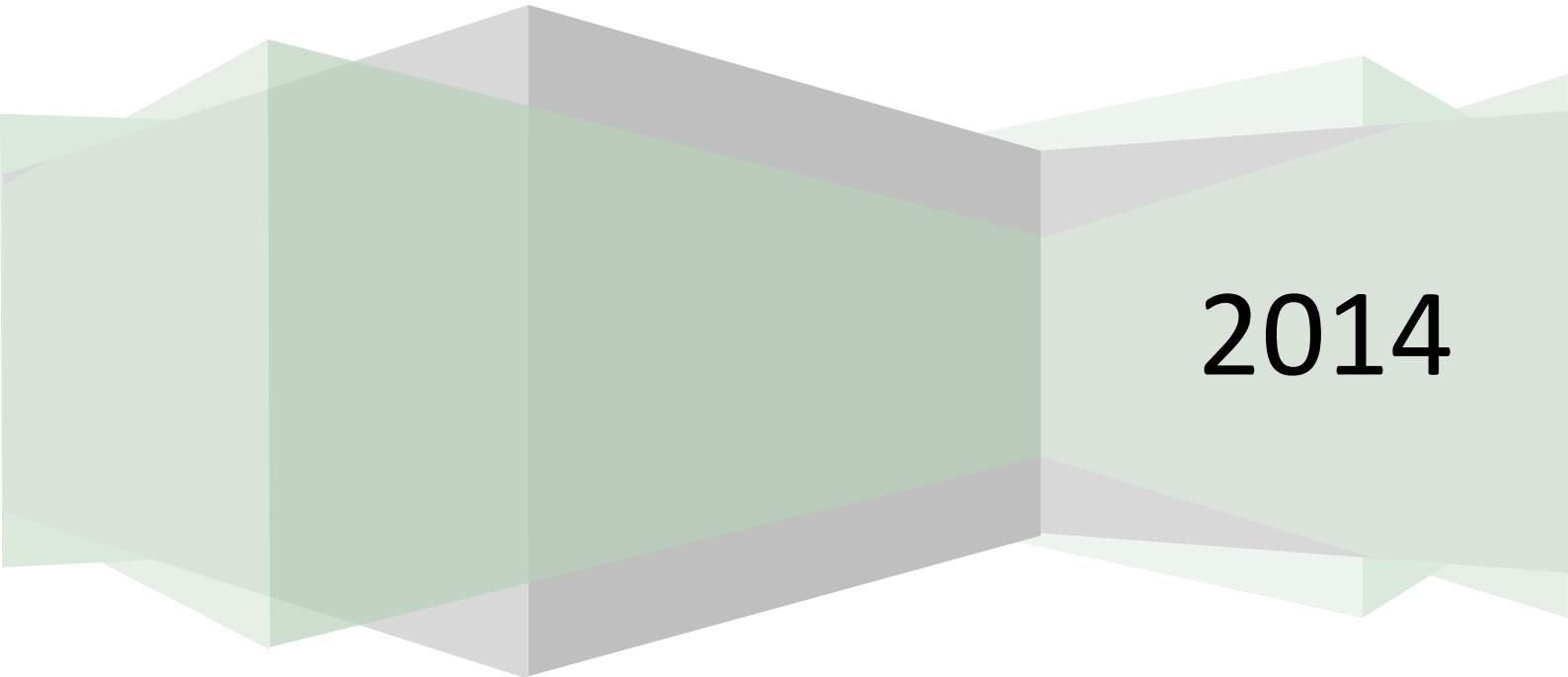


Juan José Conejero Serna

Kit de supervivencia para aprobar IC en Julio

Sólo para valientes

Grado Ingeniería Informática UA



2014

SUPERESCALARES

CAUCE

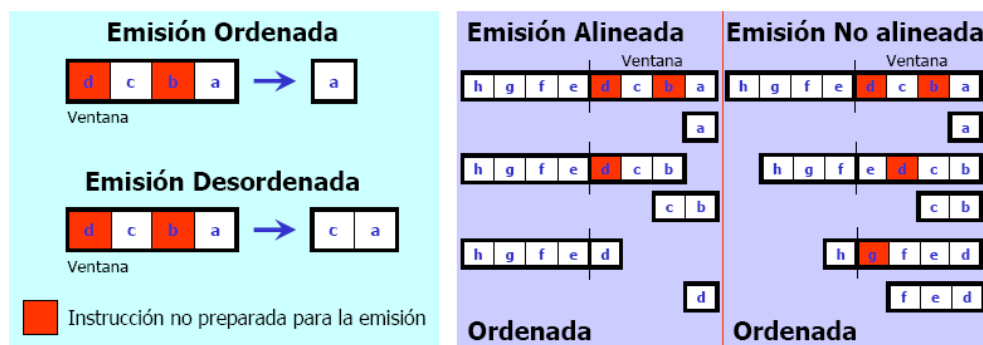
En una arquitectura superescalar, aparte de de las etapas comunes de una máquina normal, (IF, ID, EX, WEB) se añade la etapa de Emisión de instrucciones (ISS), que controla que puedan existir más de una operación realizándose a la vez.

EMISIÓN DE INSTRUCCIONES (ISS)

La ventana de instrucciones almacena las instrucciones pendientes ya decodificadas y se utiliza un bit para indicar si esa instrucción está disponible. Po su naturaleza, la emisión de las instrucciones depende del *alineamiento* y del *orden*:

La emisión es **alineada** si no pueden introducirse nuevas instrucciones en la ventana de instrucciones hasta que ésta no esté totalmente vacía. En la emisión **no alineada**, mientras que exista espacio en la ventana, se pueden ir introduciendo instrucciones para ser emitidas.

En la emisión **ordenada** se respeta el orden en que las instrucciones se han ido introduciendo en la ventana de instrucciones; si una instrucción incluida en la ventana de instrucciones no puede emitirse, las instrucciones que la siguen tampoco podrán emitirse, aunque tengan sus operandos y la unidad que necesitan esté disponible. En cambio, en la emisión **desordenada** no existe este *bloqueo*, ya que pueden emitirse todas las instrucciones que dispongan de sus operandos y de la correspondiente unidad funcional.



ESTUCTURAS

RENOMBRADO DE REGISTROS

En los procesadores con finalización desordenada, se pueden producir riesgos de dependencias WAR o WAW (Write After Read o Write After Write), y para ello haremos uso del **renombrado de registros** y **reorden de los mismos**, y dentro de ellos existen dos tipos: **Implementación Dinámica**, que se realizan durante la ejecución y requieren circuitería adicional, y la **Implementación Estática**, que se realiza durante la compilación y es la que vemos a continuación:

BUFFER RENOMBRAMIENTO

- Buffer de Renombrado de **Acceso Asociativo**
 - Permite varias escrituras pendientes a un mismo registro
 - Se utiliza el último bit para marcar cual ha sido la más reciente
- Buffer de Renombrado de **Acceso Indexado**
 - Sólo permite una escritura pendiente a un mismo registro
 - Se mantiene la escritura más reciente.

BUFFER REORDEN (ROB)

Una vez que hemos realizado todas las operaciones, los resultados se encuentran en el buffer de renombramiento, el **problema** surge al decidir el momento en que los resultados se escriben en los registros. Esto, puede coincidir o no con el **orden** en que las instrucciones estén en el programa, pero en cualquier caso, debe respetarse la *semántica* del programa. Para ello utilizaremos el **ROB**.

Además el **ROB** permite gestionar correctamente el procesamiento especulativo de las instrucciones de **salto** y las interrupciones; además que también se puede utilizar para el renombramiento.

PROCESAMIENTO DE INSTRUCCIONES DE SALTO

El efecto en los saltos de los superescalares es más pernicioso que en el resto, ya que en cada ciclo puede haber una instrucción de salto. Y también se compone de diferentes etapas:

- Detección de la instrucción de salto: Si se detecta la instrucción de un salto condicional en el momento de la captación y se conoce la dirección de destino en ese mismo ciclo, en el siguiente ciclo se pueden captar instrucciones a partir de esa dirección sin penalización de tiempo. Las posibilidades son:
 - **Detección paralela.**
 - **Detección anticipada**
 - **Detección integrada por captación**
- Gestión de saltos condicionales no resueltos: Si en el momento de evaluar la instrucción aun no se ha evaluado. Y se puede o utilizar un proceso especulativo o una comprobación directa.
- Acceso a las instrucciones de destino del salto: hay que implementar procesos que permitan el acceso más rápido a la secuencia de instrucciones.

Dentro de éste tipo de predicciones podemos encontrar dos tipos: predicción fija y predicción verdadera, que ésta segunda a su vez se descompone en predicción estática y dinámica.

Predicción Fija		
Siempre No Tomado		<ul style="list-style-type: none"> ▪ Toda condición de salto no resuelta, se predice que no da lugar a un salto. ▪ Después se evalúa si era buena.
Siempre Tomado		<ul style="list-style-type: none"> ▪ Toda condición de salto no resuelta, se predice que da lugar a un salto. ▪ Después se evalúa si era buena.
Predicción Verdadera		
Predicción Estática		<ul style="list-style-type: none"> ▪ Para ciertos códigos de operación, se predice que el salto se toma, y para otros que el salto no se toma. ▪ Saltos hacia atrás (bucles) y saltos hacia delante (if,then,else)
Predicción Dinámica	<i>Pr. Dinámica Implícita</i>	Predice hacer lo mismo que ocurrió la última vez que se ejecutó esa instrucción
	<i>Pr. Dinámica Estática</i>	Para cada instrucción de salto condicional, existe unos bits de historia que codifican la información del pasado de la instrucción.

PARALELISMO

CONCEPTOS BÁSICOS

- **Procesamiento distribuido:** Ejecución de múltiples aplicaciones al mismo tiempo utilizando múltiples recursos, situados en distintas localizaciones físicas.
- **Procesamiento paralelo:** división de aplicaciones en unidades independientes, y su ejecución en varios procesadores.

Y la clasificación de los computadores paralelos es:

- **Multiprocesadores:** Comparten el mismo espacio de memoria.
 - Mayor Latencia
 - Poco Escalable
 - Variables compartidas
 - Prog. Sencilla
- **Multicomputadores:** Cada procesador (o nodo) tiene su propio espacio de direcciones.
 - Menor Latencia
 - Mayor Escalabilidad
 - Paso de mensajes
 - Prog. Complicada

TIPOS DE PARALELISMO

Paralelismo de Datos

Implícito en operaciones con estructuras de datos tipo *vector* o *matriz*. Grandes volúmenes de datos independientes entre sí (superescalares y máquinas segmentadas).

Paralelismo Funcional

Reorganización de la estructura lógica de una aplicación. Nivel de Programas, Funciones, Bloques en funciones u Operaciones (granularidad de gruesa a fina).

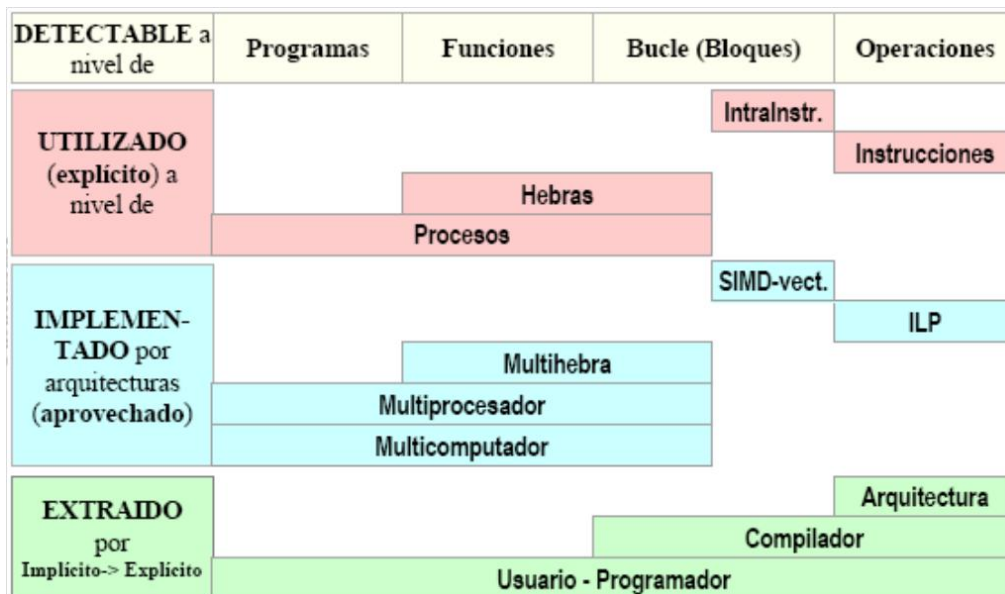
- *Explícito*: No presente de forma inherente en las estructuras de programación y que se debe indicar expresamente.
- *Implícito*: Presente debido a la propia estructura de los datos (vectores) o de la aplicación.

UNIDADES DE EJECUCIÓN

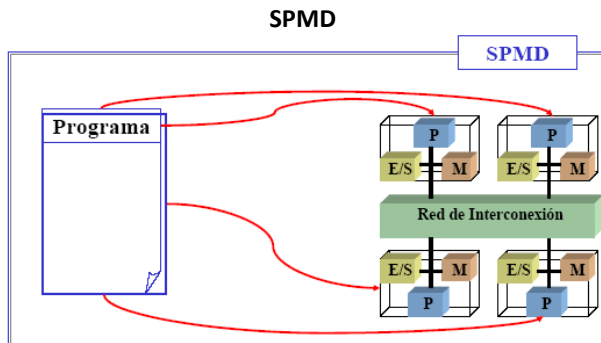
Hardware: Gestiona la ejecución de instrucciones (a nivel de procesador).

Software: Gestiona la ejecución de hilos y procesos.

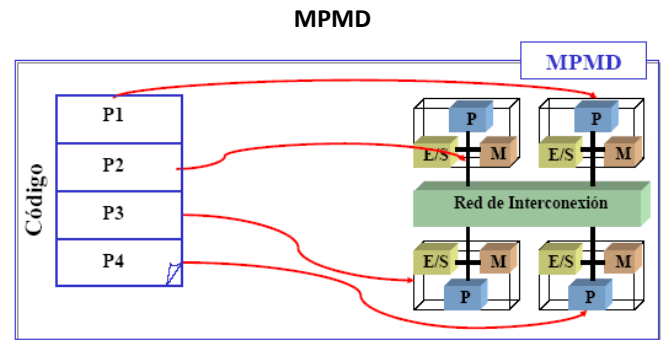
- *Proceso*: espacio de direcciones virtuales propio.
- *Hilos*: comparten direcciones virtuales, se crean y destruyen rápido y su comunicación también.



Modos de programación paralela:



Cada procesador realiza un bloque de código (if, for)

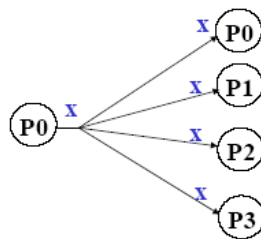


Se divide el programa en trozos y cada uno hace una función.

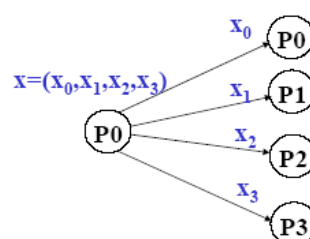
Comunicación entre estructuras paralelas:

Uno a todos

Difusión (*broadcast*)

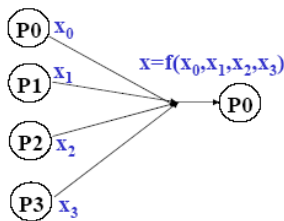


Dispersión (*scatter*)

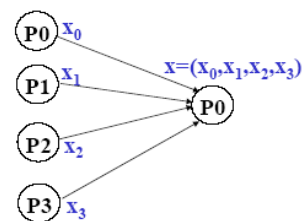


Todos a uno

Reducción



Acumulación (*gather*)



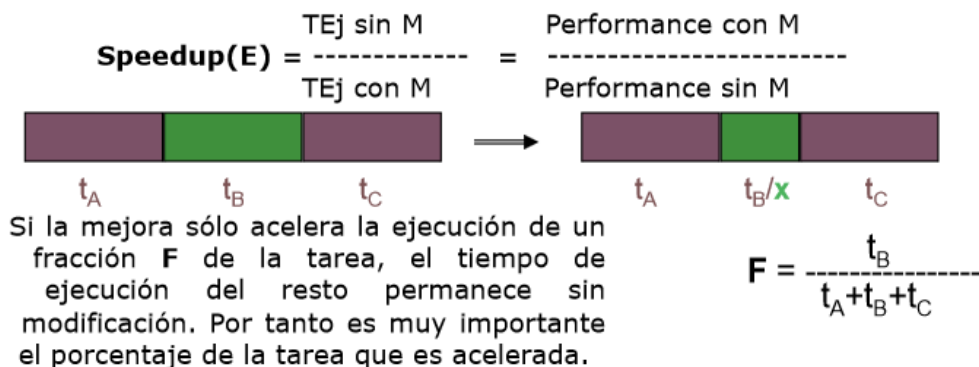
PROBLEMAS DE PARALELISMO

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.



LEY DE AMDHAL

$$\text{TEj}_{\text{nuevo}} = \text{TEj}_{\text{antiguo}} \times [(1 - \text{Fraccion}_{\text{mejora}}) + \text{Fraccion}_{\text{mejora}} / X]$$

$$\text{Speedup} = \text{TEj}_{\text{antiguo}} / \text{TEj}_{\text{nuevo}} = 1 / [(1 - \text{Fraccion}_{\text{mejora}}) + \text{Fraccion}_{\text{mejora}} / X]$$

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

$$\text{TEj}_{\text{nuevo}} = \text{TEj}_{\text{antiguo}} \times (0.9 + 0.1 / 2) = 0.95 \times \text{TEj}_{\text{antiguo}} \quad \text{Speedup} = 1 / 0.95 = 1.053$$

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.

$$\text{TEj}_{\text{nuevo}} = \text{TEj}_{\text{antiguo}} \times (0.1 + 0.9 / 100) = 0.109 \times \text{TEj}_{\text{antiguo}} \quad \text{Speedup} = 1 / 0.109 = 9.17$$

El uso de 100 procesadores sólo multiplica la velocidad por 9.17

REDES DE INTERCONEXIÓN

Elemento fundamental en arquitecturas paralelas con varios elementos de proceso que se comunican.

Los *parámetros básicos* que tienen estos elementos son:

- **Número de nodos (N)**
- **Grado del nodo (D):** Número de canales de entrada **Y** salida.
 - Nodos unidireccionales: Grado de salida y Grado de entrada
- **Diámetro de la red:** Longitud máxima del camino más corto entre dos nodos cualquiera

CLASIFICACIÓN DE REDES ESTÁTICAS (DIRECTAS)

Estrictamente Ortogonales: Cada nodo tiene al menos un enlace en cada dimensión, y supone un desplazamiento en una dimensión.

No ortogonales: estructura de árbol.

Propiedades:

- Grado
- Diámetro
- Simetría (se ve semejante desde cualquier nodo)
- Regularidad (todos los nodos tienen el mismo grado)

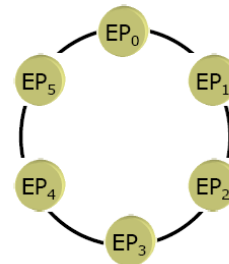
TIPOS DE REDES ESTÁTICAS

Anillo Unidireccional

➤ F. interconexión: $F+1(i) = (i+1) \bmod N$

➤ Grado de entrada/salida: 1/1

➤ Diámetro: $N-1$



Malla Abierta

F. interconexión:

➤ $F+1(i) = (i+1)$ si $i \bmod r < r-1$

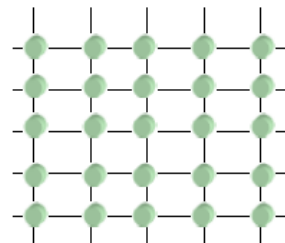
➤ $F-1(i) = (i-1)$ si $i \bmod r > 0$

➤ $F+r(i) = (i+r)$ si $i \bmod r < r-1$

➤ $F-r(i) = (i-r)$ si $i \bmod r > 0$

➤ Grado: 4

➤ Diámetro: $2(r-1)$, donde $N=r^2$



Malla Illiac

F. interconexión:

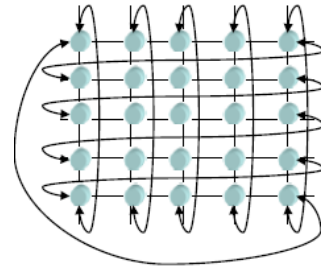
$$\triangleright F+1(i) = (i+1) \bmod N$$

$$\triangleright F-1(i) = (i-1) \bmod N$$

$$\triangleright F+r(i) = (i+r) \bmod N$$

$$\triangleright F-r(i) = (i-r) \bmod N$$

 \triangleright Grado: 4

 \triangleright Diámetro: $(r-1)$, donde $N=r^2$
**Malla toro**
 \triangleright n dimensiones, k nodos

 \triangleright F. interconexión toro 2D:

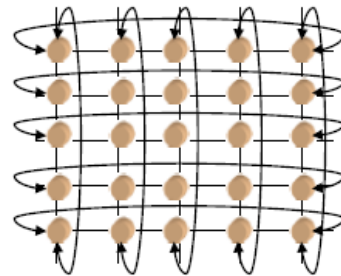
$$\triangleright F+1(i) = (i+1) \bmod r + (i \text{ DIV } r) \cdot r$$

$$\triangleright F-1(i) = (i-1) \bmod r + (i \text{ DIV } r) \cdot r$$

$$\triangleright F+r(i) = (i+r) \bmod N$$

$$\triangleright F-r(i) = (i-r) \bmod N$$

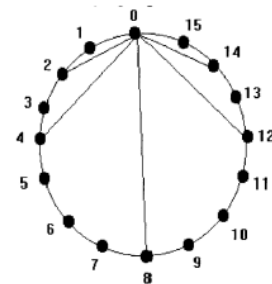
 \triangleright Grado: 4

 \triangleright Diámetro: $2 \cdot \frac{r}{2}$, donde $N=r^2$
**Desplazador Barril**
 \triangleright F. interconexión:

$$\triangleright B+k(i) = (i+2^k) \bmod N$$

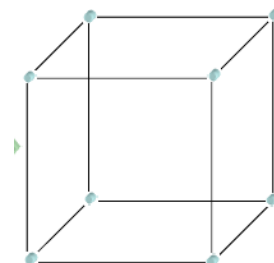
$$\triangleright B-k(i) = (i-2^k) \bmod N$$

$$\triangleright K=0 \dots n-1, n=\log N, i=0 \dots N-1$$

 \triangleright Grado: $2n - 1$
 \triangleright Diámetro: $n/2$
**Hipercubo**
 \triangleright F. interconexión:

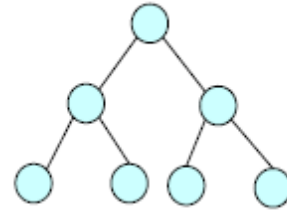
$$\triangleright F_i(h_{n-1}, \dots, h_i, \dots, h_0) = h_{n-1}, \dots, h_i, \dots, h_0$$

 \triangleright Grado: n ($n=\log N$)

 \triangleright Diámetro: n


Árbol Binario

- Balanceado: todas las ramas del árbol tienen la misma longitud
- Cuello de botella → nodo raíz
- N (balanceado) = $2^k - 1$ (k = niveles del árbol)
- Grado: 3
- Diámetro: $2(k-1)$



CLASIFICACIÓN DE REDES DINÁMICAS (INDIRECTAS)

Uso de conmutador: Las redes dinámicas se caracterizan por tener conmutadores (interruptores que desvían la red a un lado o a otro) para la interconexión ente nodos.

Modelo $G(N, C)$:

- N , conjunto conmutadores
- C , enlaces ente conmutadores

Distancia ente nodos: distancia entre los conmutadores que conectas los nodos + 2

TIPOS DE REDES DINÁMICAS

Crossbar	<ul style="list-style-type: none"> ➤ Conexión directa nodo-nodo ➤ Gran ancho de banda y capacidad de interconexión ➤ Conexión Proc. – Mem. → limitado por los accesos a memoria (columnas) ➤ Conexión $\text{Proc}(N) - \text{Proc}(N) \rightarrow$ máximo de N conexiones ➤ Coste elevado: $O(N \cdot M)$ 	
Redes Min	<ul style="list-style-type: none"> ➤ Conectan dispositivos de entrada con dispositivos de salida mediante un conjunto de etapas de conmutadores, donde cada conmutador es una red crossbar. ➤ Concentradores → n° entradas > n° salidas ➤ Expansores → n° salidas > n° entradas ➤ Conexión de etapas adyacentes → Patrón de conexión ➤ Patrón basado en permutaciones: conmutadores con el mismo número de entradas y salidas. ➤ Número de entradas a_n y número de salidas b_n (red $a_n \times b_n$) ➤ n etapas de conmutadores (C_0, C_1, \dots, C_{n-1}) ➤ Conmutadores $a \times b$ ➤ $a_{n-1-i} \times b_i$ conmutadores en la etapa C_i ➤ Funcionalidad de los conmutadores: barras cruzadas, reducción, difusión ➤ Subred de interconexión entre etapas: R_0, R_1, \dots ➤ Tipos de canales: unidireccionales, bidireccionales 	<p>Red Omega Red Mariposa Red Cubo Red Delta</p>

TÉCNICAS DE CONMUTACIÓN

Existen 5 técnicas de conmutación, pero solo veremos las resaltadas en negrita: **Store & Forward**, **Wormhole**, Cut-Through y Conmutación de Circuitos.

Store & Forward

- El conmutador almacena el **paquete completo** antes de ejecutar el algoritmo de encaminamiento.
- La unidad de transferencia (paquete) entre interfaces ocupa **sólo un canal** en cada instante.
- Almacenamiento en conmutadores: **múltiplos de un paquete**.
- El número de enlaces ociosos (libres) influye en el ancho de banda: para un tamaño de *buffer* 1, un paquete bloqueado deja ocioso un canal.

Latencia de transporte

$$T_{AR} = D \cdot \left[T_r + T_w \cdot \left(\frac{L}{W} + 1 \right) \right]$$

- **1 flit** = w bits
- **Cabecera** = 1 flit
- Tamaño total del paquete = **L bits + w bits** (cabecera)
- **D**= Distancia fuente-destino = D parejas conmutador-enlace
- **T_w** = tiempo para que un flit atravesase una etapa conmutador/enlace
- **T_r** = tiempo de encaminamiento (routing)



Wormhole

- En cuanto llega la cabecera al conmutador se ejecuta el algoritmo de encaminamiento y se reenvía.
- La unidad de transferencia es el mensaje
- La transferencia se hace a través de un camino segmentado. La unidad de transferencia puede ocupar varios canales.
- Almacenamiento en conmutadores: múltiplos de un flit.
- El número de enlaces ociosos influye en el ancho de banda: para un tamaño de buffer 1, un paquete bloqueado deja ociosos varios canales.

Latencia de transporte

$$T_V = D \cdot (T_r + T_w) + T_w \cdot \left\lceil \frac{L}{W} \right\rceil$$

$$T_V: T_{\text{cabecera}} + T_{\text{resto}}$$

MEMORIA

TIPOS

La clasificación de procesadores atendiendo a la distribución de memoria es:

UNA (UNIFORM MEMORY ACCESS)

- **Memoria centralizada**, compartida entre procesadores
- **Dependencia** funcional entre procesadores (*fuertemente acoplado*)
- Cada procesador dispone de un **caché**
- Periféricos **compartidos** entre procesadores
- **Sincronización** entre procesadores utilizando variables compartidas.

NUMA (NON-UNIFORMA MEMORY ACCESS)

- **Memoria compartida** con tiempo de acceso dependiente de la ubicación de procesadores
- Cada procesador dispone de una **memoria local** (*débilmente acoplado*)
- Sistema de **acceso global** a memoria, **también**.
- **Ventajas**: Escalado de memoria con + coste/rendimiento y reducción de latencia.

COMA (CACHE-ONLY MEMORY ARCHITECTURE)

- Sólo se usa una **caché** como memoria
- Caso particular de *NUMA* donde **las memorias distribuidas se convierten en cachés**
- Las cachés forman un mismo **espacio global de direcciones**
- Acceso a las cachés por **directorio distribuido**

CONSISTENCIA DE MEMORIA

*“Un modelo de consistencia de memoria especifica el orden en el cual las operaciones de acceso a memoria deben **parecer** haberse realizado”.*

Se debe **garantizar**:

1. Cada lectura de una dirección proporcione el **último valor** escrito en dicha dirección
2. Si se **escribe varias** veces en esa dirección se debe retornar el último valor escrito.
3. Si se escribe donde se ha leído anteriormente, **no se debe obtener la lectura previa** a la escritura
4. No se puede escribir en una dirección si la escritura depende de una **condición** que **no** se cumple.

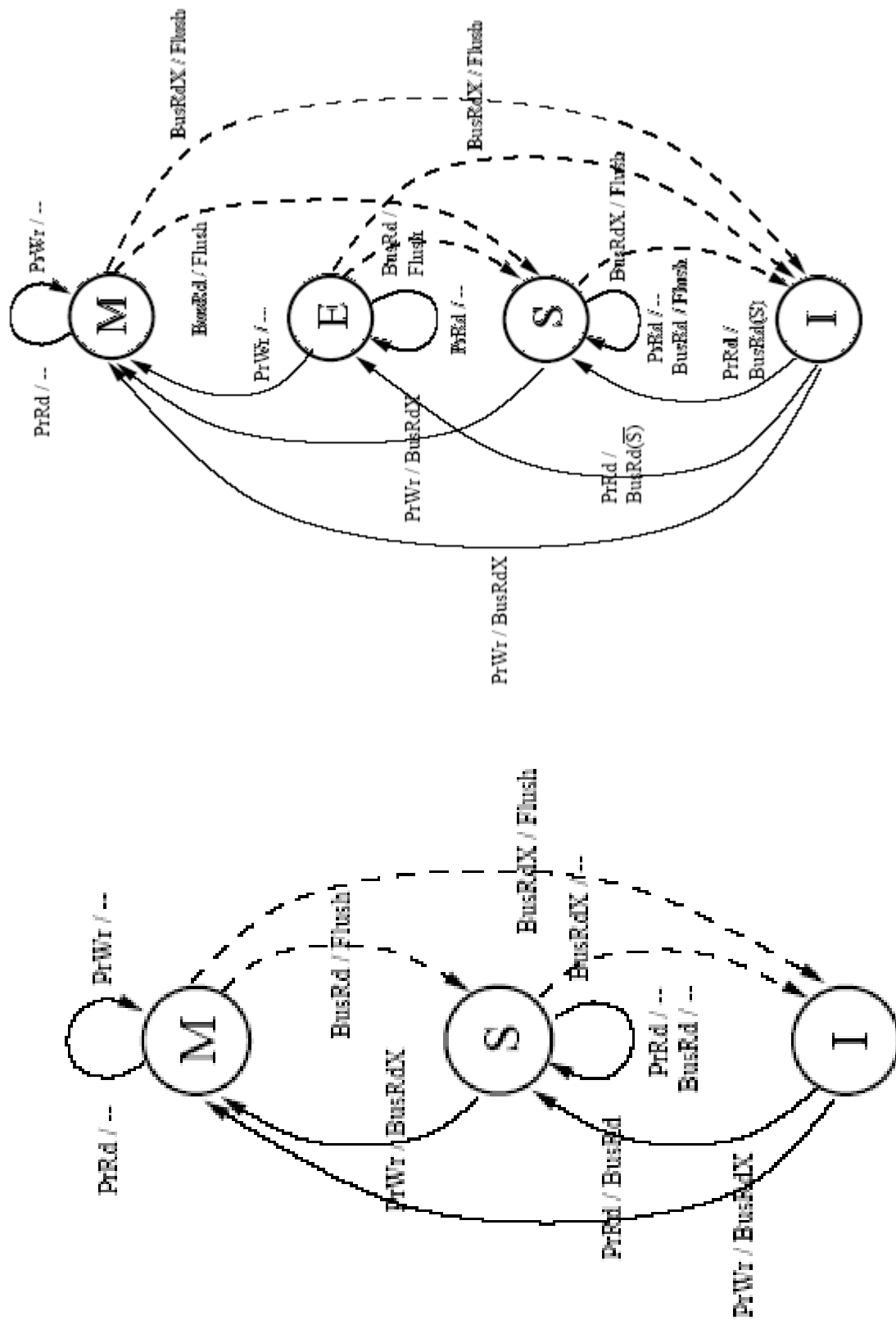
Un sistema de memoria es coherente si:

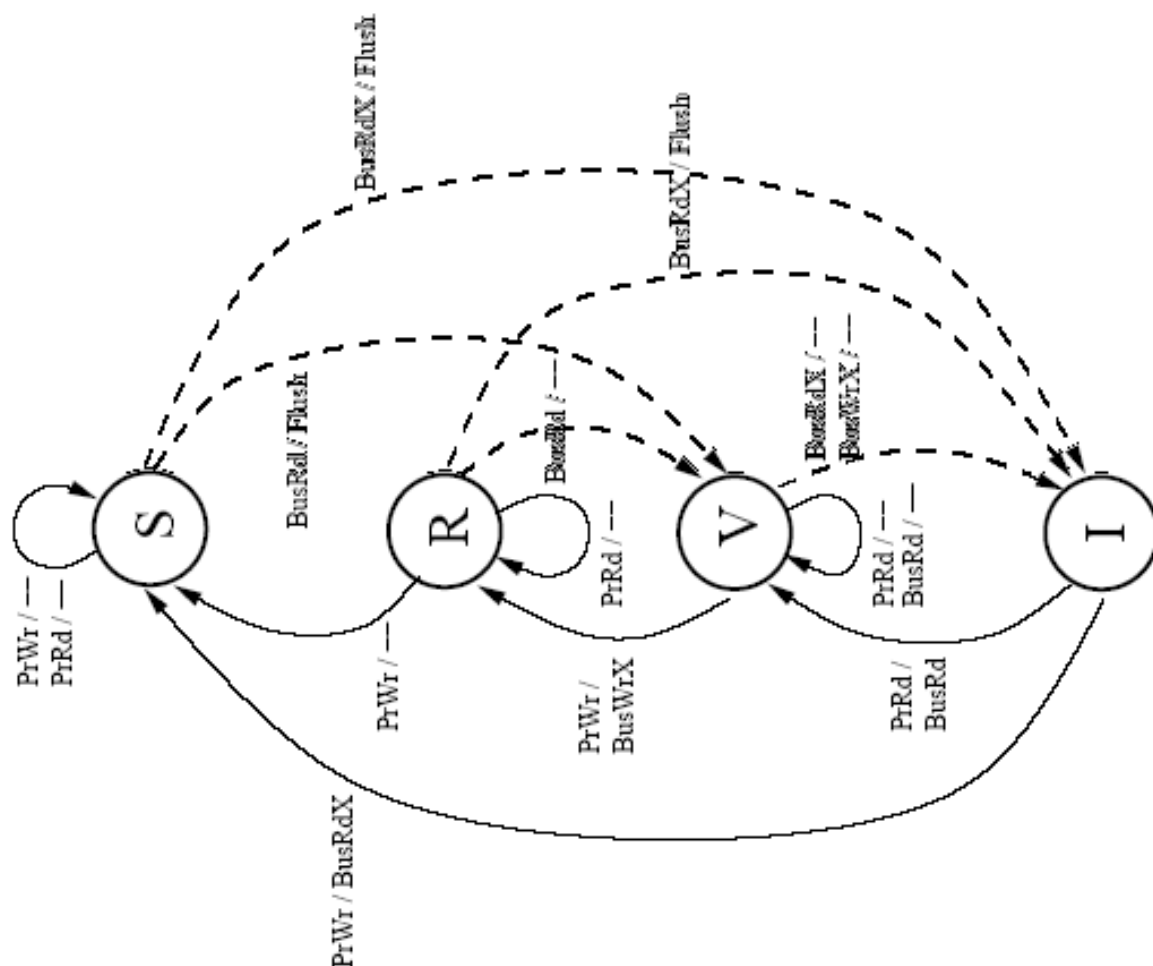
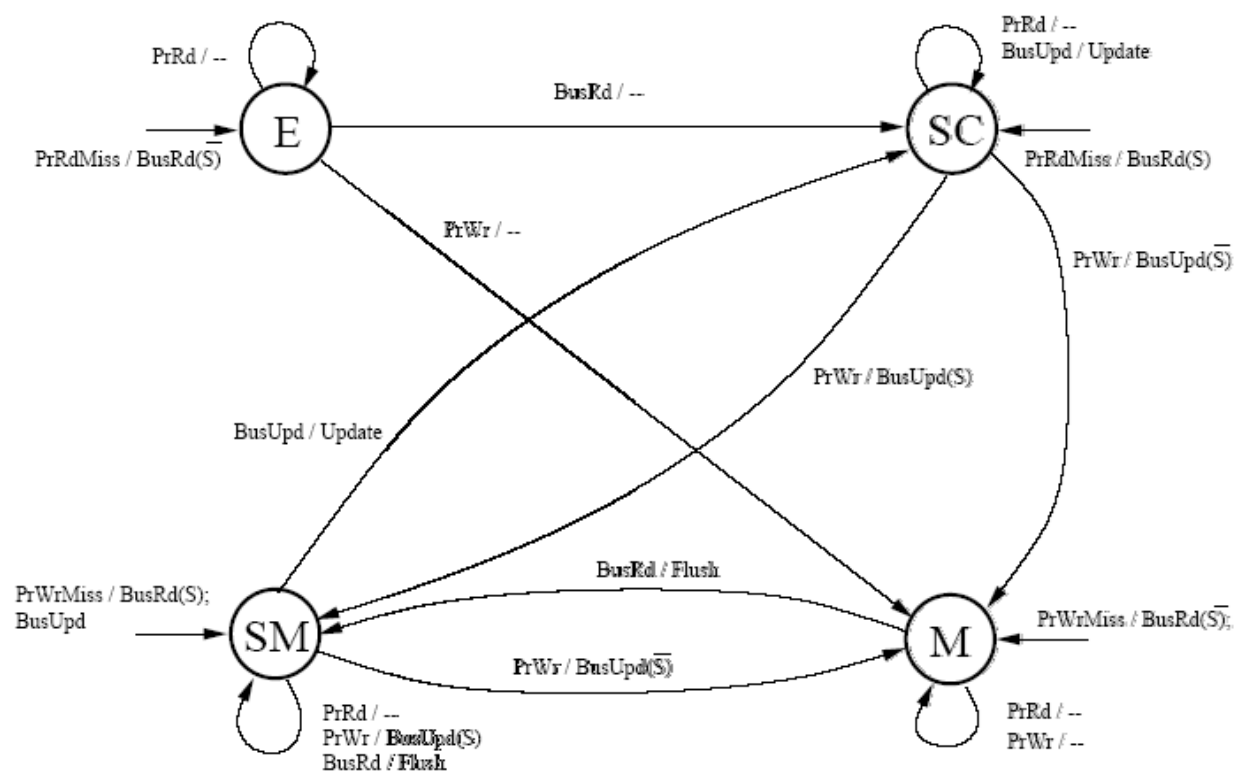
1. Cualquier lectura de un dato devuelve el valor más reciente escrito de ese dato.
2. El resultado de cualquier ejecución de un programa es tal que, para cada localización es posible construir una ordenación secuencial de las operaciones realizadas en el cual:
 - Las operaciones emitidas ocurren en la secuencia indicada y en el orden en el que dicho procesador las emite.
 - El valor devuelto por cada operación de lectura es el valor escrito por la última escritura.

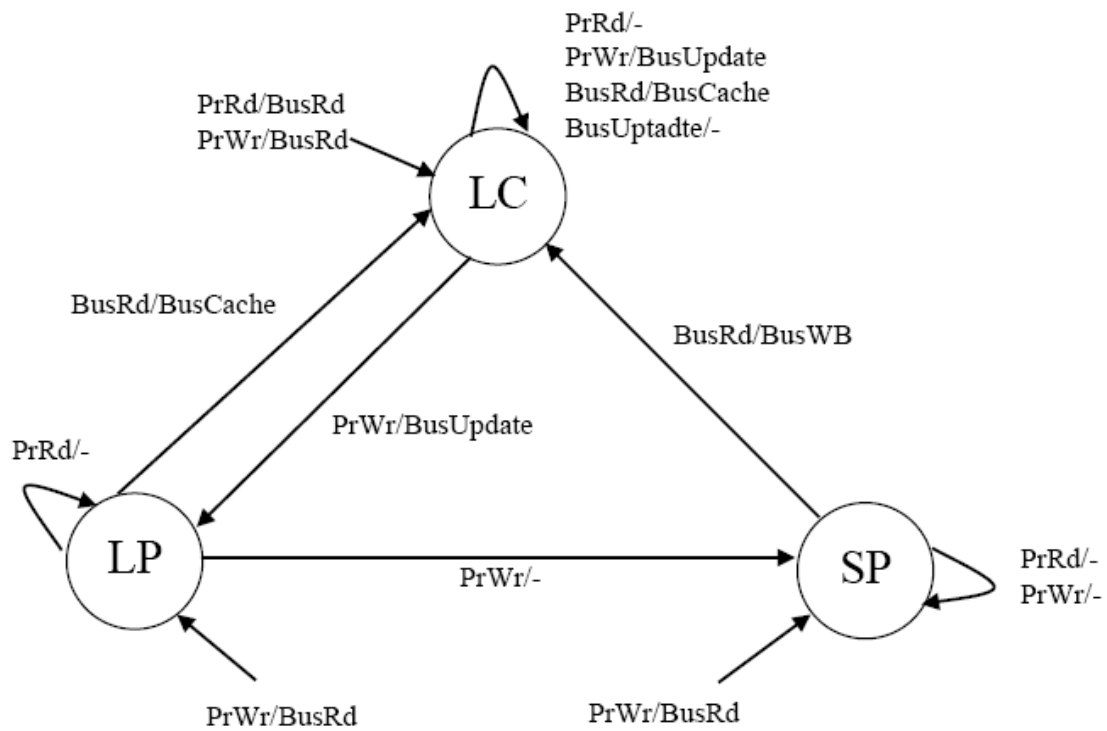
COHERENCIA DE MEMORIA

DIAGRAMAS

Sólo diagramas, ya que en casi todos los exámenes piden o corregir o dibujar: (A continuación en grande).







CONCLUSIÓN

Mierda de computadores. Quien cojones me mandaría a mí meterme en la carrera.