Conceptos básicos de procesamiento paralelo (1)

- **Paralelismo**: En un sistema computador hay paralelismo cuando, al menos, **durante algunos instantes de tiempo** ocurren **varios eventos similares**
- Ejecución **concurrente vs. paralela**:
  - Concurrente: Modelo de varios clientes y un servidor. Es secuencial pero de apariencia paralela
  - Paralela: Modelo de N clientes y N servidores (ejemplo: N procesadores y N procesos)

Conceptos básicos de procesamiento paralelo (2)

- **Granularidad**
  - Indica el tamaño de las tareas en que se descomponen los programas con objeto de paralelizarlos
    - **Tareas pequeñas**: **grano** de paralelismo **fino**
    - **Tareas grandes**: **grano** de paralelismo **grueso**
  - Grano más fino supone:
    - Inconveniente: mayores dificultades de sincronización de tareas
    - Ventaja: mayor aprovechamiento del paralelismo

Conceptos básicos de procesamiento paralelo (y 3)

- **Procesos y *threads*** (hilos, procesos ligeros)
  - Ambos consisten en la tarea de ejecutar un fragmento de programa y se crean para poder ejecutarse concurrentemente o en paralelo
  - El S.O. asigna recursos distintos a cada proceso (espacio de memoria, procesador...) para su ejecución
  - Los *threads* se crean dentro de un proceso y comparten los recursos del proceso
    - La existencia de *threads* supone un grano de paralelismo más fino que la existencia de procesos sin *threads*

Tipos y niveles de paralelismo (1)

- Paralelismo **disponible**: el que hay en las soluciones a los problemas
  - **Funcional**:
    - Surge de la lógica de la solución de un problema (ej.: varias instrucciones de un algoritmo que pueden ejecutarse a la vez)
    - Es irregular y débil (se paralelizan menos de 10 operaciones) pero aparece en toda clase de problemas
  - **De datos**:
    - Surge de usar, en la solución a un problema, estructuras de datos que permiten operaciones paralelas en sus elementos (vectores y matrices)
    - Es regular y, generalmente masivo (se paralelizan 100 ó más operaciones)

Tipos y niveles de paralelismo (y 2)

- Paralelismo **utilizado**: el que se da durante la ejecución de los programas
  - Lo utilizan las arquitecturas, los compiladores y los sistemas operativos trabajando conjuntamente
- Niveles

Paralelismo funcional disponible	Paralelismo funcional utilizado
Nivel de instrucción (grano fino)	Nivel de instrucción (arquitecturas ILP y compiladores adecuados)
Nivel de bucle (grano medio)	Nivel de *thread*
Nivel de procedimiento (grano medio)	Nivel de proceso
Nivel de programa (grano grueso)	Nivel de usuario (los programas de diferentes usuarios se dividen en procesos)

Arquitecturas para explotar el paralelismo

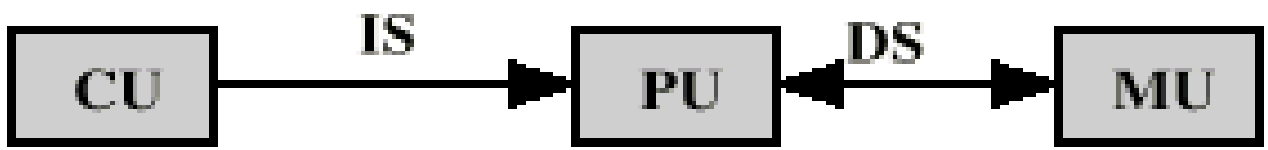
- Dos técnicas básicas
  - Pipeline (segmentación encauzada o encauzamiento)
    - El hardware se divide en etapas
    - El procesamiento de funciones se solapa (antes de que termine una empieza otra)
    - Se suele encauzar la CPU, la ALU, la memoria...
  - Replicación
    - En vez de un elemento hardware se tienen varios iguales que pueden funcionar simultáneamente
    - Se pueden replicar las ALU, los bancos de memoria, las unidades de entrada/salida, los procesadores...

Clasificación de Flynn de los computadores

- Single instruction, single data stream - SISD
- Single instruction, multiple data stream - SIMD
- Multiple instruction, single data stream - MISD
- Multiple instruction, multiple data stream- MIMD

Single Instruction, Single Data Stream - SISD

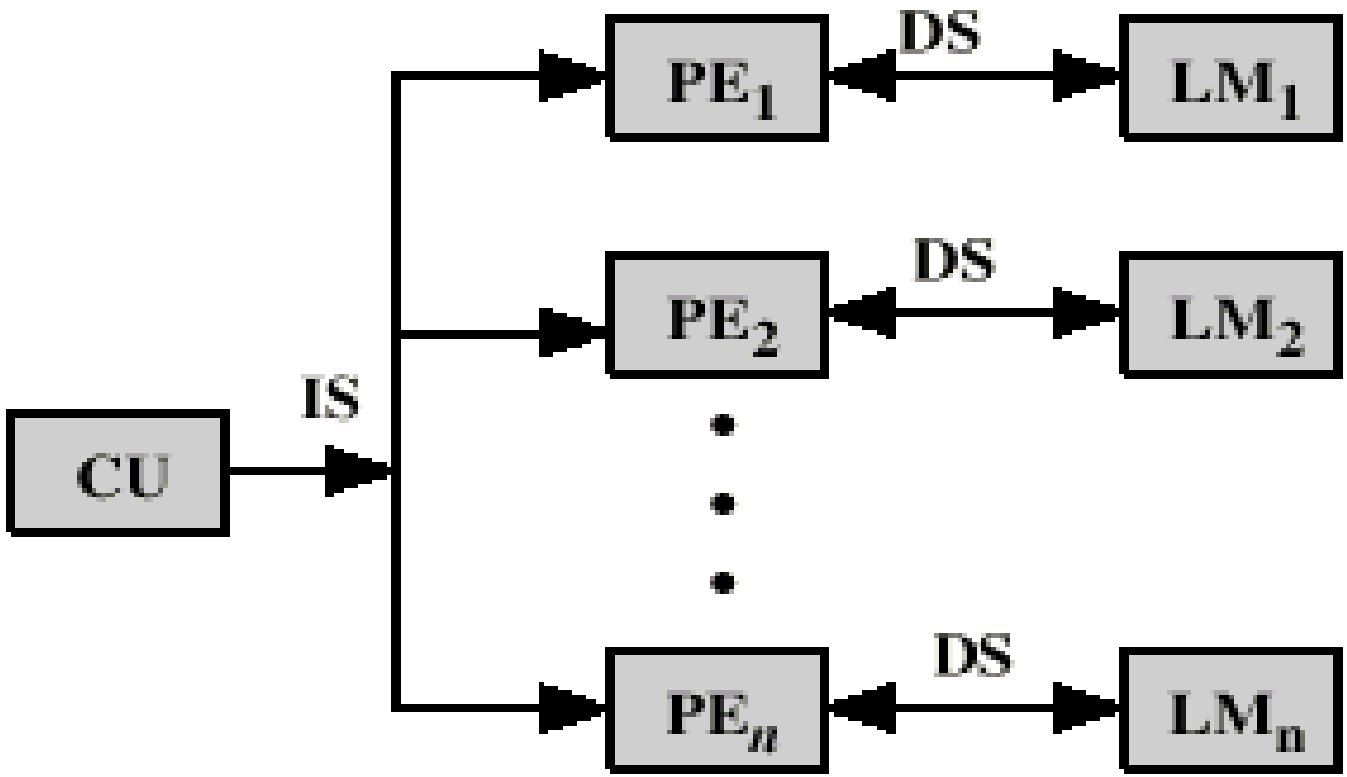
- Un único procesador (C.U. + A.L.U.)
- Un único flujo de instrucciones entre la memoria y la unidad de control
- Un único flujo de datos entre la memoria y la unidad procesadora (ALU o PU)



Single Instruction, Multiple Data Stream - SIMD

- Una única unidad de control
  => "Single instruction stream"
- Varios elementos de proceso (ALU)
  => "Multiple data stream"
- Los elementos de proceso funcionan síncronamente
- Cada elemento de proceso tiene su propia memoria local de datos
- Procesadores matriciales

Organización SIMD



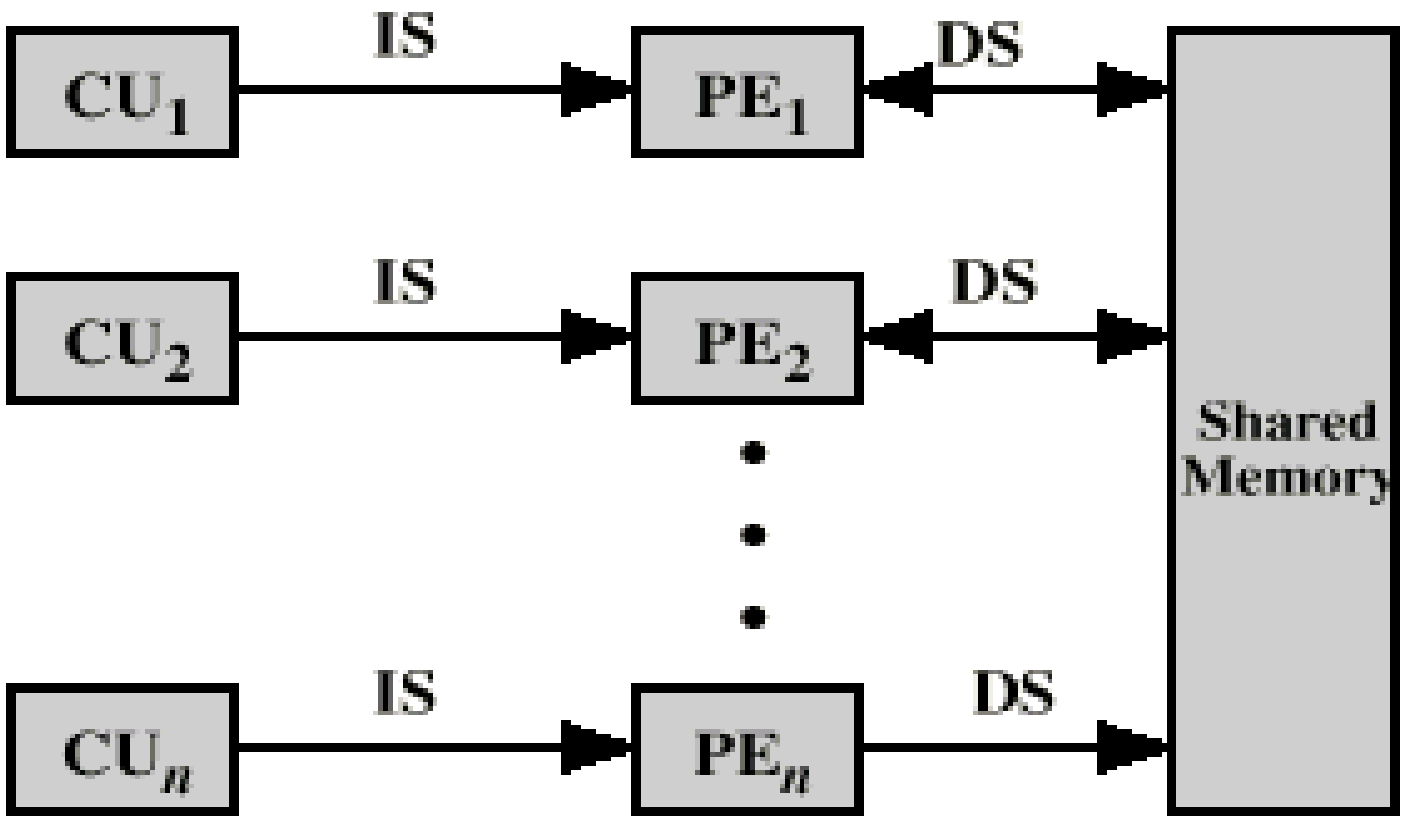
Multiple Instruction, Single Data Stream - MISD

- Una única secuencia de datos es transmitida a través de una serie de elementos de proceso
- Cada procesador ejecuta una secuencia de instrucciones diferente sobre un mismo flujo de datos
- Algunos incluyen a los procesadores sistólicos en esta categoría
- Otros opinan que no se ha construido ninguna computadora con esta arquitectura

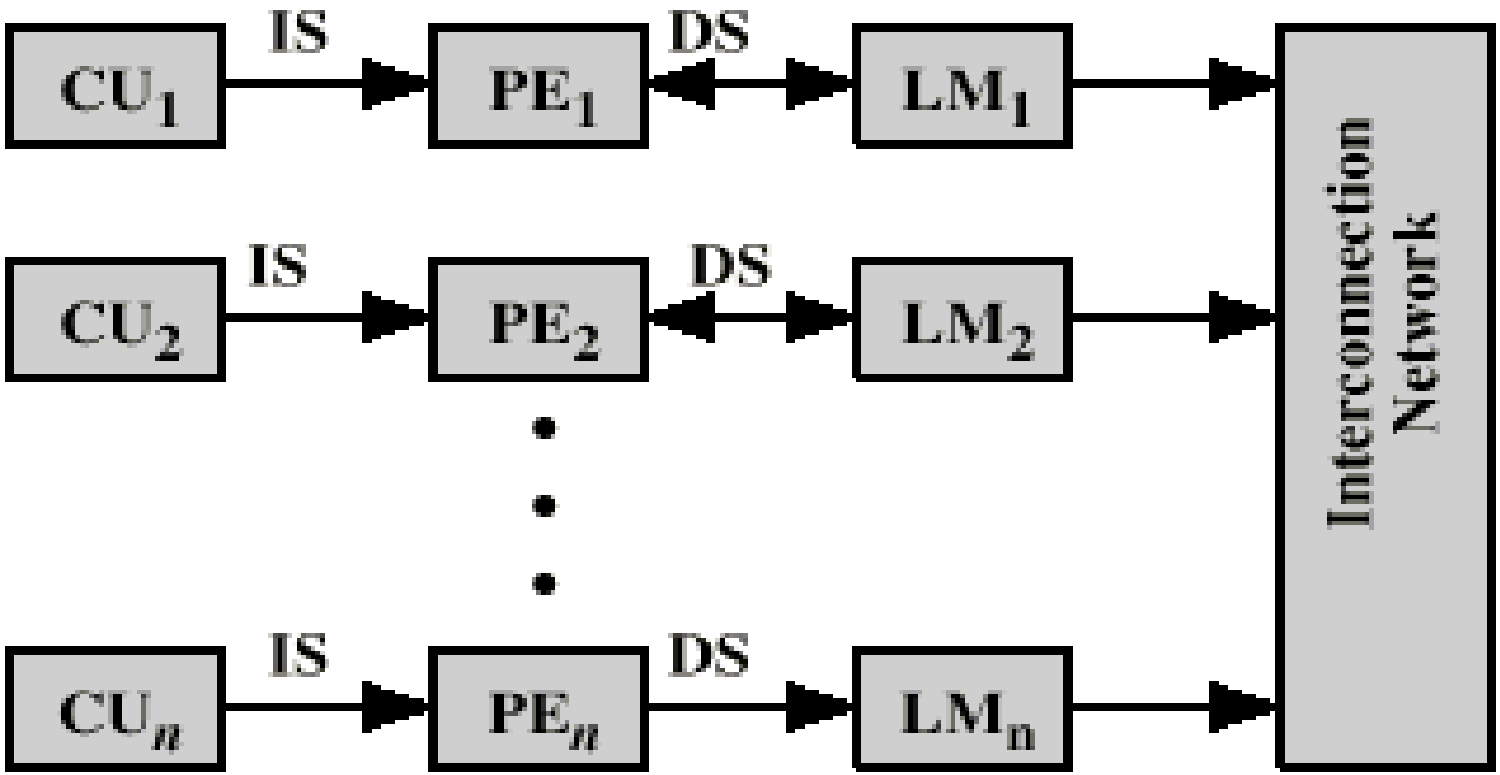
Multiple Instruction, Multiple Data Stream- MIMD

- Varios procesadores
- Si
  - Memoria compartida entonces “multiprocesadores”
  - Memoria distribuida entonces “multicomputadores”

Organización MIMD
Memoria compartida



Organización MIMD
Memoria distribuida



Clasificación de Sima, Fountain y Kacsuk [1997]

- Arquitecturas de paralelismo funcional
  - Arquitecturas ILP (paralelas a nivel de instrucción)
    - Procesadores encauzados
    - Procesadores VLIW
    - Procesadores superescalares
  - Arquitecturas paralelas a nivel de thread
  - Arquitecturas paralelas a nivel de proceso (MIMD)
    - MIMD de memoria distribuida o multicomputador
    - MIMD de memoria compartida o multiprocesador
- Arquitecturas de paralelismo de datos
  - Arquitecturas vectoriales
  - Arquitecturas asociativas y neuronales
  - Arquitecturas SIMD
  - Arquitecturas sistólicas

Ejemplo de UCP encauzada (pipelined)

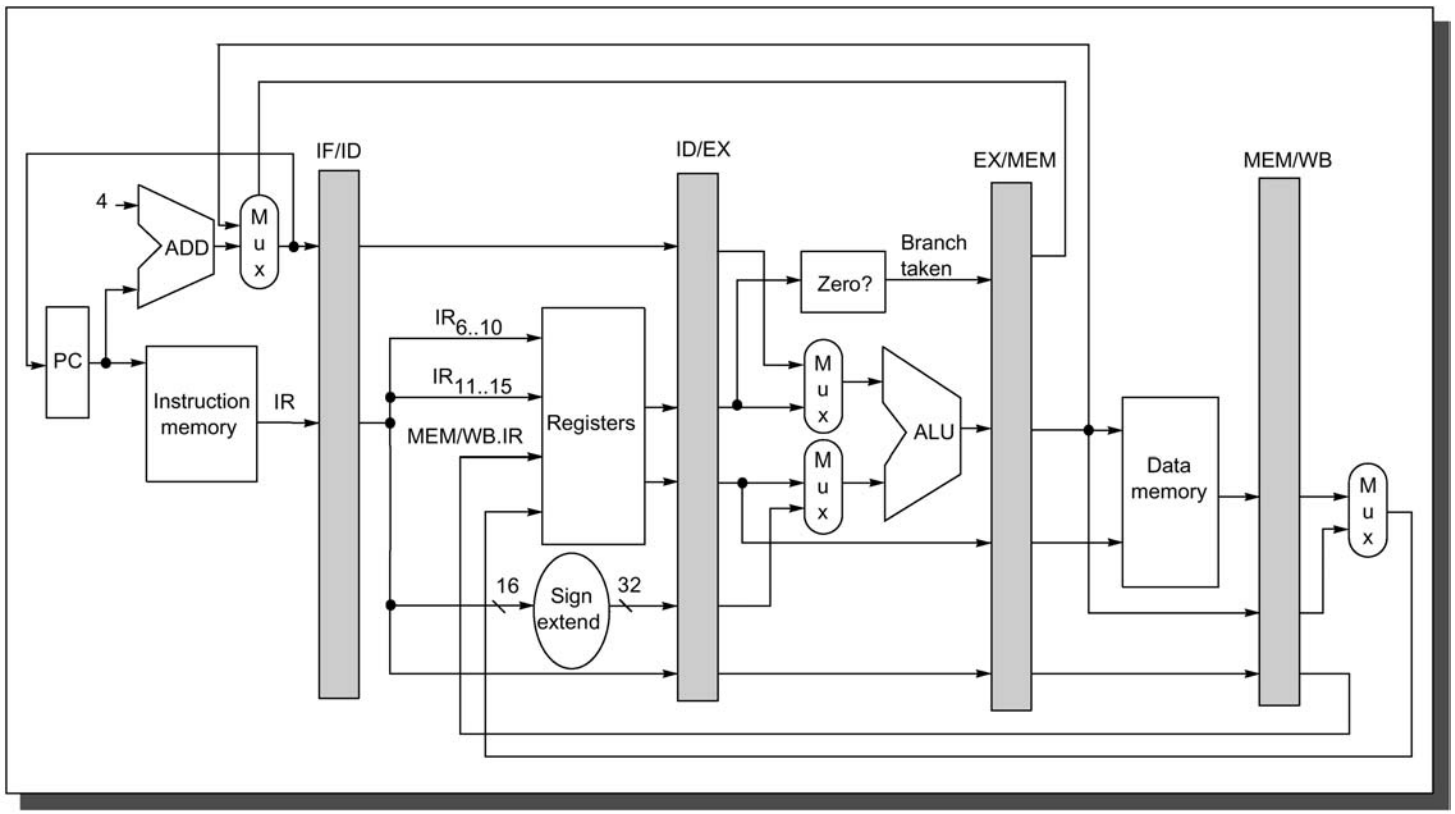
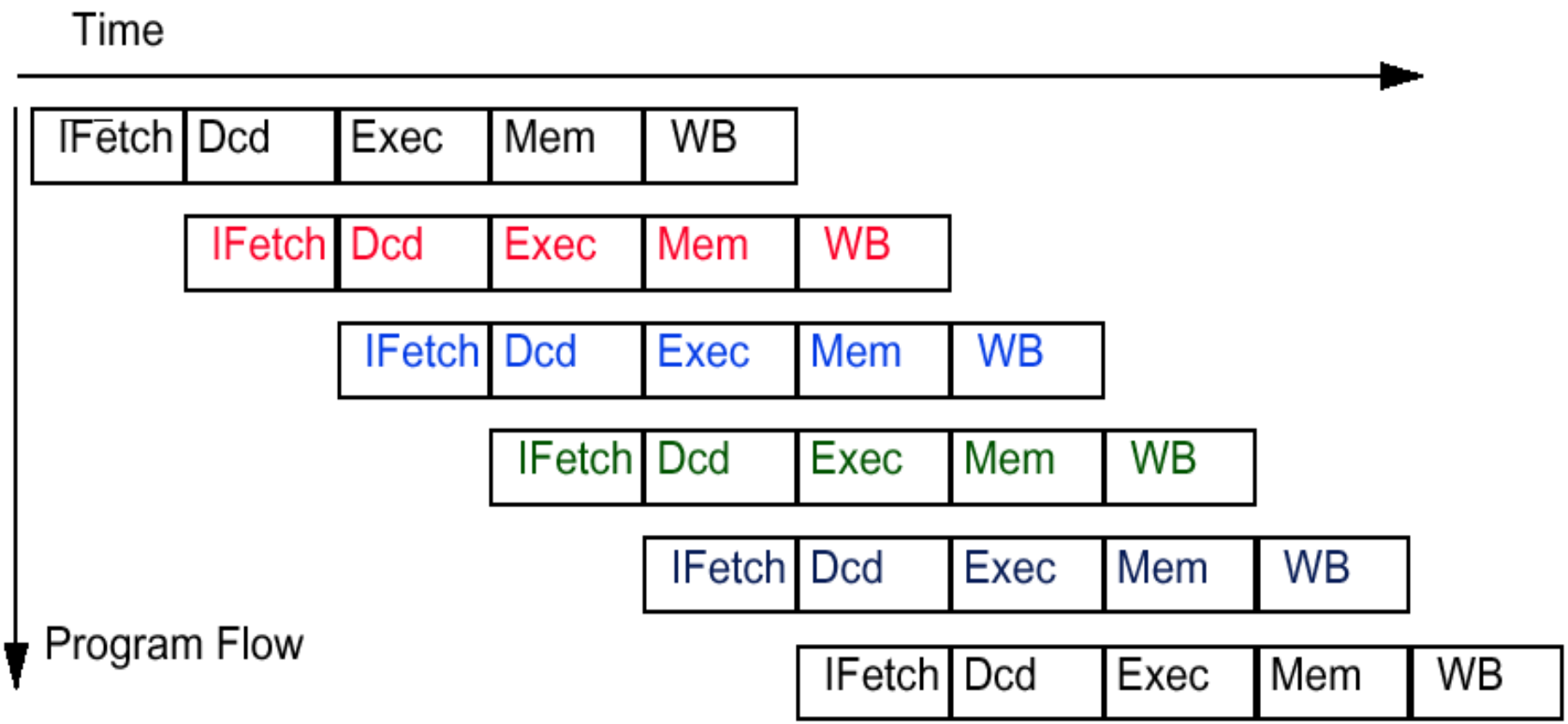


FIGURE 3.4 The datapath is pipelined by adding a set of registers, one between each pair of pipe stages.

Ejecución de instrucciones en una UCP encauzada (caso ideal)



Características generales del pipeline

- La velocidad del cauce está limitada por la de la etapa más lenta =>
  - Conviene que el retardo de todas las etapas sea el mismo (T)
- Rendimiento ideal:
  - Cada periodo T termina la ejecución de una instrucción en el cauce
  - Si T es 1 ciclo de reloj, $CPI_{ideal} = 1$
  - Si hay N etapas la mejora en velocidad es N (=NT/T)

¿Por qué no se consigue el rendimiento ideal?

- No es fácil que todas las etapas tengan el mismo retardo
- Hay retardos debidos a los registros que separan etapas
- Mientras el cauce se está llenando o vaciando hay etapas inactivas
- En algunas situaciones de riesgo (*hazard*) es preciso detener instrucciones en el cauce

Hazards

- Colisiones: Varias instrucciones intentan acceder a un mismo recurso al mismo tiempo
- Dependencias de control: Cuando hay una instrucción de salto puede ser preciso esperar a que se calcule el nuevo valor del PC
- Dependencias de datos entre dos instrucciones I y J (I anterior a J)
  - **Leer después de escribir**: J lee un operando que previamente debe calcular I
  - **Escribir después de leer**: J escribe un resultado que I debe leer antes
  - **Escribir después de escribir**: I escribe un resultado que J debe escribir después

Procesadores superescalares y VLIW (*Very Long Instruction Word*)

- Superescalares y VLIW...
  - Consiguen un CPI<1
  - Poseen varias unidades de ejecución
- Procesadores VLIW:
  - Cada instrucción codifica varias operaciones (lo que en una máquina convencional serían varias instrucciones)
  - Las operaciones de una instrucción se ejecutarán en paralelo en las distintas unidades de ejecución
  - Precisan compiladores específicos
    - Planificación estática
  - Entre 5 y 30 unidades de ejecución

Procesadores superescalares y VLIW (cont.)

- Procesadores superescalares
  - Descodifican un número pequeño de instrucciones (entre 2 y 6) en cada ciclo de reloj
    - Si pueden paralelizarse serán todas emitidas en un mismo ciclo
    - La planificación de las instrucciones es responsabilidad del compilador y del hardware
  - No precisan compiladores especiales
  - Menos unidades de ejecución que en un procesador VLIW
  - Gran implantación en los procesadores comerciales

Problemas con los procesadores superescalares y VLIW

- Cuanto mayor sea el número N de unidades de ejecución
  - Mayor dificultad para encontrar un conjunto de N operaciones entre las que no haya dependencias de datos
  - La frecuencia de saltos por cada emisión de instrucción/instrucciones es mayor
  - En definitiva, mayor dificultad para tener ocupadas todas las unidades de ejecución
- Se necesitan bancos de registros y memorias multipuerto para que varias operaciones puedan acceder a ellos a la vez

Características básicas de los computadores MIMD

- Contienen varios procesadores, generalmente de similares capacidades de cómputo
- No hay una unidad central de control (a diferencia de lo que ocurre en un SISD o en un SIMD)
- Comunicación entre procesadores:
  - Mediante memoria compartida o
  - Mediante paso de mensajes
- El hardware es manejado por un único sistema operativo

Memoria en los MIMD

- La distinción memoria compartida / memoria distribuida es demasiado simple. Debe considerarse
  - Espacio de direcciones **compartido** frente a espacios de direcciones **privados**
  - Memoria **centralizada** frente a memoria **distribuida**
    - Centralizada: igual tiempo de acceso para todos los procesadores
    - Distribuida: distintos módulos de memoria y diferentes tiempos de acceso según el procesador y el módulo de memoria considerados

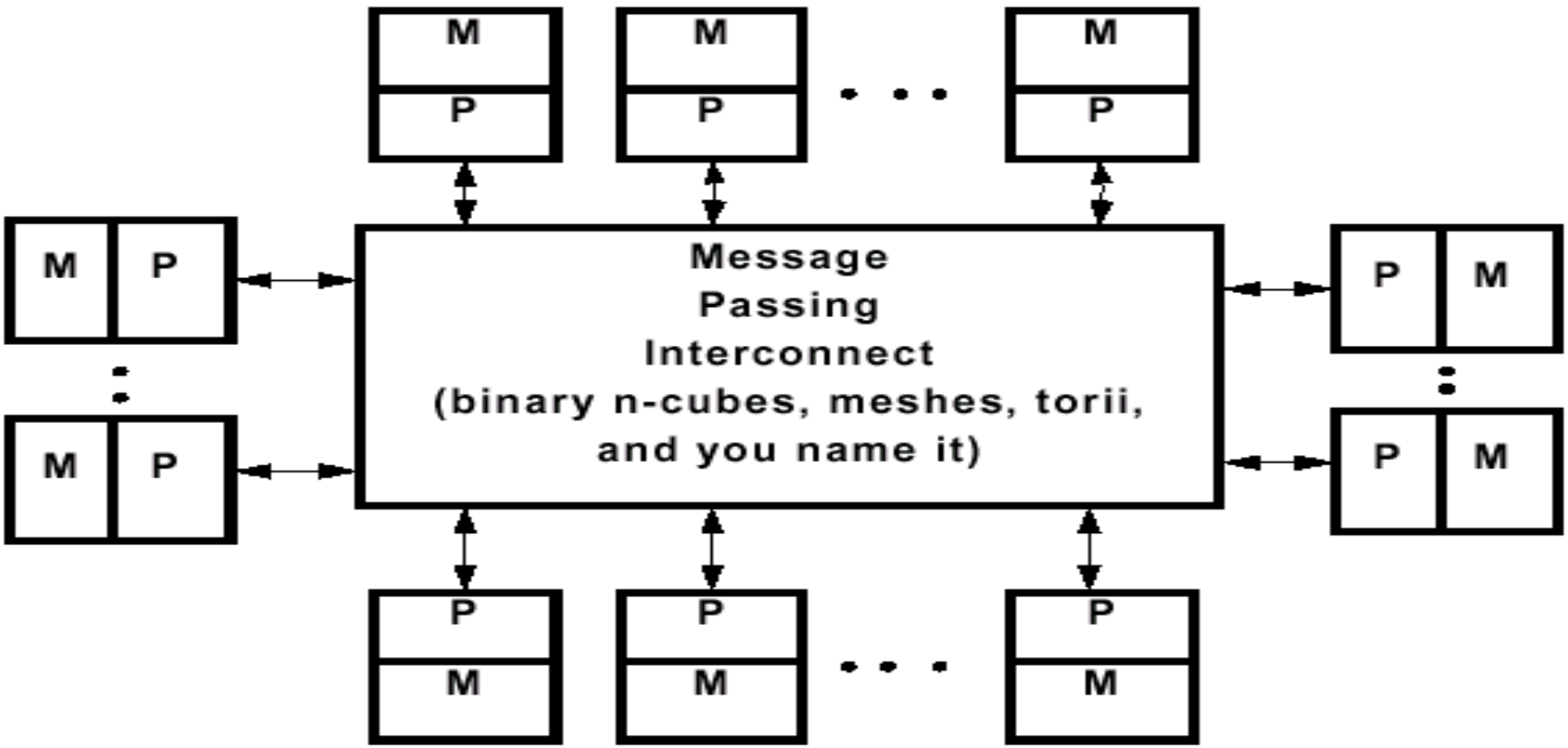
Memoria en los MIMD (cont)

- En general
  - Las máquinas con espacios de direcciones **privados** son de memoria **distribuida** (clusters) y
  - Las máquinas con memoria **compartida** son de memoria **centralizada** (**UMA**, *uniform memory access)*
- Sin embargo
  - La memoria compartida facilita el software
  - La memoria distribuida facilita el hardware
  - Así se justifica la existencia de arquitecturas con memoria **distribuida** y espacio de direcciones **compartido**: se denominan arquitecturas de memoria virtual compartida, de memoria distribuida compartida o **NUMA** (*Non-uniform memory access*)

Ventajas e inconvenientes de los modelos de memoria

- Escalabilidad
  - Mayor en los MIMD de memoria distribuida
  - En los MIMD de memoria compartida el ancho de banda de la memoria limita el número de procesadores que se pueden tener sin que la memoria se convierta en el cuello de botella. Sin embargo...
    - ¿Hay suficiente paralelismo en los programas para aprovechar un número elevado de procesadores?
- Programación
  - Más sencilla en los multiprocesadores

Multicomputadores

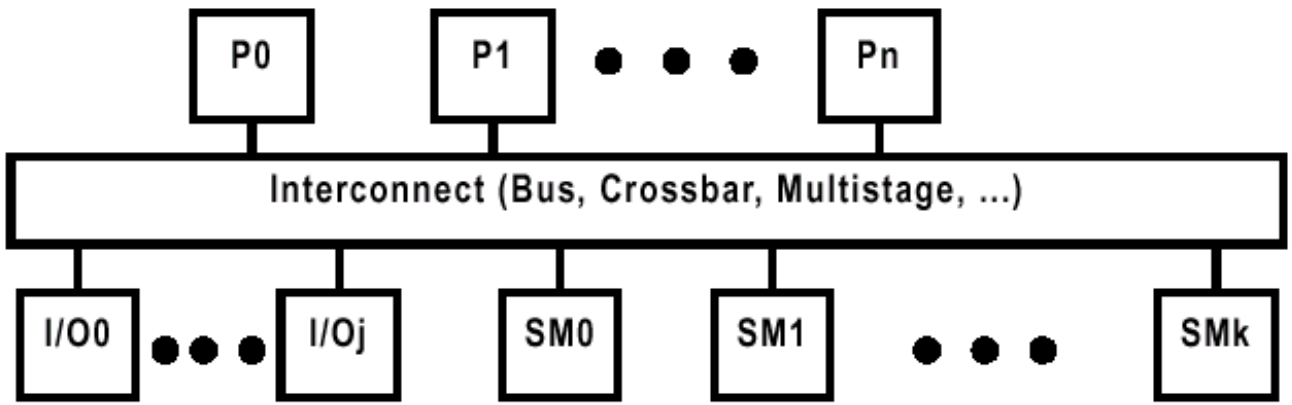


Multicomputadores

- Son una colección de computadores (CPU, memoria, elementos de E/S) que se comunican:
  - A través de una **red escalable, de alto rendimiento**
    - Muy importante la topología de la red
  - Mediante **operaciones explícitas de E/S**
    - Send: **SEND expresión TO proceso destino**
    - Receive: **RECEIVE variable FROM proceso fuente**
    - Cuando un proceso emisor y uno receptor **se nombran mutuamente** se establece un **canal lógico de comunicación** entre ambos
    - *Send* y *receive* resuelven la **sincronización** entre procesos
- Ejemplos: Schlumberger FAIM- 1, HPL Mayfly, NCUBE, Intel iPSC, Parsys SuperNode1000, Intel Paragon

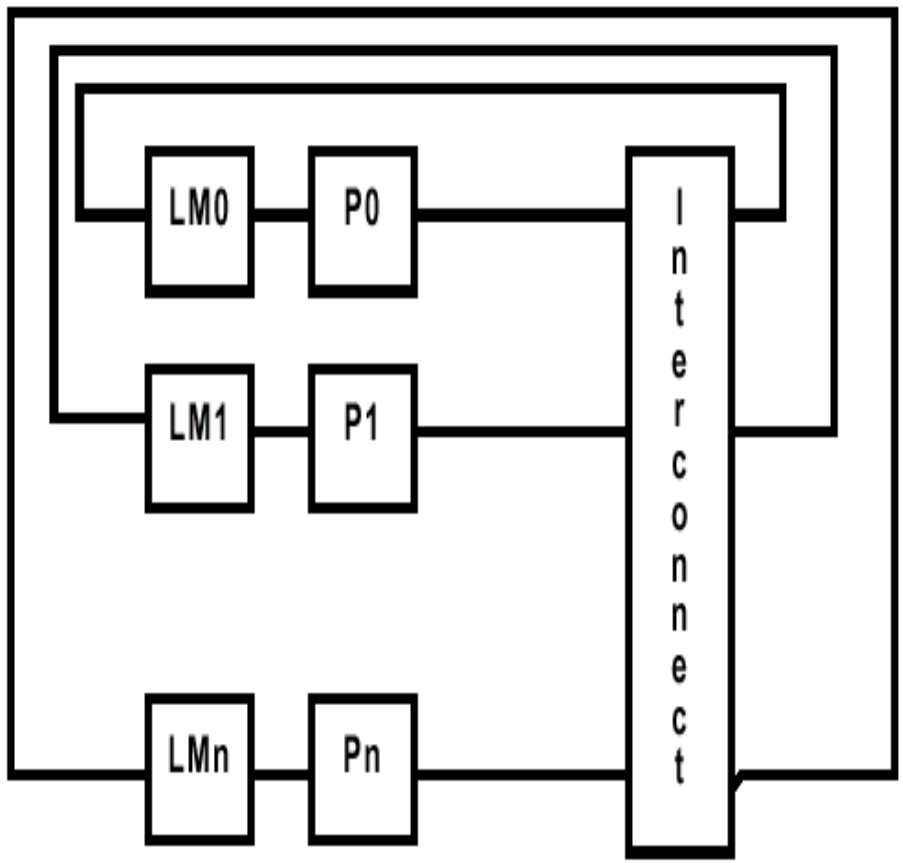
Arquitecturas UMA (multiprocesadores simétricos)

- Todos los procesadores comparten
  - Memoria (con similares tiempos de acceso)
  - Entrada/salida
- Propia de multiprocesadores de pequeña escala
- Ejemplos: Sun Enterprise 6000, SGI Challenge, Intel SystemPro



Arquitecturas NUMA

- El tiempo de acceso de un procesador a memoria es diferente en diferentes zonas de memoria
- Propia de multiprocesadores de gran escala
- Muchas variantes
- Ejemplos: Cray T3D, KSR-1, Convex SPP1000



Comunicación y sincronización en los multiprocesadores

- Comunicación mediante variables compartidas =>
  - Aparecen **secciones críticas**. Ej.:
  - *LOAD Reg[1],V* / *ADD Reg[1],#1* / *STORE V,Reg[1]* — *LOAD Reg[2],V* / *ADD Reg[2],#2* / *STORE V,Reg[2]*
- Sincronización mediante instrucciones ininterrumpibles específicas. Ej.:
  - *WAIT(S)* / Sección crítica 1 / *SIGNAL(S)* — *WAIT(S)* / Sección crítica 2 / *SIGNAL(S)*
  - S es una variable compartida (suponemos inicialmente S=1)
  - *WAIT(S) = [**If** S=0 **then** "impedir el paso" **else** S ← S-1]*
  - *SIGNAL(S) = [S ← S+1]*

Cachés en multiprocesadores

- Ventajas
  - Reducen la competencia por memoria
  - Evita conflictos y retardos en la red de interconexión
- Problema: coherencia caché
  - Caso típico:
    - Dos **procesadores P y Q leen** una **variable compartida**. **Después P la modifica**. **Si Q vuelve a leerla**, tomará el dato de su memoria caché y por tanto tomará un **dato obsoleto**
  - Tipos de protocolos para mantener la coherencia caché
    - De espionaje (*snooping*). Propios de arquitecturas de un solo bus
    - Basados en directorio. Propios de arquitecturas que soportan un número elevado de procesadores

Problemas de los MIMD escalables y posibles soluciones

- Problemas de latencia de memoria
- Pérdidas de tiempo debidas a la sincronización entre procesos
- Posibles soluciones
  - Uso de memoria **caché**
  - **Red de interconexión** eficiente
  - **Conmutación de procesos**: cuando un proceso quiere acceder a un recurso ocupado, al procesador se le asigna otro proceso para evitar que esté parado
    - Problema: el estado del proceso suspendido debe ser salvado y el del proceso activado debe ser cargado (cambio de contexto)
  - Utilización de ***threads*** junto a un **mecanismo rápido de cambio de contexto** entre *threads*

Arquitecturas multithread

- Grano más fino que las multitarea (*thread* más pequeño que proceso) =>
  - Más importante la **velocidad** en el **cambio de contexto**
  - Necesidad de **mecanismo hardware** para ese cambio
- Eficiencia procesador = busy /(busy+switching+idle)
  - *Busy, switching, idle*: cantidad de tiempo, medida sobre un largo intervalo, que el procesador está haciendo trabajo útil, cambiando de contexto y parado, respectivamente
  - Objetivo multithread: reducir *idle* sin que *switching* sufra un aumento sustancial (en todo caso es exigible que *switching* < *idle*)

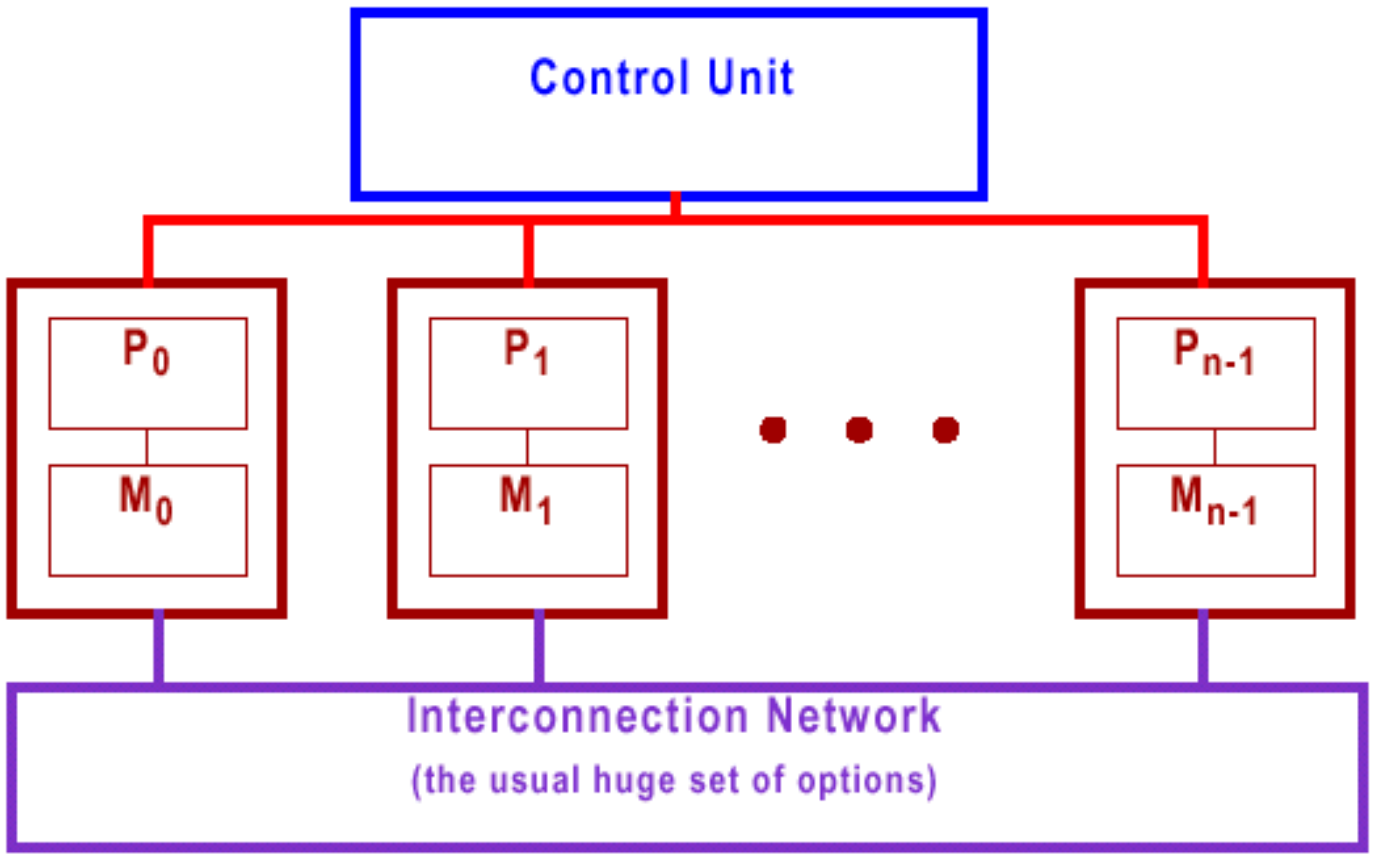
Arquitecturas multithread (cont)

- Principales decisiones de diseño:
  - Modelo von Neumann / modelo híbrido (von Neumann-flujo de datos)
  - Granularidad fina/gruesa
  - Número de *threads* pequeño (4-10) / mediano (10-100) / grande (más de 100)
  - Memoria centralizada compartida / distribuida compartida

Comparación de máquinas multithread

Máquina	Modelo com-putacional	Granularidad	Organización de memoria	*Threads* por procesador	Año de aparición
Denelcor HEP	Von Neumann	Fina	Centralizada compartida	Hasta 64	1978
Tera	Von Neumann	Fina	Distribuida compartida	Hasta 128	1990
MIT Alewife	Von Neumann	Gruesa	Distribuida compartida	Hasta 3	1990
EM-4	Híbrida-flujo de datos	Gruesa	Distribuida compartida	Ilimitado	1990
***T**	Híbrida-Von Neumann	Fina	Distribuida compartida	Ilimitado	1992

Arquitectura SIMD



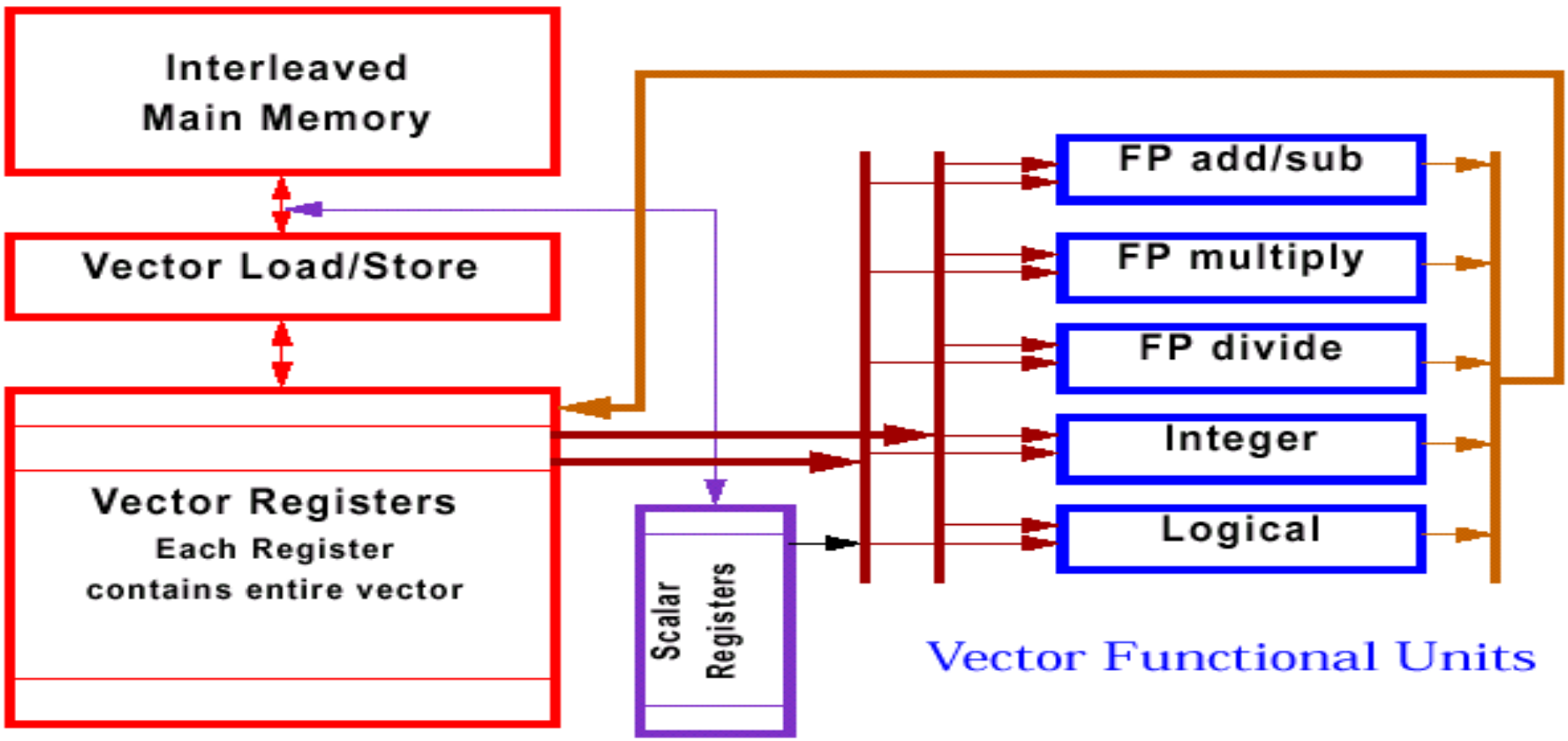
Arquitectura SIMD (2)

- N elementos de proceso iguales
  - Elemento de proceso: ALU + memoria local (para los datos distribuidos, los de las instrucciones vectoriales)
- Una única unidad de control (realmente una CPU) controla los elementos de proceso
- La red de interconexión permite comunicar unos elementos de proceso con otros
- La unidad de control busca las instrucciones en memoria principal y las descodifica. Si la instrucción buscada es
  - Escalar o de bifurcación: se ejecuta en la propia unidad de control
  - Vectorial: la unidad de control la transmite a los elementos de proceso, los cuales actúan simultáneamente
    - Un esquema de enmascaramiento permite habilitar un subconjunto de los elementos de proceso e inhabilitar el resto

Arquitectura SIMD (3)

- Demasiado inflexible
  - No se adecua a un número importante de problemas
- Buenos como procesadores de propósito especial para muchas de las tareas con un alto grado de paralelismo (procesamiento de imagen y de señal)
- Alternativas básicas de diseño:
  - Pocos procesadores complejos: MPP, CM1, Mas Par1
  - Muchos elementos de proceso sencillos: CM5

Arquitectura vectorial



Arquitectura vectorial (2)

- Instrucciones máquina vectoriales y escalares
- Varias unidades funcionales especializadas encauzadas
- Bancos de registros rápidos: vectorial y escalar
- *Vector load-store*: unidad encauzada que carga/almacena vectores desde/en memoria
- Memoria entrelazada (ancho de banda grande)

Arquitectura vectorial (3)

- Cray-1 (año 1976): durante considerable tiempo fue el supercomputador más rápido
- Otros: Cray Y-MP (1988), Convex C-1 (1985), NEC SX/2 (1984)
- Año 1996: con microprocesadores superescalares se obtienen mejores rendimientos que con las máquinas vectoriales

Evolución de las arquitecturas paralelas: introducción

- Fuertes interacciones entre la arquitectura, la tecnología y las aplicaciones (evolucionan conjuntamente)
- El cambio es la norma en arquitectura de computadores
- Un gran cambio a finales de la década de 1990:
  - Todo el espectro de computadores paralelos usa como **componente básico el micropocesador**. Ventajas:
    - Bajo coste (producción a gran escala), alto rendimiento (durante muchos años viene mejorando en torno a un 50% al año)

Evolución de las aplicaciones

- Aplicaciones propias de las arquitecturas paralelas
  - Científicas: física, química, biología...
  - De ingeniería: industria aeronáutica, electrónica, química, farmacéutica...
  - Comerciales, fundamentalmente bases de datos
  - Ocio: realidad virtual, películas de animación...
- Los progresos en los conocimientos respectivos **aumentan las demandas de rendimiento**
- Los aumentos en el rendimiento de los computadores aumentan la cantidad de aplicaciones que se utilizan en ellos

Evolución de la tecnología

- La densidad de los circuitos integrados crece aproximadamente un 50% al año
- El tamaño del dado crece entre un 10 y un 25% al año
- Por tanto en un solo chip pueden integrarse muchos más componentes. Algunas posibilidades:
  - **Integrar en el chip más memoria y soporte de E/S**, además de la UCP: solución utilizada en sistemas empotrados, portables, computadores económicos
  - Colocar **varios procesadores en el chip**: solución utilizada para procesamiento de la señal digital
- Rendimiento de los procesadores crece más deprisa que el de la memoria y el de los discos =>
  - Necesidad de varios niveles de caché (para reducir la latencia)
  - Replicar módulos de memoria y discos (aumenta ancho de banda)

Evolución de las arquitecturas de los microprocesadores

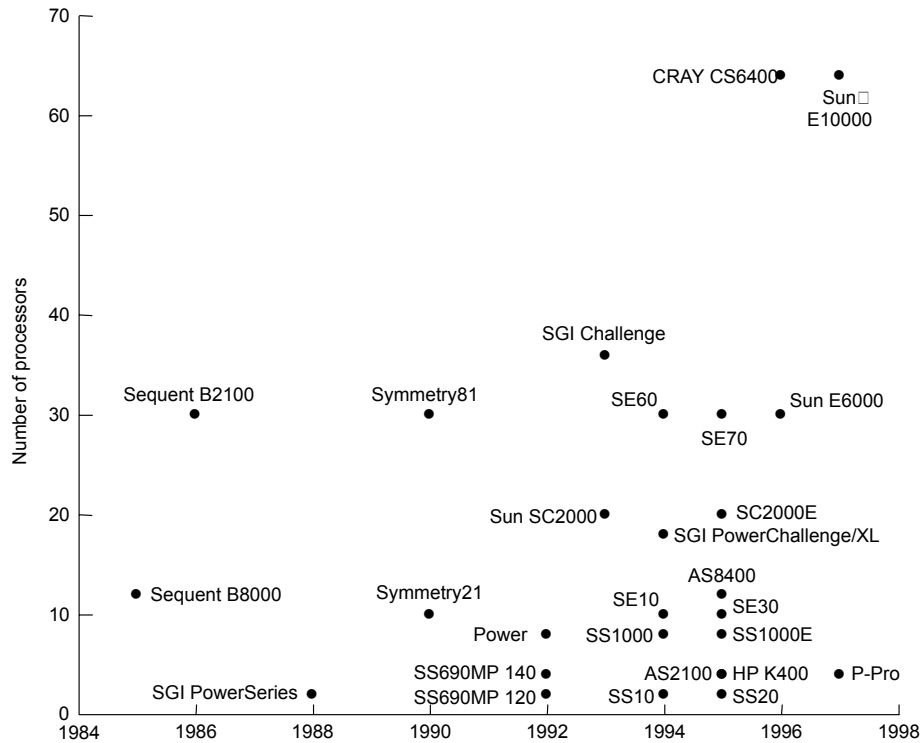
- Hasta mediados década de 1980: aumento del paralelismo a nivel de bit (mediante el aumento del ancho de palabra)
- Desde mediados de los 80: avances en el paralelismo a nivel de instrucción:
  - Microprocesadores encauzados
  - Microprocesadores superescalares cada vez con más paralelismo (dado que cada vez puede integrarse más hardware en un solo chip) pero...
    - Un paralelismo superescalar mayor que 4 incrementa poco el rendimiento
- Cabe esperar que los siguientes esfuerzos se dediquen a conseguir un buen paralelismo a nivel de *threads*

Evolución de los sistemas computadores

- Desde mediados de la década de 1980 se generalizan los computadores basados en varios microprocesadores comerciales compartiendo una memoria común
  - Desde entonces los microprocesadores comerciales tienen el hardware necesario para poder formar parte de un multiprocesador
- Principio de los 90: grandes avances en el rendimiento del bus de memoria compartida debido a: bus múltiple, transmisión más rápida de señales, rutas de datos más anchas, protocolos de comunicación encauzados
  - Estos avances eran muy necesarios porque al mejorar el rendimiento de los microprocesadores el bus se satura con menos procesadores

Evolución de los sistemas computadores (cont)

- Cada vez mayor expansión de los MIMD de memoria compartida basados en bus
  - Computadores de sobremesa y pequeños servidores soportan entre dos y cuatro procesadores
  - Servidores grandes soportan decenas de procesadores
  - Sistemas comerciales grandes, se acercan a cien
- La tendencia anterior conduce a utilizar chips con varios procesadores



Evolución de los supercomputadores

- La fórmula 1 de los computadores
- Tradicionalmente dominada por los computadores vectoriales
- Hoy día dominada por sistemas basados en microprocesadores (v. fig.)

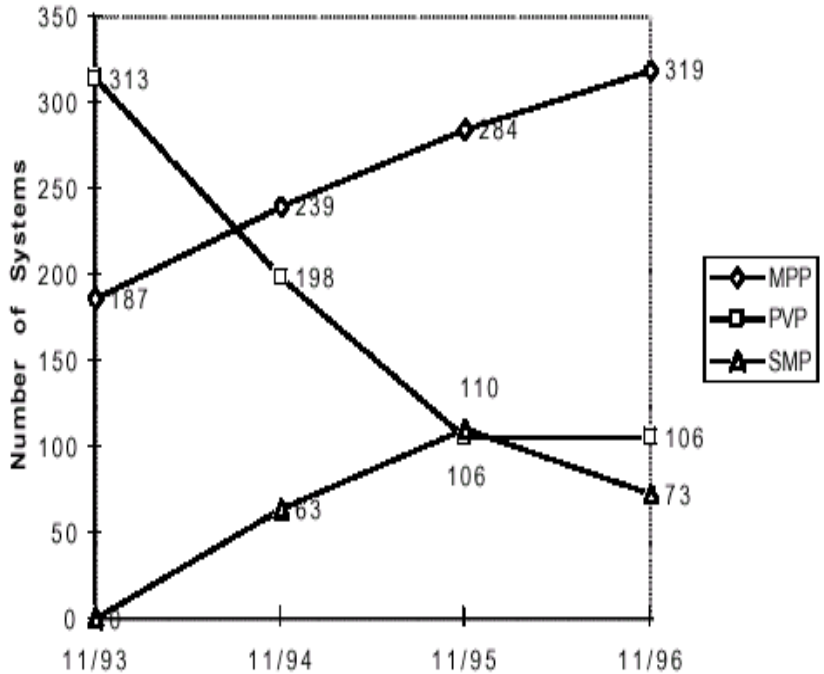


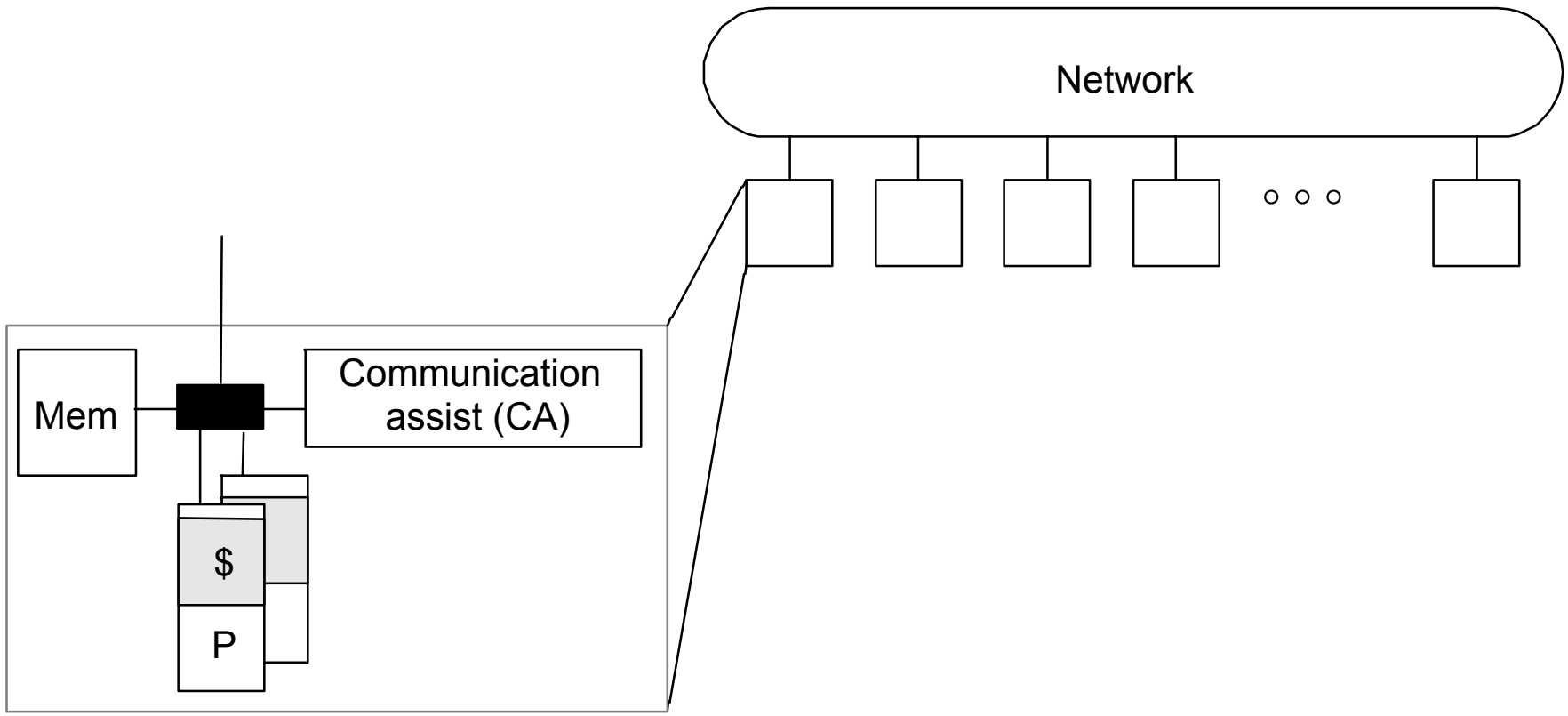
Figure 1-12 Type of systems used in 500 fastest computer systems in the world.

Parallel vector processors (PVPs) have given way to microprocessor-based Massively Parallel Processors (MPPs) and bus-based symmetric shared-memory multiprocessors (SMPs) at the high-end of computing.

Convergencia de las arquitecturas paralelas

- La evolución del hardware y del software ha conducido a una convergencia de las máquinas paralelas hacia una estructura común:
  - **Varios nodos que se comunican mediante una red escalable de propósito general y alto rendimiento**
  - **Cada nodo** incluye
    - **Uno o más procesadores (integran caché [$])**
    - **Memoria**
    - Un **controlador para las operaciones a través de la red (CA)**
- Desde esta perspectiva, la cuestión básica del diseño es
  - Qué funcionalidad debe tener el CA (*comunication assist*)
  - Cómo debe ser la interfaz entre el CA y el/los procesadores, el sistema de memoria y la red [caja negra en la fig. sgte.]

Organización genérica de un MIMD escalable



Bibliografía

- Sima, D., T. Fountain and P. Kacsuk (1997): *Advanced Computer Architectures, A Design Space Approach.* Addison-Wesley.
- Culler, D. E. and J. P. Singh (1999): *Parallel Computer Architecture, A Software/Hardware Approach.* Morgan Kauffman. San Francisco.
- Patterson D. A. and J. L. Hennessy (1995): *Organización y diseño de computadores, La interfaz hardware/software.* McGraw-Hill. Madrid.