los códigos.*

Pregunta 9: *Calcule la ganancia en energía debido a la transformación de código.*

Ejercicio 1.20

Un procesador PR1, que tiene frecuencia 2 GHz y 60 w. de potencia, interpreta instrucciones de tres tipos, que son ENT, MEM y BR.

Todas estas instrucciones tienen CPI = 2, excepto las que son de tipo MEM y fallan el acceso a cache, que entonces penalizan cada una 10 ciclos adicionales.

PR1 tarda 0.05 s en ejecutar un programa P, con 40×10^6 instrucciones distribuidas en 25% del tipo BR, 30% del tipo ENT y el resto del tipo MEM.

Pregunta 1: *Calcule la tasa de fallos en cache. Exprese el resultado de dos maneras: como porcentaje sobre el total de instrucciones y como porcentaje sobre instrucciones de tipo MEM.*

Pregunta 2: *Calcule la energía, en julios, que queda en la batería que alimenta al PR1, sabiendo que es un tercio de la energía inicial y que PR1 ha ejecutado, sin interrupciones, el programa P un millón de veces consecutivas.*

Una nueva versión del procesador, PR2, tiene el doble de frecuencia y la mitad de la tensión de alimentación que el PR. El juego de instrucciones y la capacidad efectiva equivalente son iguales que en PR1.

Cuando ejecuta el programa P, el PR2 consigue un valor medio de IPC = 0.6.

Pregunta 3: *Calcule la eficiencia de la nueva versión, expresada como el cociente adimensional FM(PR2) / FM(PR1) siendo FM la figura de mérito que se expresa en MIPS / vatio.*

Ejercicio 1.21

Una aplicación que utiliza un algoritmo de filtrado de imágenes tarda 2 segundos en efectuar el filtrado de una imagen cuando se ejecuta en un procesador A que funciona a 2 Ghz.

En el procesador A se ha ampliado el repertorio de instrucciones multimedia (SIMD). Ello permite que 4 sumas independientes de 16 bit se empaqueten en una única instrucción SIMD. A este procesador lo denominaremos B. Sin embargo, debido a problemas en la implementación de la ampliación del repertorio de instrucciones multimedia, la frecuencia de funcionamiento debe reducirse a 1.94 Ghz.

Cuando una aplicación, que utiliza instrucciones multimedia, se recompila para ejecutarla en el procesador B se reduce el número de instrucciones que se ejecutan. Al recompilar la aplicación descrita previamente, se mide que un 24% de las instrucciones se pueden reemplazar por instrucciones SIMD pertenecientes a la ampliación.

Suponga que el CPI medio es el mismo en los dos procesadores.

Pregunta 1: *Calcule el número de instrucciones en el procesador B.*

Pregunta 2: *Calcule el CPI en el procesador B.*

Pregunta 3: *Calcule el tiempo de ejecución en el procesador B.*

Pregunta 4: *Los 2 procesadores consumen 80 vatios de potencia. Calcule la reducción porcentual de energía consumida en el procesador B.*