UNIVERSIDADE DA CORUÑA

GRUPO DE ARQUITECTURA DE COMPUTADORES

INTRODUCCIÓN AL PROCESAMIENTO PARALELO

ARQUITECTURA E INGENIERÍA DE COMPUTADORES

TEMA1

ÍNDICE

Procesamiento paralelo

- Más potencia de cálculo y más fiabilidad
  - Aumentar la velocidad de trabajo de los procesadores y otros elementos
    - Depende de la tecnología, organización de los componentes del computador, y de la arquitectura del procesador.
    - Puede tener alto coste de diseño y de fabricación.
  - Conectar varios procesadores y coordinar su funcionamiento
    - **Computadores paralelos**
    - Distribución de computaciones entre sistemas remotos: **sistemas distribuidos**

¿Cómo ser más rápidos?

- Tres métodos (Pfister, 1998)
  - Trabajar más duro
  - Trabajar de forma más inteligente
  - Conseguir ayuda
- Equivalente en computación
  - Usar hardware más rápido (incrementar la frecuencia de reloj del procesador…)
  - Optimizar los algoritmos
  - Usar múltiples recursos (varios procesadores, varias unidades de procesamiento de instrucciones, …)para resolver una tarea en particular --> **paralelismo**

ÍNDICE (I)

Condiciones de paralelismo

- Concepto de **dependencia**:
  - La posibilidad de ejecutar diversos segmentos de programa en paralelo requiere la independencia de éstos.
  - Tipos de dependencias:
    - Dependencias **de datos**
    - Dependencias **de control**
    - Dependencias **de recursos**

Nota: por simplicidad, ejemplos de relaciones de dependencia entre instrucciones

Dependencias de datos

- **Dependencia de flujo** (o verdadera):
  - La instrucción S2 posee una dependencia de flujo respecto de S1, si una variable generada por S1 es utilizada como operando de S2.
  - Dependencia "inherente" al código.
- **Antidependencia**:
  - S2 es antidependiente de S1, si S2 modifica una variable utilizada como operando por S1.
- **Dependencia de salida**:
  - S1 y S2 poseen una dependencia de salida entre ambas si las dos generan la misma variable.

Dependencias de control

- Aparecen en situaciones en las que el orden de ejecución de las instrucciones no puede ser determinado antes del tiempo de ejecución.
- Ejemplo característico: diferentes caminos tomados después de una instrucción de **salto condicional** pueden introducir o eliminar dependencias de datos entre instrucciones.

Dependencias de recursos

- Conflictos de utilización de recursos **hardware** compartidos por parte de los eventos paralelos.

- Los recursos compartidos pueden ser: unidades funcionales, registros del procesador, direcciones de memoria,…

Condiciones de Bernstein (I)

- Condiciones para la detección de paralelismo entre dos "segmentos" de código:
  - Son las condiciones que deben cumplir dos procesos para poder ser ejecutados en paralelo
  - Entendemos por proceso una entidad software correspondiente a la abstracción de un fragmento de programa
  - Estas condiciones sólo son suficientes para la detección de dependencias de datos, no para las dependencias de control y dependencias de recursos

Condiciones de Bernstein (II)

- **Conjunto de entrada** Ii de un proceso Pi: conjunto de todas las variables de entrada necesitadas para ejecutar un proceso Pi
- **Conjunto de salida** Oi de un proceso Pi: conjunto de todas las variables de salida generadas después de la ejecución de Pi.

- Dados dos procesos P1 y P2 con conjuntos de entrada I1 e I2, y conjuntos de salida O1 y O2 respectivamente, ambos pueden ejecutarse en paralelo (P1//P2), si verifican todas las condiciones siguientes:
  - I1 intersec O2 = conjunto_vacío (WAR-antidependencia)
  - I2 intersec O1 = conjunto_vacío (RAW-dep_flujo)
  - O1 intersec O2 = conjunto_vacío (WAW-dep_salida)
- En general, un conjunto de procesos P1, P2,…,Pk pueden ejecutarse en paralelo P1//P2//…//Pk si y sólo si:
  - Pi//Pj para todo $i \neq j$

Paralelismo hardware y software

- **Paralelismo hardware**: posibilidades que ofrece el hardware de un sistema computador para ejecutar "segmentos" de código en paralelo.
  - Paralelismo temporal (segmentación)
  - Paralelismo espacial (replicación)
    - Paralelismo funcional.

- **Paralelismo software**: posibilidades que ofrece un código para que algunas de sus partes se ejecuten en paralelo.
  - Determinado por las dependencias de datos y de control del código.

ÍNDICE (I)

- Procesamiento paralelo
- Condiciones de paralelismo
  - Concepto de dependencia
  - Condiciones de Bernstein
  - Paralelismo hardware y software
- **Niveles de paralelismo**
- Importancia del procesamiento paralelo
- Clasificaciones de arquitecturas paralelas
  - Taxonomía de Flynn
  - Organización del sistema memoria: clasificación tradicional de sistemas MIMD.
- Medidas de rendimiento

Niveles de paralelismo (I)

- **Granularidad**: magnitud de la tarea (número de operaciones) candidata a ser ejecutada en paralelo.
  - **Granularidad hardware**: tamaño del tipico del hardware replicado en el sistema paralelo.
  - **Granularidad sofware**: tamaño tipico del software que es ejecutado en paralelo.
  - Denominaciones típicas: **grano fino, grano medio, grano grueso**.

Niveles de paralelismo (II)

- Niveles de **paralelismo software** en una aplicación (I)
  - Nivel de programas: grano grueso
    - Normalmente poca dependencia entre ellos
  - Nivel de funciones: grano medio
    - Es paralelismo de tareas
  - Nivel de bucles: grano medio-fino
    - Se pueden ejecutar en paralelo las iteraciones
    - Es paralelismo de datos
  - Nivel de operaciones: grano fino
    - Operaciones independientes se pueden ejecutar en paralelo

Niveles de paralelismo (III)

- **Paralelismo software** a nivel de ejecución(II)
  - Nivel de instrucciones
  - Nivel de proceso
    - Cada proceso en ejecución tiene su propia asignación de memoria.
  - Nivel de hebra (thread)
    - Un proceso puede componerse de varias hebras
    - Cada hebra tiene su propia pila, contenido de registros (entre ellos el IP) pero comparte código, variables locales y otros recursos (archivos abiertos, cache,…) con las hebras del mismo proceso.
    - La creación, destrucción, comunicación, sincronización y conmutación de hebras son más rápidas que la de procesos (procesos ligeros)

NOTA: actualmente, en algunos textos y documentación de fabricantes de computadores se habla de procesos y threads indistintamente sin exigir a los threads la compartición de recursos.

Niveles de paralelismo (III)

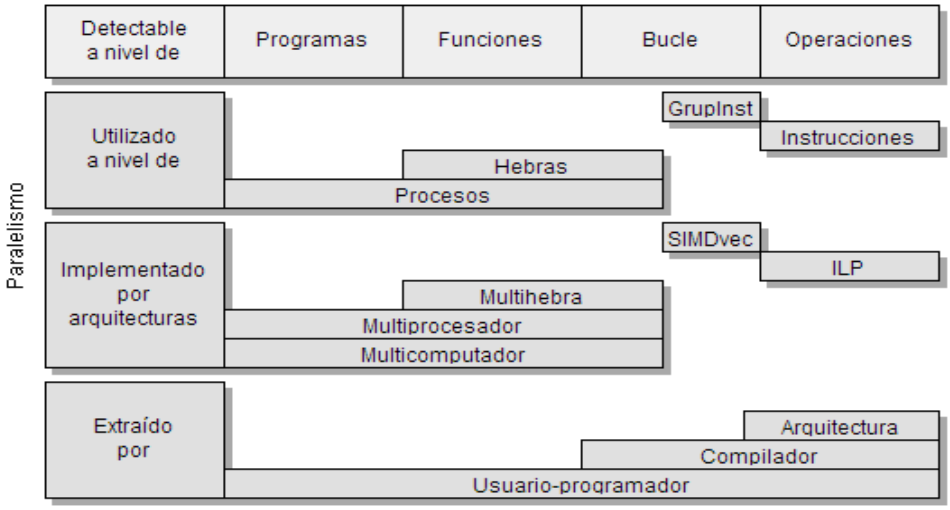
- Paralelismo implementado a **nivel hardware** en sistemas computador
  - Multiprocesadores/Multicomputadores: grano grueso, grano medio.
  - Procesadores Multihebra (*Multithreading*): grano medio.
  - Paralelismo a nivel de instrucción: grano fino.
    - Segmentación
    - Replicación de unidades funcionales
    - Replicación de unidades de procesamiento de instrucciones.

Paralelismo y arquitecturas (I)

- Relación entre paralelismo a nivel software y arquitecturas/procesadores (nivel hardware) paralelos
  - Paralelismo entre programas: a nivel de procesos
    - Multiprocesador/Multicomputador
  - Paralelismo entre funciones: a nivel de procesos o de hebras (threads)
    - Multiprocesador/Multicomputador, procesadores multihebra, procesadores multinúcleo (multicores)
  - Paralelismo de bucles: a nivel de procesos o de hebras
    - Multiprocesador/Multicomputador, procesadores multihebra, procesadores multinúcleo (multicores)
    - Instrucciones vectoriales: extensiones multimedia, procesadores vectoriales, GPUs
  - Paralelismo de operaciones/instrucciones:
    - Segmentación
    - Procesadores superescaleres y VLIW
    - Paralelimo funcional: diferentes unidades funcionales especializadas o replicadas.
    - Instrucciones vectoriales: extensiones multimedia, procesadores vectoriales, GPUs

Paralelismo y arquitecturas (II)

- Utilización, implementación y detección del paralelismo



ILP: Instruction Level Parallelism, paralelismo a nivel de instrucción

ÍNDICE (I)

- Procesamiento paralelo
- Condiciones de paralelismo
  - Concepto de dependencia
  - Condiciones de Bernstein
  - Paralelismo hardware y software
- Niveles de paralelismo
- **Importancia del procesamiento paralelo**
- Clasificaciones de arquitecturas paralelas
  - Taxonomía de Flynn
  - Organización del sistema memoria: clasificación tradicional de sistemas MIMD.
- Medidas de rendimiento

Motivación del procesamiento paralelo

- Motivación al estudio de los sistemas con uno o varios niveles de paralelismo:
  - **Aplicaciones** que requieren estas arquitecturas
  - **Accesibilidad** al hardware y al software
  - Necesidad de estudiar diferentes **facetas de diseño** de sistemas computador que incluyan algún tipo de paralelismo

Motivación: aplicaciones

- Aplicaciones que requieren computadores paralelos
  - Aplicaciones que requieren una potencia mayor que las de un sistema uniprocesador: tiempo y calidad aceptables
    - Grandes sistemas de bases de datos, servidores de aplicaciones o de Internet
      - Ejemplo: Google
        - Asigna una petición a múltiples procesadores
        - Cluster de 15000 computadores convencionales

Motivación: aplicaciones

- Grandes aplicaciones científicas o de ingeniería
  - Respuesta en un tiempo o calidad aceptable
  - Predicción meteorológica
  - Modelado de seísmos, océanos
  - Bioinformática: genoma humano
  - Química computacional
  - …
  - En general, aplicaciones *Grand Challenge*
- Tratamiento de imágenes y gráficos
  - Rendering de películas:
    - Pixar utilizaron RenderMan en 2003: Buscando a Nemo, Matrix Reloaded, Terminator 3, …

Motivación: aplicaciones

- Aplicaciones que requieren **tolerancia de fallos**
  - Redundancia de componentes
    - Permiten que la aplicación siga ejecutándose aunque falle algún componente
    - Aplicaciones en las que un fallo suponga riesgo para la vida o cuantiosas pérdidas económicas
      - Centrales nucleares, proyectos espaciales
  - En aplicaciones que requieren **alta disponibilidad**
    - Aplicaciones de empresas con grandes bases de datos
      - Bancos, grandes empresas

Motivación: aplicaciones

- Aplicaciones embebidas *(embedded, sistemas empotrados)*
  - Aplicaciones que engloban señales multimedia (audio, vídeo, gráficos, documentos)
    - Requieren buenas prestaciones (tiempo real), bajo coste, bajos consumos de potencia, tamaño reducido
  - Usan múltiples procesadores de bajo coste, bajo consumo y tamaño reducido: DSP, procesadores media, microcontroladores, procesadores de propósito general, con capacidad para conectarlos
  - Cámaras digitales, PDA, DVD, consolas de videojuegos,…

Motivación: aplicaciones

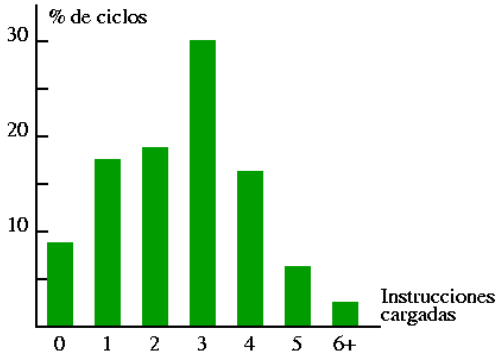
- Aplicaciones que requieren altas prestaciones
  - **HPC** *(High Performance Computing)*
    - Sometidas a restricciones de tiempo
      - Predicción meteorológica
  - **HTC** *(High Throughput Computing)*
    - No sometidas a restricciones de tiempo
    - Aprovechamiento a largo plazo
- Computación Grid
  - Resolver aplicaciones que deben utilizar múltiples recursos situados en diferentes lugares geográficos y bajo diferentes dominios de administración.

Motivación: accesibilidad

- Accesibilidad al hardware y software de computadores paralelos
  - Varios niveles según coste y prestaciones
    - PCs y estaciones de trabajo (menos de 8.000 €)
    - Servidores básicos (menos de 20.000 €)
    - Servidores gama media (entre 20.000 y 400.000 €)
    - Grandes servidores, *mainframe servers*, (desde 400.000 €)
    - **Supercomputadores**: computadores que ofrecen las mayores prestaciones (mayores precios)
      - Lista top500 (http://www.top500.org) desde 1993
    - **Cluster**: mejor relación prestaciones/coste

Motivación: accesibilidad

- Futuro de los computadores paralelos
  - Garantizado por las limitaciones de los uniprocesadores
    - ILP: máximo de 4 instrucciones por flujo secuencial
    - Procesadores multihebra (Power 5, Xeon, Pentium 4 hypertheading)
    - Replicar procesadores en un chip: procesadores multicore en los ordenadores personales (ahora mismo!!)



  - Posibilidad de expansión
  - Uso de SO independientes del fabricante (Linux, etc.)
  - Existencia de software que permite la portabilidad entre arquitecturas
    - MPI, OpenMP

Motivación: facetas de diseño

- Estudio de diferentes facetas de diseño de sistemas paralelos:
  - Arquitectura, estructura y organización tanto a nivel de un procesador como de múltiples procesadores:
    - Elección de la más conveniente según el trabajo a realizar (tipo de aplicación)
      - Obtener la mejor relación prestaciones/coste
        - Pocas comunicaciones: red de bajo coste…
        - Paralelismo grano fino: memoria compartida…
    - Elección del software
      - Cluster de multiprocesadores de mem. compartida: MPI + OpenMP
    - Extracción del paralelismo
      - A nivel de proceso, de hebra, etc.
    - Optimización del rendimiento del sistema

ÍNDICE (I)

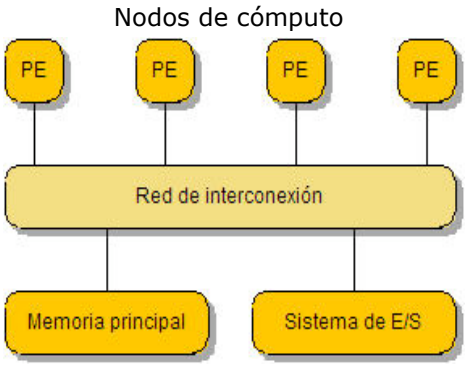
- Procesamiento paralelo
- Condiciones de paralelismo
  - Concepto de dependencia
  - Condiciones de Bernstein
  - Paralelismo hardware y software
- Niveles de paralelismo
- Importancia del procesamiento paralelo
- **Clasificaciones de arquitecturas paralelas**
  - Taxonomía de Flynn
  - Organización del sistema memoria: clasificación tradicional de sistemas MIMD.
- Medidas de rendimiento.

Clasificaciones de arquitecturas paralelas

- Diferentes criterios de clasificación
  - Taxonomía de Flynn
  - Organización del sistema de memoria
  - Escalabilidad
  - Utilización de componentes disponibles comercialmente
  - Relación prestaciones/coste
  - Niveles de paralelismo que aprovechan
  - …

Espacio de diseño

- Nodo de cómputo
  - Temas 2 y 3 de la asignatura
- Sistema de memoria
  - Temas 4 y 5 de la asignatura
- Sistema de comunicación
  - Memoria compartida
    - Temas 4 y 5 de la asignatura
  - Red de interconexión
    - Tema 6 de la asignatura
- Sistema de E/S
  - No se trata en esta asignatura



ÍNDICE (I)

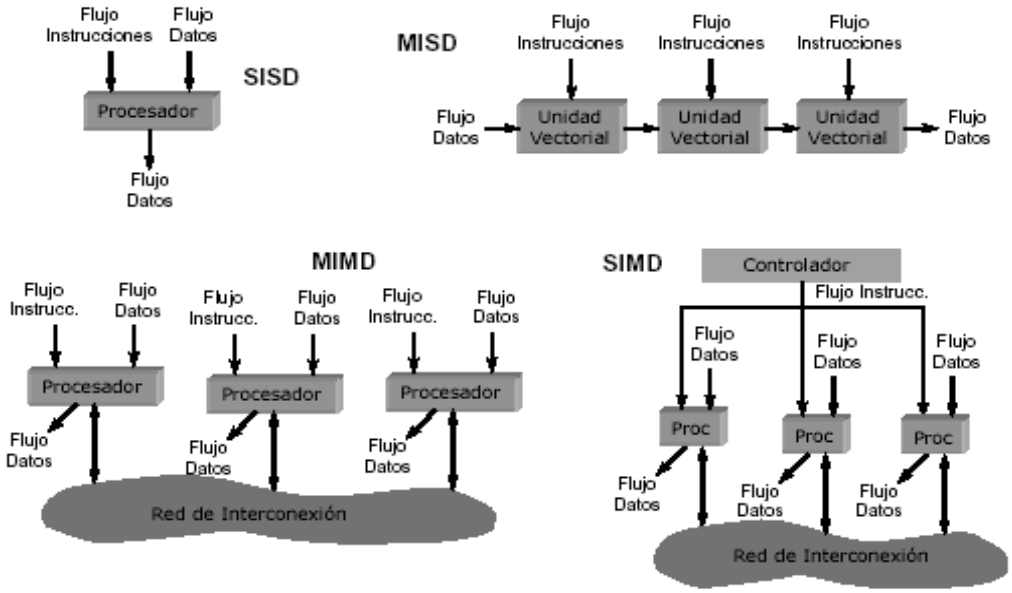
- Procesamiento paralelo
- Condiciones de paralelismo
  - Concepto de dependencia
  - Condiciones de Bernstein
  - Paralelismo hardware y software
- Niveles de paralelismo
- Importancia del procesamiento paralelo
- Clasificaciones de arquitecturas paralelas
  - **Taxonomía de Flynn**
  - Organización del sistema memoria: clasificación tradicional de sistemas MIMD.
- Medidas de rendimiento

Taxonomía de Flynn (1966)

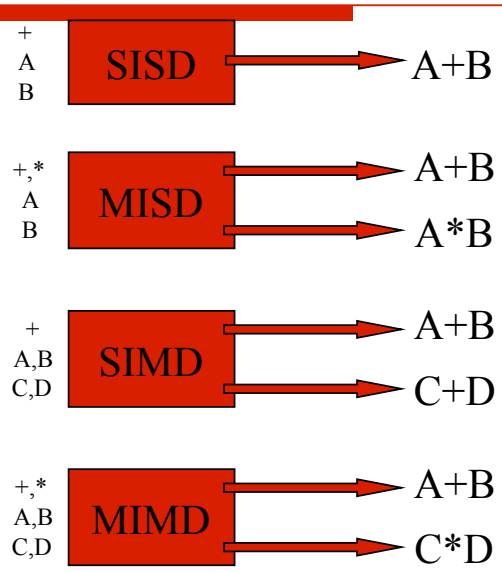
- Basada en el flujo se instrucciones y datos

- **SISD**: *Single Instruction Single Data*
  - Uniprocesadores
- **MISD**: *Multiple Instruction Single Data*
  - Múltiples procesadores sobre un único flujo de datos
- **SIMD**: *Single Instruction Multiple Data*
  - Modelo de programación sencillo (SPMD)
  - Ejecución **síncrona** de la misma instrucción sobre datos diferentes
- **MIMD**: *Multiple Instruction Multiple Data*
  - Ejecución **asíncrona** de múltiples "segmentos" de código diferentes sobre datos diferentes

Taxonomía de Flynn

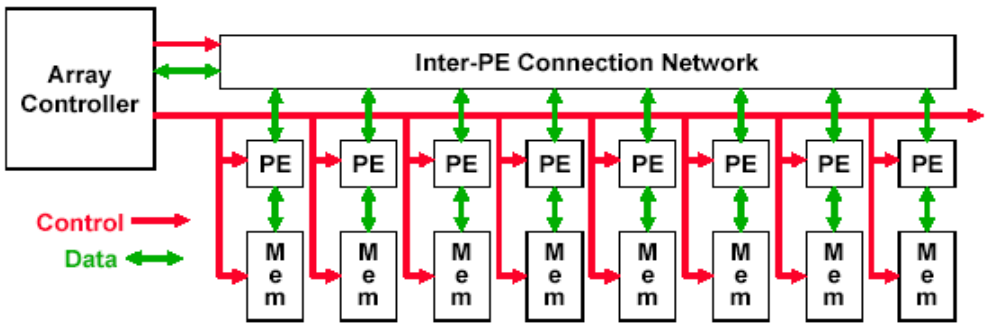


Flynn: potencial de cada tipo



Arquitectura SIMD

- Controlador central radia instrucciones a todos los elementos de procesamiento (PEs)



  - Sólo requiere un único controlador para todos los PEs
  - Sólo una copia del programa
  - Computaciones totalmente sincronizadas

Arquitecturas SIMD

- **Computadores matriciales** *(array processors)*:
  - Illiac IV (1972)
  - ICL DAP (Distributed Array Processor) (1980)
  - Thinking Machines Connection Machine
    - CM-1 (1985), CM-2
  - Maspar
    - MP-1(1989), MP-2
- **Supercomputadores vectoriales** *(vector processors)*
    - TI ASC (1971), CDC Star- 100 (1973), Cray-1(1976)
    - En la actualidad: multiprocesadores vectoriales (Earth Simulator, Cray X1)

Nota: sólo para algunos autores los procesadores vectoriales son clasificados como SIMD.

SIMD en la actualidad

- **Procesamiento en modo SIMD**:
  - Extensiones multimedia de los repertorios de instrucciones de los microprocesadores actuales (instrucciones vectoriales): SSE de Intel, Altivec para el PowerPC (IBM), 3DNow para AMD.
  - GPUs (Graphic Processing Unit): tarjetas gráficas como GeForce 8 Series, Quadro FX 5600/4600…
  - Cell processor: es el procesador de la Playstation3 -> dentro de cada unidad unidad SPE (tiene 8 en el mismo procesador) utiliza instrucciones vectoriales.
    - Nota: El conjunto de 8 SPEs + la unidad principal PPE funcionan en modo asíncrono (MIMD).
  - Un sistema MIMD puede implementar una ejecución paralela en modo SIMD (SPMD).

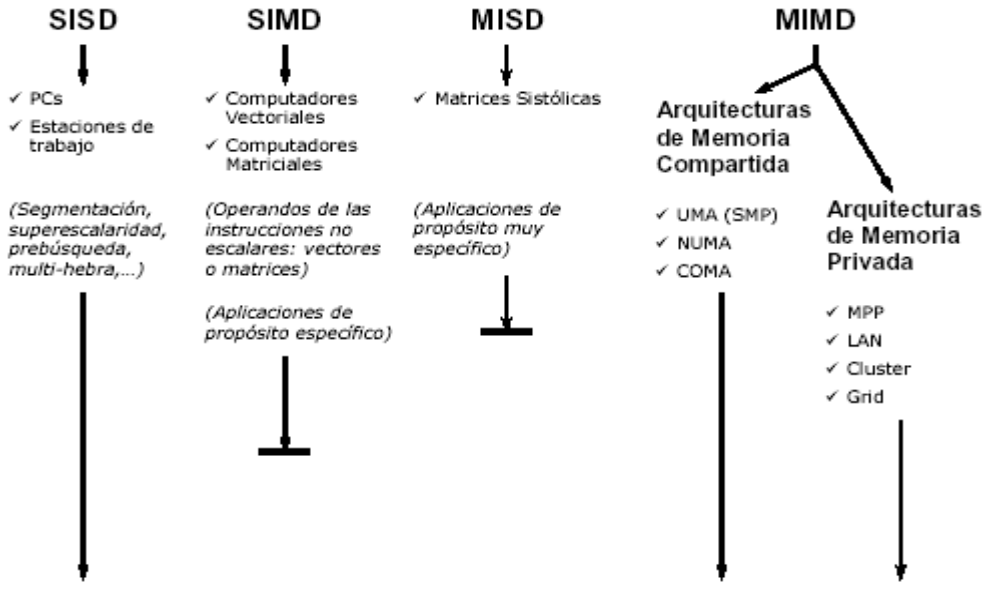
Arquitecturas MISD

- No existen modelos comerciales con este tipo de arquitectura
  - Razón: pocas aplicaciones significativas pueden requerir este modelo de procesamiento.
  - Algunos autores: arquitecturas sistólicas.

Arquitecturas MIMD

- Arquitecturas paralelas más extendidas:
  - Multiprocesadores de memoria compartida
  - Multiprocesadores de memoria distribuida (multicomputadores)
  - Multicores: multiprocesadores en un chip.
  - Otros nombres: se verán en temas posteriores.

Taxonomía de Flynn: más actual



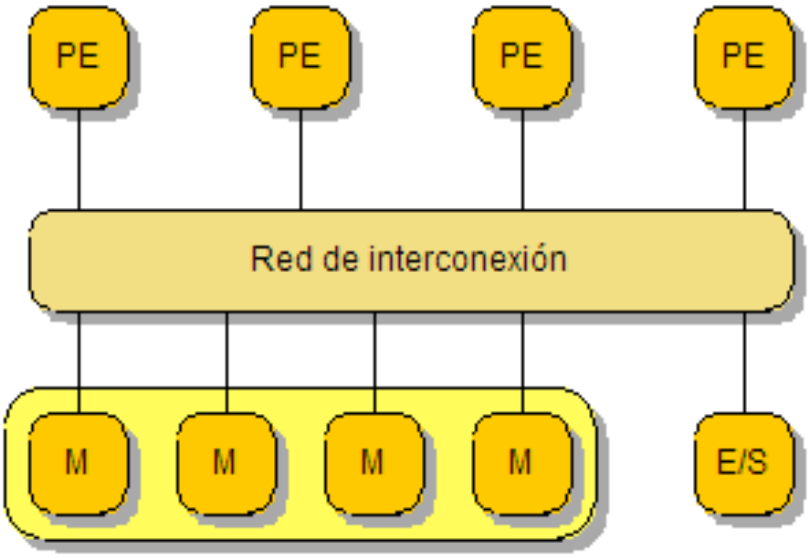
ÍNDICE (I)

- Procesamiento paralelo
- Condiciones de paralelismo
  - Concepto de dependencia
  - Condiciones de Bernstein
  - Paralelismo hardware y software
- Niveles de paralelismo
- Importancia del procesamiento paralelo
- Clasificaciones de arquitecturas paralelas
  - Taxonomía de Flynn
  - **Organización del sistema memoria: clasificación tradicional de sistemas MIMD.**
- Medidas de rendimiento

Organización del sistema memoria: clasificación tradicional de sistemas MIMD.

- Sistemas con memoria compartida, *(Shared memory),* **multiprocesadores**
  - Todos los procesadores comparten el mismo espacio de direcciones
  - El programador no necesita conocer la ubicación de los datos
- Sistemas con memoria distribuida, *(Distributed memory),* **multicomputadores**
  - Cada procesador tiene su espacio de direcciones particular
  - El programador necesita conocer dónde están los datos

Multiprocesadores

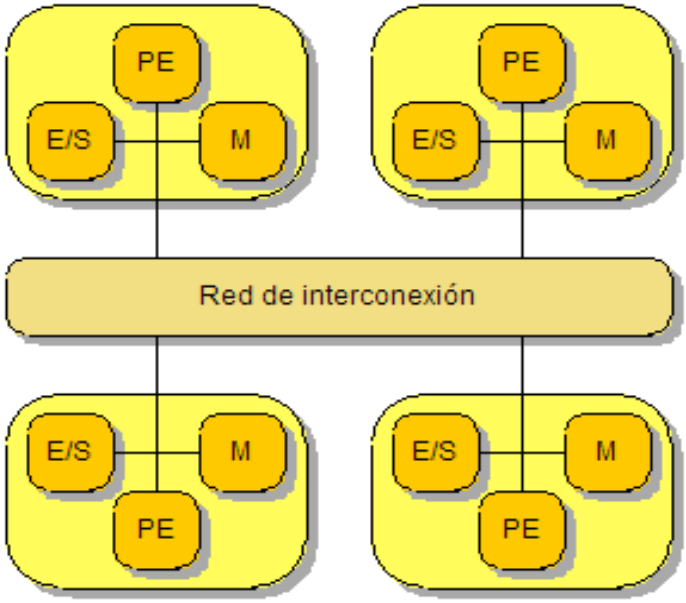


- Red de interconexión
  - Bus
  - Barras cruzadas
  - Multietapa bidireccional
  - …

Multiprocesadores

- La memoria separada de los procesadores por la red de interconexión
- Tiempo de acceso a memoria es igual independiente de la posición accedida: SMP *(Symmetric MultiProcessor)* **multiprocesadores simétricos**
- Acceso a memoria vía la red de interconexión
- Comunicación mediante operaciones de lectura/escritura en memoria
- La E/S también centralizada

Multicomputadores



- Red de interconexión
  - Barras cruzadas
  - Multietapa bidireccional
  - Árbol grueso
  - Toroide
  - …

Multicomputadores

- Cada procesador tiene un módulo de memoria "cerca", con su espacio de direcciones, al que accede directamente
- Transferencia de datos compartidos: mensaje a través de la red de interconexión
  - Tamaño de los mensajes depende del programador
- Ejecución de aplicaciones:
  - Implican comunicaciones ->transferencia (copia) de datos
  - Implican sincronizaciones: normalmente se realizan en las comunicaciones

ÍNDICE (I)

- Procesamiento paralelo
- Condiciones de paralelismo
  - Concepto de dependencia
  - Condiciones de Bernstein
  - Paralelismo hardware y software
- Niveles de paralelismo
- Importancia del procesamiento paralelo
- Clasificaciones de arquitecturas paralelas
  - Taxonomía de Flynn
  - Organización del sistema memoria: clasificación tradicional de sistemas MIMD.
- **Medidas de rendimiento.**

MEDIDAS DE RENDIMIENTO

- Dos conceptos clave: **tiempo de ejecución, y ancho de banda**

Avión	Wa a París	Velocidad	Pasajeros	Throughput (p.km/h)
Boeing 747	6.5 horas	970 km/h	470	455900
Concorde	3 horas	2160 km /h	132	285120

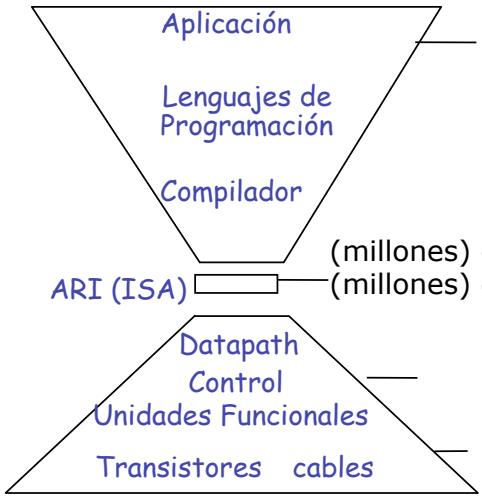
MEDIDAS DE RENDIMIENTO

- ✓ **Tiempo de Ejecución (TEj)** : Tiempo que tarda en completarse una tarea
  - ✓Otros nombres: Tiempo de respuesta, latencia
- ✓ **Rendimiento o ancho de banda** ("Performance", Throughput) : tareas por hora, día ,…
- ✓ "X es n veces más rápido que Y" significa

$$\frac{TEj(Y)}{TEj(X)} = \frac{Performance(X)}{Performance(Y)} = n$$

- ✓ Reducir el TEj incrementa el rendimiento

MEDIDAS DE RENDIMIENTO



Respuestas por mes, hora, segundo
Operaciones por segundo TPC

(millones) de Instrucciones por segundo: **MIPS**
(millones) de (FP) operaciones por segundo: **MFLOP/s**

Megabytes por segundo

Ciclos por segundo (frecuencia de reloj)

La única medida fiable es el tiempo de ejecución programas reales
Dos aspectos: **Rendimiento del computador**, **Rendimiento del procesador**

MEDIDAS DE RENDIMIENTO

- **Rendimiento del procesador**

$$T_{CPU} = N * CPI * t$$

  - ✓ N: nº de instrucciones (Compiladores y repertorio de instrucciones)
  - ✓ CPI: (repertorio de instrucciones, implementación, paralelismo)
  - ✓ t: período de reloj (implementación, tecnología)

- **Ciclos medios por instrucción (CPI)**

CPI = (T_{CPU} * Frecuencia de reloj) / Numero de Instrucciones
= Ciclos / Numero de Instrucciones

TCPU = Tiempo de ciclo * $\sum_{j=1}^{n}$ (CPI j * I j) (n = tipos de instr., Ij = nº instr. tipo j ejecutadas)

CPI = $\sum_{j=1}^{n}$ CPI j * F j (donde Fj es la frecuencia de aparición de la instrucción tipo J)

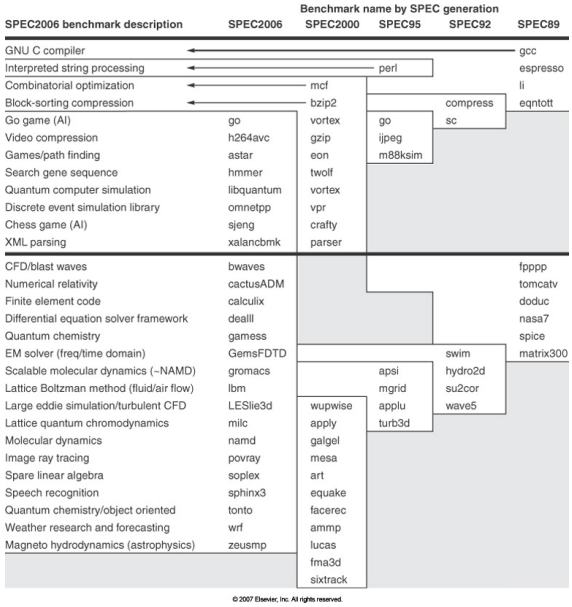
Ejemplo: ALU 1 ciclo(50%), Ld 2 ciclos(20%), St 2 ciclos(10%), saltos 2 ciclos(20%)
CPI: ALU 0.5, Ld 0.4, St 0.2, salto 0.4 TOTAL CPI = 1.5

CONCLUSIÓN: INVERTIR RECURSOS DONDE SE GASTA EL TIEMPO

MEDIDAS DE RENDIMIENTO

- **Rendimiento global del computador: Benchmarks**
  - ✓ La única forma fiable es ejecutando distintos programas reales.
    - ✓ Programas "de juguete": 10~100 líneas de código con resultado conocido. *Ej:: Criba de Erastótenes, Puzzle, Quicksort*
    - ✓ Programas de prueba (*benchmarks*) sintéticos: simulan la frecuencia de operaciones y operandos de un abanico de programas reales. *Ej:: Whetstone, Dhrystone*
  - ✓ Programas reales típicos con cargas de trabajo fijas (actualmente la medida más aceptada)
    - ✓ **SPEC89**: 10 programas proporcionando un único valor.
    - ✓ **SPEC92**: 6 programas enteros (SPECint92) y 14 en punto flotante (SPECfp92).Sin limites en opciones de compilación
    - ✓ **SPEC95**: 8 programas enteros (SPECint95) y 10 en punto flotante (SPECfp95). Dos opciones en compilación: la mejor para cada programa y la misma en todos (base)
    - ✓ **SPEC2000** 12 programas enteros y 14 en punto flotante. Dos opciones de compilación (la mejor: spec--, la misma spec--_base

MEDIDAS DE RENDIMIENTO



MEDIDAS DE RENDIMIENTO

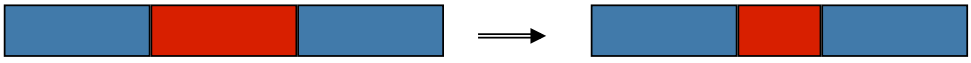
- **Rendimiento global del computador: Benchmarks**
  - ✓ Otros
    - ✓ HPC: LINPACK, SPEChpc96, Nas Parallel Benchmark
    - ✓ Servidores: SPECweb, SPECSFS(File servers), TPC-C
    - ✓ Graficos: SPECviewperf(OpenGL), SPECapc(aplicaciones 3D)
    - ✓ Winbench, EEMBC

LEY DE AMDHAL

- **Un principio básico**: Hacer rápidas las funciones frecuentes --> Gastar recursos donde se gasta el tiempo.
- **Ley de Amdahl**: Permite caracterizar este principio

Permite la evaluación del speedup que se obtendrá al aplicar una cierta mejora, M, que permite ejecutar una parte del código **x** veces más rápido.

$$\textbf{Speedup(E)} = \frac{\text{TEj sin M}}{\text{TEj con M}} = \frac{\text{Performance con M}}{\text{Performance sin M}}$$



Si la mejora sólo acelera la ejecución de un fracción **F** de la tarea, el tiempo de ejecución del resto permanece sin modificación. Por tanto es muy importante el porcentaje de la tarea que es acelerada.

$$\mathbf{F} = \frac{t_B}{t_A + t_B + t_C}$$

LEY DE AMDHAL

$$TEj_{nuevo} = TEj_{antiguo} \times [(1 - Fraccion_{mejora}) + Fraccion_{mejora} / X]$$

$$\textbf{Speedup} = Tej_{antiguo}/Tej_{nuevo} = 1 / [(1 - Fraccion_{mejora}) + Fraccion_{mejora} / X]$$

Ejemplo 1:. El 10% del tiempo de ejecución de mi programa es consumido por operaciones en PF. Se mejora la implementación de la operaciones PF reduciendo su tiempo a la mitad

$TEj_{nuevo} = TEj_{antiguo} \times (0.9 + 0.1 / \mathbf{2}) = 0.95 \times TEj_{antiguo}$ Speedup = 1 / 0.95 = 1.053

Mejora de sólo un 5.3%

Ejemplo 2: Para mejorar la velocidad de una aplicación, se ejecuta el 90% del trabajo sobre 100 procesadores en paralelo. El 10% restante no admite la ejecución en paralelo.

$TEj_{nuevo} = TEj_{antiguo} \times (0.1 + 0.9 / \mathbf{100}) = 0.109 \times TEj_{antiguo}$ Speedup = 1 / 0.109 = 9.17

El uso de 100 procesadores sólo multiplica la velocidad por 9.17

Bibliografía

- *Arquitectura de Computadores*, Julio Ortega, Mancia Anguita y Alberto Prieto. Thompson, 2005.
- *Computer Architecture: A Quantitative Approach.* J.L. Hennessy, D.A. Patterson. Morgan Kaufmann, 2007.
- *Organización y Arquitectura de Computadores* (séptima edición). W. Stallings. Prentice Hall, 2007.

- **Agradecimientos**: Prof. José C. Cabaleiro (USC), Prof. Francisco Tirado (UCM).