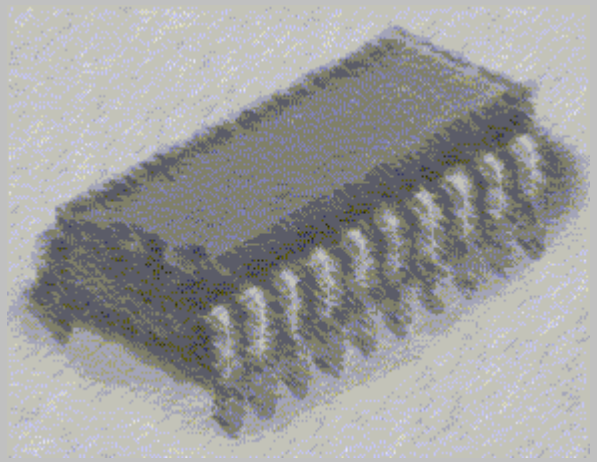
Tema 7. Mejora del rendimiento: introducción a la segmentación y a las arquitecturas paralelas



Arquitectura de Computadores

I. T. Informática de Gestión

Curso 2009-2010

Índice

Introducción (I)

- La arquitectura Von Neumann tiene limitaciones debido a que la memoria actúa como cuello de botella
- La tecnología sobre la que se implementa la arquitectura Von Neumann también limita la velocidad que puede alcanzar
- Interesa aumentar:
  - El ***throughput*** (cantidad de cálculo realizada en el tiempo) de los computadores
  - La **eficiencia** relación entre la velocidad y la cantidad de procesadores utilizados

Introducción (II)

- Como solución surge la idea del procesamiento paralelo para poder favorecer los procesos concurrentes
- Tipos de paralelismo
  - Paralelismo interno (con una única CPU)
    - Segmentación
    - División funcional
  - Paralelismo explícito (con varias CPUs)
- Existe un límite a la eficiencia añadiendo procesadores

Aplicando la ley de Amdahl si se quiere ejecutar un 90% de las instruccio nes de manera paralela, podemos ver que el aumento de velocidad no es mayor que 10 veces

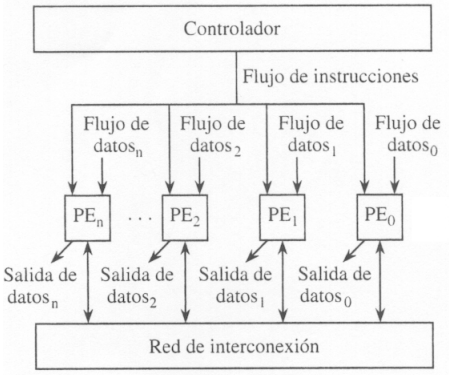
$$S = \frac{1}{(1-f)+\frac{f}{p}} \qquad S = \frac{1}{0{,}1+\frac{0{,}9}{10}} = 5{,}3 \qquad S = \frac{1}{0{,}1+\frac{0{,}9}{\infty}} = 10$$

$$p = 10 \ procesadores \qquad p = \infty \ procesadores$$

Taxonomía de Flynn (I)

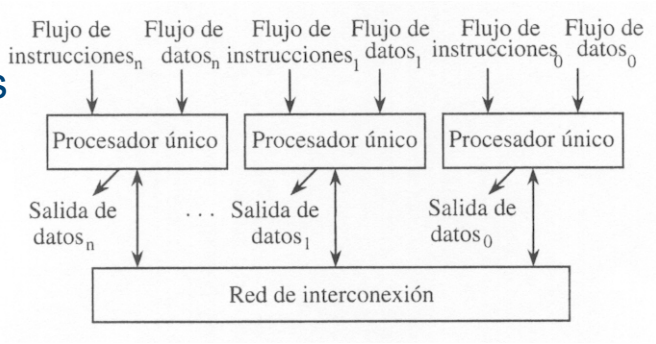
- Clasifica las diferentes arquitecturas según sus flujos de datos y de instrucciones
  - **SISD** (single instruction stream, single data stream) Se ejecutará una única instrucción en cada instante de tiempo. Sin embargo, con la segmentación, puede que en un instante varias instrucciones se encuentren en diferentes fases de ejecución

  - **SIMD** (single instruction stream, multiple data stream) Varios procesadores idénticos realizan la misma secuencia de operaciones sobre distintos conjuntos de datos

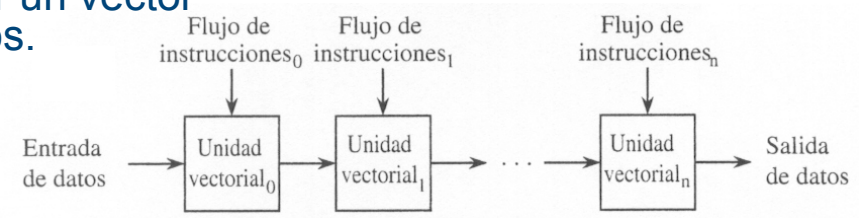


Taxonomía de Flynn (II)

- Clasifica las diferentes arquitecturas según sus flujos de datos y de instrucciones
  - **MIMD** (multiple instruction stream, multiple data stream) Varios procesadores distintos, realizan operaciones distintas sobre datos, también distintos, pero todos ellos se hallan coordinados con el objeto de ejecutar un único programa paralelo

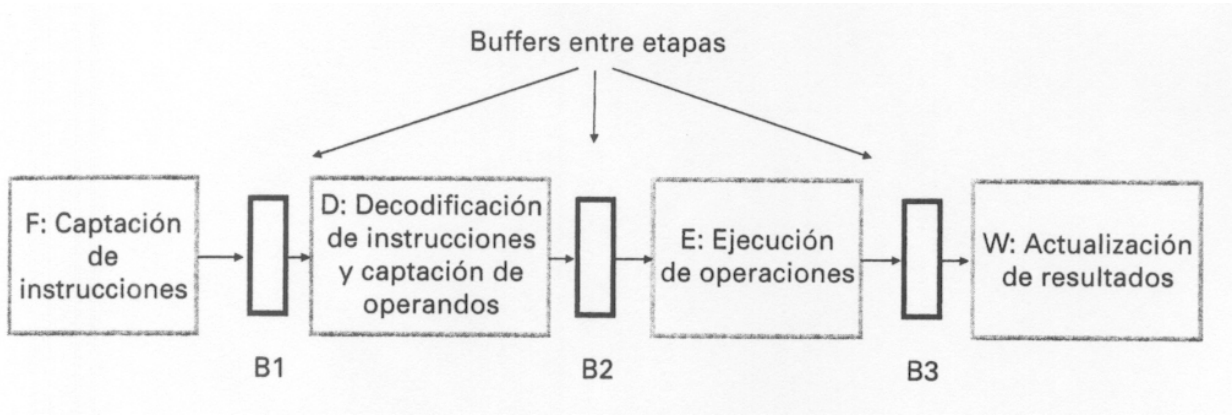


  - **MISD** (multiple instruction stream, single data stream) Varias unidades funcionales actúan sobre un único conjunto de datos. El conjunto de datos suele ser un vector de flujos de datos relacionados.



Paralelismo interno (I)
Segmentación (I)

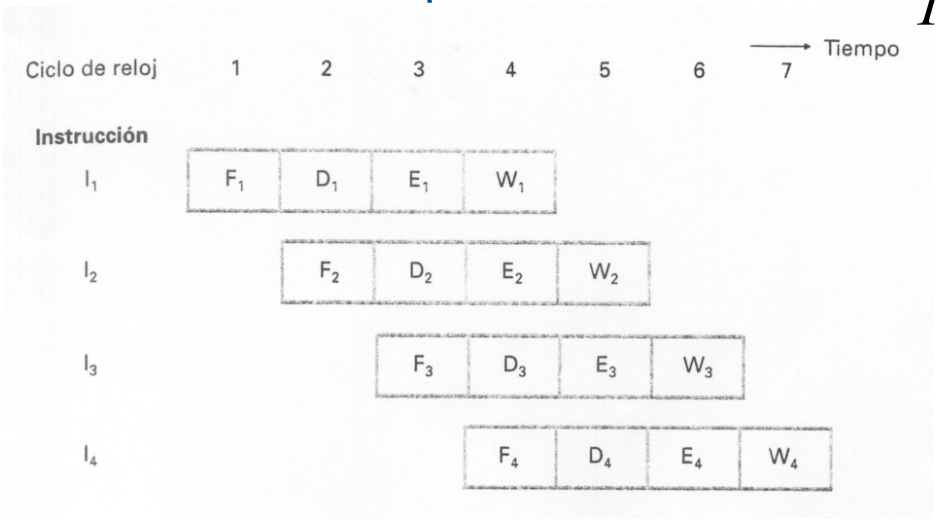
- Consiste en dividir al ejecución de un proceso en etapas consecutivas que se pueden realizar de manera independiente
- En el caso de la segmentación de instrucciones, las distintas fases por la que pasa una instrucción máquina se pueden segmentar, haciendo que **la cadencia entre fases sea la de la etapa más lenta**
- El tiempo para una instrucción será el mismo que sin segmentación, pero, en conjunto dará la impresión de que se ejecutan más instrucciones por ciclo de reloj



Paralelismo interno (II)
Segmentación (II)

- Si una etapa tarda un tiempo *t* en completarse y tenemos *k* etapas, el tiempo de ejecución de una instrucción sin segmentación será de $k \cdot t$ unidades de tiempo
- La ventaja de la segmentación es que aunque todas las instrucciones tardan el mismo tiempo en ejecutarse, la salida se daría cada *t* unidades de tiempo, por lo que para procesar *n* instrucciones el tiempo total sería:

$$T = k \cdot t + (n-1) \cdot t$$



Paralelismo interno (III)
Segmentación (III)

- La segmentación se puede aplicar tanto a los datos como a las instrucciones de los programas.
- Riesgos de la segmentación
  - **Riesgos estructurales**. Significa que el hardware no puede soportar la combinación de instrucciones que se quieren ejecutar en el mismo ciclo, por ejemplo dos instrucciones accediendo a la memoria
  - **Riesgos de control**. Se debe a la necesidad de tomar una decisión basada en los resultados de una instrucción mientras las otras se están ejecutando, por ejemplo un salto condicional
  - **Riesgos de datos**. Se debe a que una instrucción depende del resultado de otra previa que todavía se encuentra en el cauce
- Para favorecer la segmentación a la hora de diseñar el repertorio de instrucciones se debe tener en cuenta:
  - Que todas las instrucciones tengan la misma longitud
  - Que existan pocos formatos de instrucción y con campos sistemáticos
  - Optar por una arquitectura registro-registro
  - Garantizar que los accesos a memoria se encuentren alineados

Paralelismo interno (IV) Segmentación (y IV)

- Para conseguir procesadores aún más rápidos se han seguido tres direcciones que extienden los procesadores segmentados:
  - **Supersegmentación**. Consiste en hacer más larga la segmentación, dividiendo la misma en un mayor número de etapas. Computadores recientes emplean 8 o más etapas
  - **Superescalar**. Consiste en la replicación de los elementos internos del computador de manera que se puedan ejecutar múltiples instrucciones en cada etapa. Al ejecutar varias instrucciones por ciclo de reloj tendríamos un CPI <1
  - **Segmentación dinámica**. Consiste en que sea le hardware el que evite los riegos estructurales, de datos y de control según se van ejecutando las instrucciones

- La mejora de la velocidad en los casos anteriores hace que se complique el control de la segmentación y el modelo de ejecución de las instrucciones

Paralelismo interno (V)
Computadores vectoriales (I)

- **Computadores vectoriales** son máquinas segmentadas que incluyen instrucciones máquina que operan sobre vectores
- Las operaciones vectoriales se caracterizan por repetir la misma operación sobre los elementos de un vector
- **Ventajas:**
  - En cada instrucción vectorial, el cálculo de cada componente del vector resultado es independiente del resto de componentes con lo que se reduce el número de dependencias de datos
  - Una instrucción vectorial supone gran cantidad de trabajo reduciendo el cuello de botella de la memoria de instrucciones
  - Una instrucción vectorial equivale a un bucle de instrucciones escalares, sin la sobrecarga de las instrucciones de control de bucle, ni de los riesgos de control
  - Una instrucción vectorial que accede a memoria tiene un patrón de acceso conocido a priori con lo que las organizaciones de memoria de múltiples módulos funcionan *sin intervención* de una ante memoria.

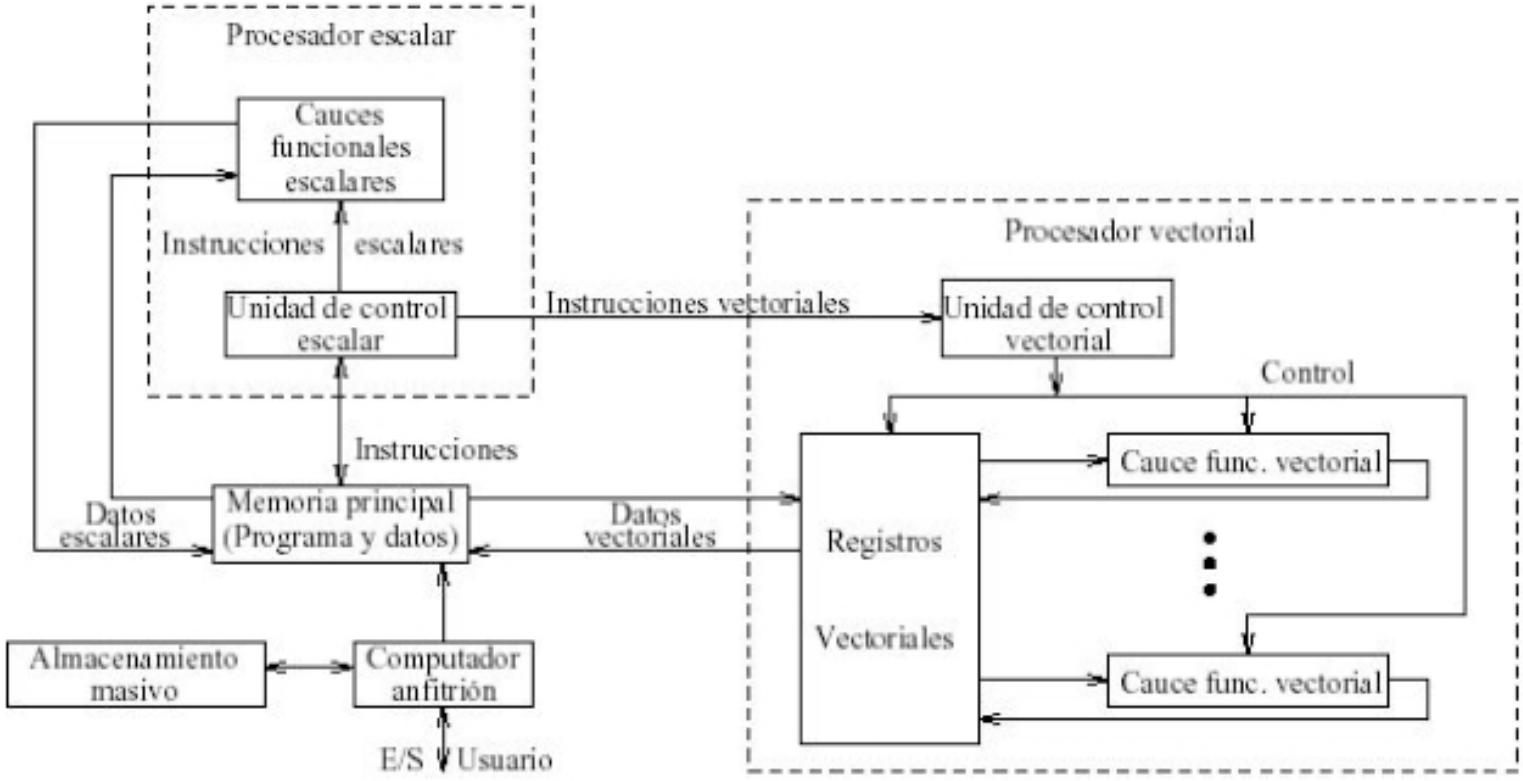
Paralelismo interno (VI)
Computadores vectoriales (II)

- Existen dos tipos de máquinas con arquitectura del juego de instrucciones vectorial:
  - Computadores vectoriales segmentados con una unidad aritmético-lógica segmentada en muchas etapas
  - Computadores SIMD con múltiples unidades aritmético-lógicas

- Tipos de computadores vectoriales segmentados:
  - **Con registros vectoriales**. Los operandos vectoriales se encuentran en registros vectoriales, con instrucciones para cargarlos desde memoria o almacenarlos en memoria
  - **Memoria-memoria**. Los operandos vectoriales se encuentran en la memoria principal con lo que el tamaño de los operandos vectoriales es ilimitado pero todas las instrucciones sufren el acceso a memoria

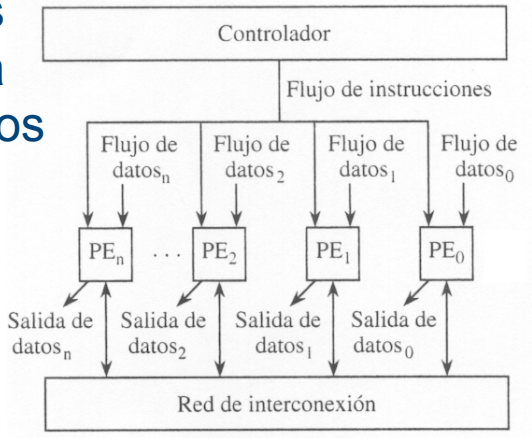
Paralelismo interno (y VII)
Computadores vectoriales (y III)

- La estructura de un computador vectorial es:



Paralelismo externo (I)
Computadores matriciales (I)

- Los computadores matriciales están formados por varias unidades de proceso que realizan la misma secuencia de operaciones sobre distintos conjuntos de datos de manera simultánea controlados por una única unidad de control



- Responden al modelo SIMD de Flynn

- Pueden realizar las operaciones sobre matrices, grupos de datos o vectores diferenciándose de las vectoriales en que en éstas cada elemento se procesa de manera simultánea

Paralelismo externo (II)
Computadores matriciales (y II)

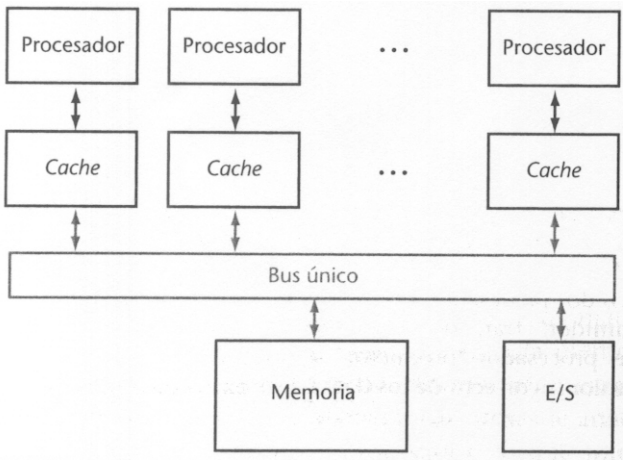
- Los computadores matriciales pueden ser de dos tipos:
  - **Memoria compartida**. Los módulos de memoria son compartidos por todos los elementos de proceso. Para evitar los conflictos de acceso, se divide la memoria en un número de módulos independientes de memoria diferente al de las unidades de ejecución
  - **Memoria distribuida.** En este esquema cada elemento de proceso posee su propio módulo de memoria con datos locales. La manera de llevarlo a cabo es distribuyendo los datos antes de la ejecución
- Dado que no todos los elementos de la colección de datos pueden tener que usarse en todas las instrucciones, proporcionan mecanismos para dejar ociosas a las unidades de ejecución que no se requieran

Paralelismo externo (III)
Multiprocesadores y multicomputadores (I)

- **Multiprocesador** es un único computador formado por múltiples procesadores que comparten un mapa de memoria común
- **Multicomputador** es un sistema formado por varios computadores independientes, cada uno con su mapa de memoria. También son llamados cluster
- Tanto multiprocesadores como multicomputadores emplean varios procesadores distintos que realizan operaciones distintas sobre datos, también distintos, pero todos ellos se hallan coordinados con el objeto de ejecutar un único programa paralelo. Responden al modelo MIMD de la taxonomía de Flynn

Paralelismo externo (IV)
Multiprocesadores y multicomputadores (II)

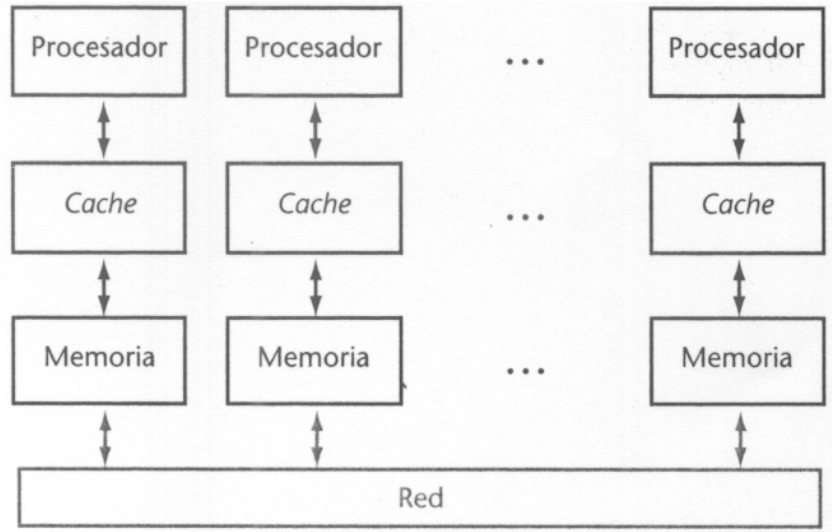
- **Procesadores de memoria compartida**. Ofrecen al programador un único espacio de direcciones que comparten todos los procesadores que se comunican a través de variables compartidas. A su vez se dividen en:
  - **Con Acceso Uniforme a Memoria (UMA) o multiprocesadores simétricos (SMP)**. El tiempo de acceso a memoria es el mismo para todos los procesadores
  - **Con Acceso NO Uniforme a memoria (NUMA)**. Los accesos son más o menos rápidos dependiendo del procesador que los hace y de la palabra accedida



Paralelismo externo (V)
Multiprocesadores y multicomputadores (III)

- **Procesadores con memoria distribuida**. Los procesadores se comunican mediante paso de mensajes ya que cada procesador cuenta con su propia memoria privada

- Son necesarios los mecanismos de caché con protocolos de escucha vistos en el tema de sistemas de memoria



Paralelismo externo (y VI)
Multiprocesadores y multicomputadores (y IV)

- **Multicomputadores o clusters**. Sistemas formados por dos o más computadores, unidos por una red de alta velocidad y que trabajan de manera conjunta para resolver una tarea comportándose como si fuese un único computador
- A cada elemento de un multicomputador se le denomina nodo, pudiendo ser estos muy diferentes entre si
- **Ventajas:**
  - Tolerantes a fallos
  - Facilita la expansión del sistema
  - Potencia de cálculo equivalente a grandes máquinas más asequible
- **Inconvenientes:**
  - Coste de administración
  - Conexión al bus de E/S en lugar de al de memoria
  - Memoria independiente por nodo

Programación en sistemas paralelos

- En la mayor parte de los sistemas paralelos, quitando en el de la segmentación que es transparente al programador, se necesita un nuevo paradigma de programación
- La programación tiene que aprovechar al máximo el paralelismo de las máquinas
- En las máquinas vectoriales existen dos alternativas:
  - Lenguajes de programación vectoriales
  - Diseñar compiladores capaces de vectorizar
- En las máquinas matriciales, multiprocesador o multicomputador se emplea:
  - **Multihebra**. Para la programación de sistemas con memoria compartida
  - **MPI**. Para la programación en sistemas con memoria distribuida

Bibliografía

- Estructura y diseño de computadores
  David A. Patterson y John L. Hennessy. Reverté, 2000
  Capítulo 6 y 9
- Arquitectura de computadores
  José A. de Frutos y Rafael Rico. Servicio de Publicaciones de la Universidad de Alcalá, 1995
  Capítulo 7 y 8
- Principios de arquitectura de computadores
  Miles J. Murdocca y Vicent P. Heuring. Prentice Hall 2002
  Capítulo 10
- Arquitectura de computadores. Un enfoque cuantitativo
  John L. Hennessy y David A. Patterson. Mc Graw Hill, 2ª ed, 1996
  Capítulo 3, 4 y 8