Capítulo 1

PROCESADORES SEGMENTADOS

1.1. Guión-esquema

Los contenidos que se tratan a lo largo del tema se resumen en los siguientes puntos:

- Diferencias entre procesadores RISC y procesadores CISC.
- Clasificación y características generales de las arquitecturas paralelas.
- Medidas para evaluar el rendimiento de un computador.
- Características de los procesadores segmentados.
- Descripción de una arquitectura segmentada genérica.
- Tipos de riesgos presentes en la segmentación.
- Técnicas que evitan los riesgos en la segmentación.
- El algoritmo de Tomasulo como técnica de planificación dinámica en la segmentación.

1.2. Introducción

El capítulo se dedica al estudio de los procesadores segmentados, cuya principal característica es que utilizan la segmentación como técnica de diseño hardware para incrementar su rendimiento. Este capítulo se centra, principalmente, en analizar las cuestiones que se plantean a la hora de aplicar la segmentación para mejorar el rendimiento de un procesador. Un procesador segmentado es capaz de procesar varias instrucciones simultáneamente, aunque cada una de ellas puede encontrarse en una etapa distinta de su procesamiento. Por esta razón, el procesador es capaz de aprovechar el paralelismo intrínseco existente entre las instrucciones y aumentar su rendimiento, ejecutando más instrucciones por unidad de tiempo sin tener que reducir necesariamente el tiempo de ejecución de cada instrucción.

El origen del uso de la segmentación en los computadores se puede situar en el IBM 709, un computador basado en tubos de vacío con memoria de núcleos de ferrita que apareció en 1958 como una mejora del IBM 704. Entre las nuevas técnicas que incorporaba, una de las más relevantes consistía en la incorporación de unidades de sincronización de datos que permitían el procesamiento solapado de las instrucciones y de las entradas/salidas. Con la misma frecuencia de reloj que el IBM 704, el IBM 709 conseguía un 60 % más de prestaciones. En 1962 aparece el IBM 7094I, un computador construido con transistores y memoria de núcleos de ferrita que incluía un registro, el IBR (*Instruction Backup Register*), para almacenar la siguiente instrucción a ejecutar, permitiendo el solapamiento entre la ejecución de una instrucción y la captación de la siguiente. La frecuencia de reloj del IBM 7094I era 6 veces mayor que la del IBM 709 pero la velocidad era 7,5 veces mayor.

El término segmentación aparece por primera vez en 1956 en en el proyecto Stretch, cuyo objetivo era conseguir una máquina 100 veces más rápida que el IBM 704. Precisamente, el IBM 7030 Stretch se puede considerar el primer procesador de propósito general segmentado. Su microarquitectura estaba implementada a través de una segmentación o cauce con cuatro etapas en la que se solapaban la lectura de instrucciones, su decodificación y su ejecución. En 1961 se demostró que solo habían conseguido ser 30 veces más rápidos que el IBM 704 debido a que el IBM 7030 tardaba demasiado tiempo en recuperarse de un error, lo cual originó una fuerte caída del precio de la máquina.

Aunque la segmentación se utilizó frecuentemente en los computadores diseñados desde finales de los años 50, en la década de los 70 pasó a tener un papel secundario. Esto fue debido a que se hizo más énfasis en las máquinas orientadas a simplificar el software (la máquina más representativa en este periodo fue el VAX[1]). A pesar de ello, en los años 70 aparecen una serie de trabajos dedicados a la aplicación de la segmentación que ya introducen la terminología asociada y muchas de las técnicas que se utilizarán posteriormente. Como consecuencia de la abundante investigación y desarrollo que se realiza en la década de los 70, en 1981 se publica el libro de Kogge "Architecture of Pipelined Computers" considerado como el primer texto dedicado completamente y de una manera formal a la segmentación. Durante la década de los ochenta, la segmentación se convirtió en la piedra angular del enfoque RISC para el diseño de procesadores. Los procesadores RISC se diseñaron desde el principio

[1]Máquina CISC sucesora de la PDP-11, producida por Digital Equipment Corporation. Fue la primera máquina comercial de arquitectura de 32 bits, lo que la convierte en un hito destacable en la historia de la computación.

como procesadores segmentados y, precisamente, uno de los argumentos más empleados para justificar las arquitecturas RISC frente a las CISC [2] era que, puesto que las instrucciones RISC son más simples, su procesamiento segmentado se puede realizar en cauces con menos etapas, facilitando el trabajo del compilador a la hora de generar código que optimice su aprovechamiento. No obstante, es difícil encontrar el número de etapas óptimo del cauce, debido a que éste dependerá de la arquitectura y de las características de los programas que se ejecuten. Así, han aparecido microprocesadores segmentados con un número de etapas muy diverso, desde las dos etapas del RISC I (1982), las 10 etapas del Pentium III, las 20 ó 31 etapas del Pentium 4 (2000-2005) según modelo, hasta las 14 etapas de los procesadores basados en la microarquitectura Intel Core (2006-2010). Desde mediados de los 80, los microprocesadores ya incorporan la segmentación en sus microarquitecturas de forma generalizada.

Para explicar el funcionamiento de los procesadores segmentados, en este capítulo se analizará una arquitectura segmentada genérica junto con ejemplos prácticos que facilitarán la comprensión de los conceptos teóricos. Además, se verá que el aprovechamiento de la segmentación en la ejecución de instrucciones plantea una serie de problemas que deben resolverse para que sea posible obtener un rendimiento aceptable. Esos problemas se refieren a los riesgos que aparecen entre las instrucciones, pudiendo ser riesgos por dependencias de datos, riesgos de control y riesgos estructurales. Finalmente, se estudiarán diversas técnicas basadas tanto en el hardware como en el software que permitirán gestionar los problemas de la segmentación.

Los objetivos que se pretenden alcanzar con el estudio de este capítulo son:

- Aprender las diferencias entre los procesadores RISC y los CISC.
- Clasificar y conocer las características generales de las arquitecturas paralelas.
- Conocer y saber utilizar las medidas para evaluar el rendimiento de un computador.
- Conocer las características de los procesadores segmentados.
- Conocer y saber analizar los distintos tipos de riesgos presentes en la segmentación.
- Aprender las técnicas que evitan los riesgos en la segmentación.
- Entender el funcionamiento del algoritmo de Tomasulo como técnica de planificación dinámica en la segmentación.

1.3. Procesadores RISC frente a procesadores CISC

En la década de los años 70 la técnica más apropiada para el desarrollo de procesadores era la microprogramación, que consiste en un conjunto de microinstrucciones que representan las señales eléctricas internas de la unidad de control de un microprocesador. Debido al alto coste por byte en esos

[2] Los términos RISC y CISC se explican en la siguiente sección.

años, los computadores no disponían, en general, de grandes cantidades de memoria. Por esa razón, los diseñadores comenzaron a condensar la mayor cantidad posible de operaciones dentro del microcódigo reduciendo el tamaño de los programas tanto como fuese posible. En consecuencia, los procesadores fueron dotados de conjuntos de instrucciones muy potentes que realizaban gran cantidad de operaciones internas. Estas instrucciones eran decodificadas internamente en la unidad de control y ejecutadas mediante una serie de microinstrucciones almacenadas en una ROM interna, conocida como *memoria de control*. Para este proceso se requerían varios ciclos de reloj, siendo de al menos uno por microinstrucción a ejecutar. Debido a la gran cantidad de operaciones internas que realizaban las instrucciones de estos procesadores se los pasó a denominar CISC (*Complex Instruction Set Computer* - Computadores con Conjunto de Instrucciones Complejo). Simultáneamente y con el afán de disminuir la cantidad de instrucciones que un programa debía almacenar en memoria, se introdujeron en la codificación de las instrucciones una gran cantidad de modos de direccionamiento. Esto fue debido a que en los años 60 y 70, el tiempo de acceso a memoria era mucho menor que el tiempo de procesamiento, por lo cual realizar sucesivos accesos a memoria para buscar operandos era la mejor alternativa con que se contaba para disminuir el tamaño del código. De esta forma surgieron las arquitecturas memoria-memoria y registro-memoria.

La arquitectura tipo CISC dificultaba, en general, el paralelismo a nivel de instrucciones debido a que para realizar la ejecución de una instrucción, esta debía ser cargada desde la memoria y traducida a microoperaciones antes de ejecutar la siguiente instrucción. Por lo tanto, la gran cantidad de dependencias entre las microoperaciones pertenecientes a una misma instrucción dificultaba el paralelismo.

Buscando aumentar la velocidad del procesamiento, se descubrió en base a distintos estudios, principalmente los de Patterson en el desarrollo del procesador RISC-I y los de Hennessy en el desarrollo del MIPS, que con una determinada arquitectura de base, la ejecución de programas compilados con instrucciones simples (de complejidades similares a las microinstrucciones de los procesadores CISC) resultaban ser más eficientes. Como resultado de estos estudios, se observó que el 80 % del tiempo de procesamiento era consumido por solo el 20 % de las instrucciones, con una mayor preponderancia de los almacenamientos (*stores*), cargas (*loads*) y bifurcaciones (*branches*). Esto permitió visualizar que se poseían memorias de control muy grandes cuyo contenido era muy poco utilizado.

Por otra parte, el avance en la tecnología de fabricación de los circuitos integrados provocó que el aumento de velocidad de los procesadores pasase a ser mucho mayor que el de las memorias, con lo que el objetivo de diseño de las arquitecturas RISC (*Reduced Instruction Set Computer* - Computadores con Conjunto de Instrucciones Reducido) pasó a ser el de disminuir la cantidad de accesos a memoria tanto como fuese posible mediante un aumento de la cantidad de registros internos en el procesador.

Debido a que un procesador RISC tiene un conjunto de instrucciones simplificado, ya no es necesario disponer de un hardware de control extremadamente complejo. La decodificación de las instrucciones puede ser directamente implementada por hardware dentro de la CPU, disminuyendo la lógica de control y obteniéndose una mayor velocidad de ejecución. La alternativa RISC se basa en esto y propone un conjunto de instrucciones sencillas con formatos fijos, permitiendo la posibilidad de introducir la

ejecución segmentada de las instrucciones (*pipeline*) y el diseño de procesadores cableados con un repertorio reducido de instrucciones sencillas pero frecuentemente utilizadas. Esto posibilita que todo código complejo puede descomponerse en instrucciones elementales en las que, para evitar los retardos de la memoria principal, se recurre a numerosos registros internos para albergar la mayor cantidad posible de variables. En resumen, las principales ventajas de un diseño RISC, además de aumentar la velocidad de ejecución, son:

- Reduce el tamaño de la CPU, por lo que se dispone de más espacio para más recursos, por ejemplo, mayor cantidad de registros o de memoria caché.
- Al ser la lógica de control más simple, se facilita el diseño.
- Permite máquinas más compactas y con menor consumo al reducirse el tamaño de la CPU.
- Posibilita la segmentación y el paralelismo en la ejecución de instrucciones.
- Reduce los accesos a memoria debido a que los operandos se cargan en registros.

El rendimiento de la implementación *pipeline* de un procesador RISC obtenido al simplificar el repertorio de instrucciones depende directamente de los riesgos que se generan durante la ejecución de las instrucciones, considerando que un riesgo aparece cuando dos o más instrucciones entran en conflicto. Para minimizar los riesgos, la arquitectura RISC plantea en su filosofía de diseño una relación muy estrecha entre los compiladores y la propia arquitectura.

Por esta razón, el compilador pasa a jugar un papel clave en el diseño de un sistema RISC equilibrado. Todas las operaciones complejas que aparecen en el código fuente se descomponen en multitud de operaciones sencillas RISC. De esta manera, el compilador asume la función de un mediador entre el programa de aplicación y el microprocesador. Es decir, se hace un gran esfuerzo para mantener al hardware tan simple como sea posible, aun a costa de hacer al compilador considerablemente más complejo. Esta estrategia se encuentra en clara contraposición con las máquinas CISC que tienen instrucciones y modos de direccionamiento muy complejos intentando asemejar el lenguaje de máquina a los lenguajes de alto nivel.

En un procesador RISC se obtiene una ganancia en velocidad por el hecho de que dominan las instrucciones más frecuentemente utilizadas, mientras que las operaciones menos frecuentes deben descomponerse en operaciones simples. Por esta razón, la cantidad de instrucciones que requiere un RISC para ejecutar un programa resulta mayor que en el caso de un procesador CISC de características similares.

Aun así, con todos los datos obtenidos a favor del RISC y en contra del CISC, hay que tener en cuenta otro factor importante en el momento de tomar la decisión de migrar a RISC: la compatibilidad con el software preexistente. CISC y RISC no solo se refieren a la cantidad de instrucciones que poseen ambos tipos de arquitecturas, sino que también reflejan dos filosofías de diseño opuestas. Los atributos *complejo* y *reducido* describen las diferencias entre los dos modelos de arquitectura para microprocesadores solo de forma superficial. Sin embargo, se requieren muchas otras características esenciales para definir de

forma detallada los RISC y los CISC. Así, los términos *complejo* y *reducido* expresan muy bien una característica importante siempre que no se tomen solo como referencia los conjuntos de instrucciones, sino que se considere también la complejidad del hardware de los procesadores.

Con tecnologías de semiconductores comparables e igual frecuencia de reloj, un procesador RISC típico tiene una capacidad de procesamiento mayor que la de un CISC con una estructura de hardware mucho más simple, con lo cual se puede implementar en una fracción de la superficie ocupada por el circuito integrado de un procesador CISC, teniendo los beneficios de disminuir el consumo de potencia y reducir los costes. A su vez, el aumento de rendimiento de una arquitectura RISC se sustenta en el diseño de un compilador eficiente.

1.4. Clasificación de las arquitecturas paralelas

La contribución de la arquitectura al aumento de las prestaciones de los computadores ha venido dada conjuntamente por el *paralelismo* y por la utilización de los principios de *localidad espacial* y *temporal* de los datos almacenados en memoria. Ambas alternativas compiten por el uso de los recursos disponibles e interactúan de formas muy diversas en todos y cada uno de los niveles de un computador. Un ejemplo de esta situación se tiene en la utilización de las memorias cachés en los multiprocesadores[3]. Gracias a ellas se reducen los accesos a los niveles más lentos de la jerarquía de memoria y se evitan colisiones en los elementos compartidos (redes y buses), pero a su vez ocasionan problemas de coherencia a los que hay que dedicar recursos. Otro ejemplo de compromiso entre localidad y paralelismo se tiene en la decisión de utilizar más transistores en un microprocesador para aumentar el tamaño de la caché interna, o para aumentar el número de las unidades funcionales[4] y así permitir un mayor paralelismo entre las instrucciones.

El paralelismo se implementa en las arquitecturas siguiendo fundamentalmente dos líneas. Por una parte está la *replicación de elementos*, incluyendo unidades funcionales, procesadores, módulos de memoria, etc., entre los que se distribuye el trabajo. Como ejemplos de esta alternativa a nivel de sistema están los multiprocesadores y los canales/procesadores de E/S (para permitir la concurrencia entre operaciones de E/S), y a nivel de procesador se tiene el uso de varias unidades funcionales en los procesadores superescalares, los procesadores VLIW (*Very Long Instruction Word* - Palabra de Instrucción Muy Larga) y los procesadores vectoriales.

La segunda alternativa para implementar el paralelismo es la *segmentación de cauce* o simplemente *segmentación* (en inglés *pipelining*), técnica a través de la cual un elemento (unidad funcional, procesador, circuito, etc.) se divide en una serie de etapas que funcionan de forma independiente, y por las que van pasando los operandos y las instrucciones que procesa cada elemento. De esta

[3]Computador que cuenta con dos o más microprocesadores (CPUs).

[4]También denominadas unidades de ejecución. Son una parte de la CPU que realiza las operaciones y cálculos de las instrucciones de los programas. Tienen a menudo su propia unidad de control de secuencia (no confundir con la unidad de control principal de la CPU), algunos registros, otras unidades internas como una sub-ALU o una unidad de coma flotante, y algunos componentes menores más específicos.

forma, dicho elemento puede realizar simultáneamente operaciones distintas en las diferentes etapas en que se encuentra dividido. Esta técnica se ha aplicado en el diseño de procesadores, permitiendo un mayor aprovechamiento del paralelismo entre instrucciones en los procesadores superescalares y VLIW, y en el diseño de las unidades funcionales incluidas en los procesadores (sumadores segmentados, multiplicadores segmentados) que, por ejemplo, en los procesadores vectoriales donde lo que prima es aprovechar el paralelismo de datos al ejecutar una operación sobre conjuntos de operandos diferentes.

A continuación, se presenta la clasificación de las arquitecturas de computadores ideada por Michael Flynn en los años 60. Como toda clasificación, pretende dividir el universo de elementos sobre el que se aplica, en este caso computadores, en una serie de clases caracterizadas por un conjunto de propiedades. De esta forma, si se sabe a qué clase pertenece un computador, automáticamente se conocen toda una serie de características interesantes del mismo. Esta taxonomía divide el universo de computadores en cuatro clases, según el número de secuencias o flujos de instrucciones y secuencias o flujos de datos que pueden procesarse simultáneamente en el computador. La taxonomía de Flynn está relacionada, sobre todo, con el aprovechamiento del paralelismo a partir de la réplica de elementos por lo que existen otros aspectos del paralelismo que no se reflejan del todo bien. Sin embargo, constituye un buen punto de partida porque se trata de una clasificación suficientemente simple que da una idea bastante buena de las distintas arquitecturas paralelas e introduce una nomenclatura que es ampliamente utilizada. Los cuatro computadores que se distinguen en esta clasificación son los siguientes:

- Computadores SISD: Un único flujo de instrucciones (SI, *Single Instruction*) procesa operandos y genera resultados, definiendo un único flujo de datos (SD, *Single Data*).
- Computadores SIMD: Un único flujo de instrucciones (SI) procesa operandos y genera resultados, definiendo varios flujos de datos (MD, *Multiple Data*). Dado que cada instrucción codifica realmente varias operaciones iguales, cada una actúa sobre operandos distintos.
- Computadores MIMD: El computador ejecuta varias secuencias o flujos distintos de instrucciones (MI, *Multiple Instructions*), y cada uno de ellos procesa operandos y genera resultados de forma que existen también varios flujos de datos (MD), uno por cada flujo de instrucciones.
- Computadores MISD: Se ejecutan varios flujos de instrucciones (MI) aunque todos actúan sobre el mismo flujo de datos (SD).

Para ilustrar el significado de la terminología empleada al describir los distintos tipos de computadores, en la Figura 1.1 se muestran esquemas estructurales de las cuatro clases de computadores que establece la taxonomía de Flynn.

En un computador SISD (Figura 1.1.a) existe una única unidad de control (UC) que recibe las instrucciones de memoria, las decodifica y genera los códigos que definen la operación correspondiente a cada instrucción que debe realizar la unidad de procesamiento (UP) de datos. El flujo de datos se establece a partir de los operandos fuente necesarios para realizar la operación codificada en cada instrucción y de los resultados generados por las instrucciones. En un computador SIMD (Figura

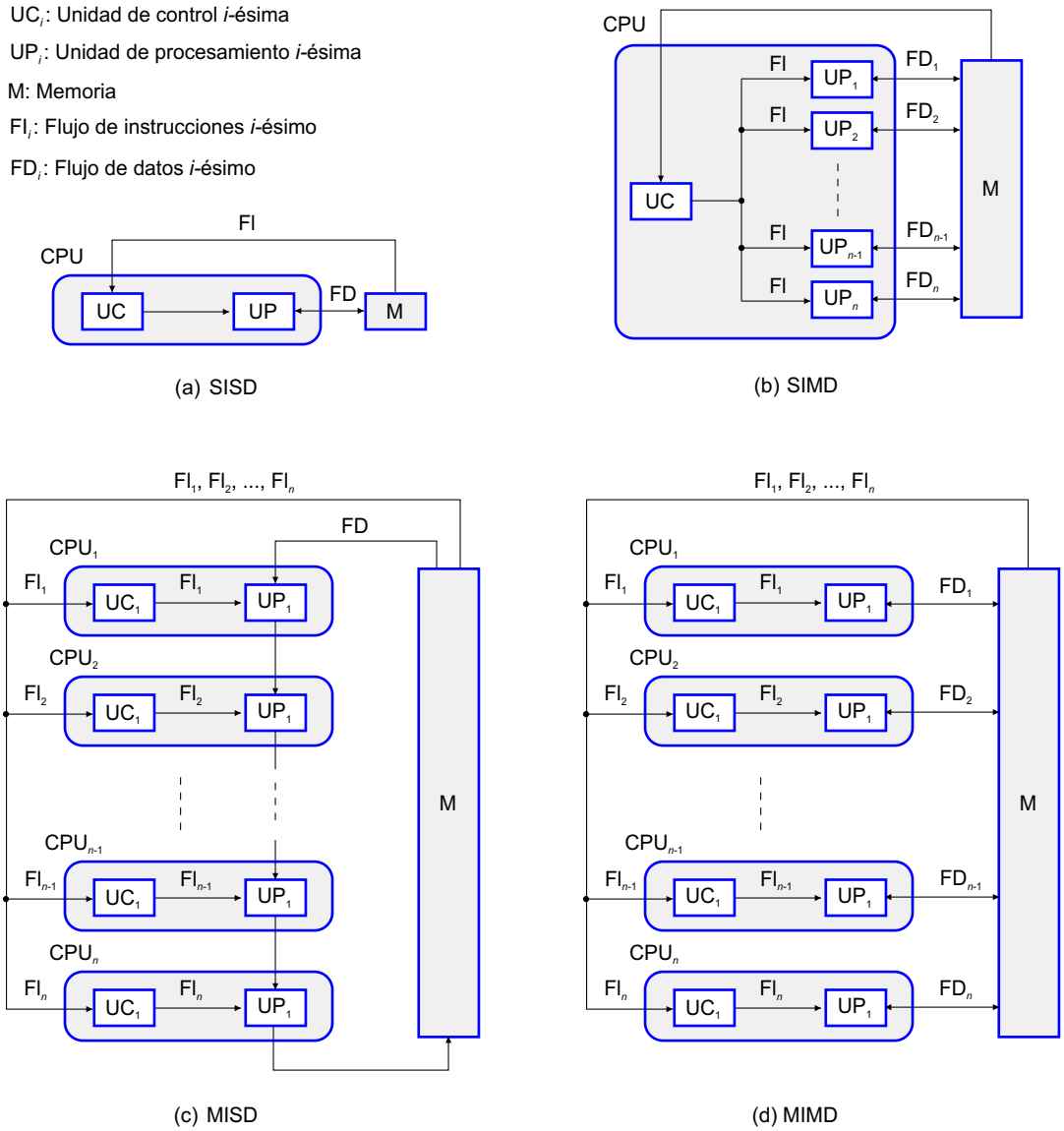


Figura 1.1: Clases de computadores que establece la taxonomía de Flynn.

1.1.b), el código de operación que genera la única unidad de control del computador a partir de cada instrucción actúa sobre varias unidades de procesamiento (UPi). De esta forma, un computador SIMD puede realizar varias operaciones similares simultáneas sobre operandos distintos. Cada una de las secuencias de operandos y resultados utilizados por las distintas unidades de proceso define un flujo de datos diferente. En un computador MIMD (Figura 1.1.d) existen varias unidades de control que decodifican las instrucciones correspondientes a distintos programas. Cada uno de esos programas procesa conjuntos de datos diferentes, que definen distintos flujos de datos. Los computadores MISD (Figura 1.1.c) constituyen una clase de computadores cuyo comportamiento se puede replicar con similares prestaciones en un computador MIMD en el que sus procesadores se sincronizan para que los datos vayan pasando desde un procesador a otro. Por esta razón, aunque se pueden identificar computadores SISD (computadores monoprocesador), computadores SIMD (procesadores matriciales y procesadores vectoriales) y computadores MIMD (multiprocesadores y multicomputadores), no existen computadores MISD específicos. Si la forma de procesamiento MISD fuera necesaria para algún problema, ésta se puede implementar en un computador MIMD en el que los datos son procesados sucesivamente por los procesadores del sistema.

La mejora de prestaciones y el paralelismo que aprovechan las arquitecturas SIMD y MIMD se pueden entender mejor estimando los tiempos de procesamiento que se tendrían en la ejecución de un fragmento de código. El siguiente ejemplo permite visualizar con claridad esta idea.

Considérese el fragmento de pseudocódigo de la Figura 1.2. El código implica realizar un total de 12 operaciones (4 sumas, 4 restas y 4 multiplicaciones), además de las operaciones correspondientes al control del bucle (en este ejemplo se consideran despreciables). Si se consideran intervalos de tiempos del mismo orden para todas las operaciones, en un SISD se necesitarían del orden de 12 intervalos (se supone que cada intervalo es el tiempo que consume una operación de suma, resta o multiplicación).

```
for (i=1; i<=4; i++)
    C[i] := A[i] + B[i];
    F[i] := D[i] - E[i];
    G[i] := K[i] * H[i];
end for;
```

Figura 1.2: Fragmento de pseudocódigo del ejemplo.

El pseudocódigo de la Figura 1.3 correspondería a un procesador matricial con cuatro unidades de procesamiento (UPi, i=1, . . . , 4). La Figura 1.4 muestra un esquema que ilustra la forma de ejecutarse el código de la Figura 1.3 y el paralelismo que se aprovecha. En este caso se necesitarían tres intervalos de tiempo. Es decir, cuatro veces menos que en el computador SISD, como sería de esperar, debido a que se están utilizando cuatro unidades de procesamiento.

En cuanto al uso de computadores MIMD para ejecutar el código de la Figura 1.2, existen distintas

```
for all UPi ( i=1; i<=4; i++)
    C[i] := A[i] + B[i];
    F[i] := D[i] - E[i];
    G[i] := K[i] * H[i];
end for;
```

Figura 1.3: Fragmento de pseudocódigo del ejemplo correspondiente a un computador SIMD con cuatro unidades de procesamiento.

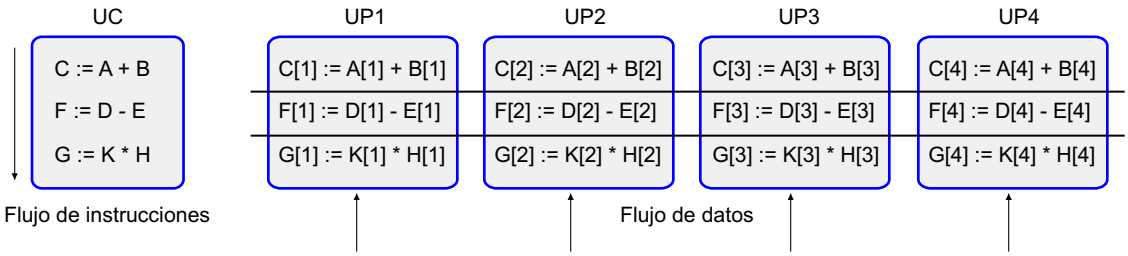


Figura 1.4: Esquema de procesamiento de tres instrucciones en un computador SIMD con cuatro unidades de procesamiento.

posibilidades de aprovechamiento del paralelismo. En la Figura 1.5 se consideran tres procesadores, y cada uno ejecuta una operación distinta sobre cuatro componentes de los vectores. En este caso, el coste temporal sería de cuatro intervalos de tiempo, es decir, tres veces menos que en la implementación secuencial.

También se puede utilizar un MIMD con cuatro procesadores, en el que cada uno de ellos realiza las tres operaciones con una de las cuatro componentes de los vectores. En este caso, el tiempo se reduciría a tres intervalos, cuatro veces menos que en el caso de un SISD. En esta situación se dice que el computador MIMD utiliza una implementación SPMD (*Simple Program Multiple Data*) del problema.

La taxonomía de Flynn pone de manifiesto dos tipos de paralelismo que pueden aprovecharse según

```
Procesador 1                Procesador 2                Procesador 3
for  (i=1; i<=4; i++)       for  (i=1; i<=4; i++)       for  (i=1; i<=4; i++)
    C[i] := A[i] + B[i];        F[i] := D[i] - E[i];        G[i] := K[i] * H[i];
end for;                     end for;                    end for;
```

Figura 1.5: Fragmento de código del ejemplo cuando se paraleliza en tres procesadores.