# TEORÍA:

**1. Explica la utilidad del buffer de renombrado y enumera los tipos que hay según su direccionamiento.**

El buffer de renombrado es una técnica para evitar el efecto de las dependencias WAR y WAW. Existen dos tipos de buffer de renombrado:

Con **acceso asociativo**:

- Permite varias escrituras pendientes a un mismo registro.
- Se utiliza el bit último para marcar cuál ha sido la escritura más reciente.

Con **acceso indexado**:

- Sólo permite una escritura pendiente a un mismo registro.
- Se mantiene la escritura más reciente.

**2. Explica la diferencia entre predicción dinámica explícita y predicción dinámica implícita.**

Predicción **dinámica explícita**.

Para cada instrucción de salto existen unos bits específicos que codifican la información de historia de dicha instrucción de salto.

Predicción **dinámica implícita**.

No hay bits de historia propiamente dichos, sino que se almacena la dirección de la instrucción que se ejecutó después de la instrucción de salto en cuestión.

**3. Justifica la diferencia que existe entre multicomputadores y multiprocesadores en términos de latencia y escalabilidad.**

En los **multicomputadores**, la latencia en el acceso a memoria es menor que en los multiprocesadores. Esto es debido a que el acceso concurrente a memoria compartida por parte de los multiprocesadores provoca que el tiempo de lectura y escritura en memoria aumente.

En los **multiprocesadores**, la escalabilidad respecto al número de núcleos es menor que en los multicomputadores. Esto es debido a que el rendimiento de los programas paralelos no aumenta en la misma proporción que el aumento de núcleos, debido a los conflictos de acceso a memoria.

**4. ¿Cuál es la característica distintiva de las redes de interconexión dinámicas frente a otros tipos de redes de interconexión?**

A diferencia de otros tipos de redes de interconexión, las redes de interconexión dinámicas se caracterizan porque pueden variar su topología durante la ejecución de los procesos o entre la ejecución de dos procesos.

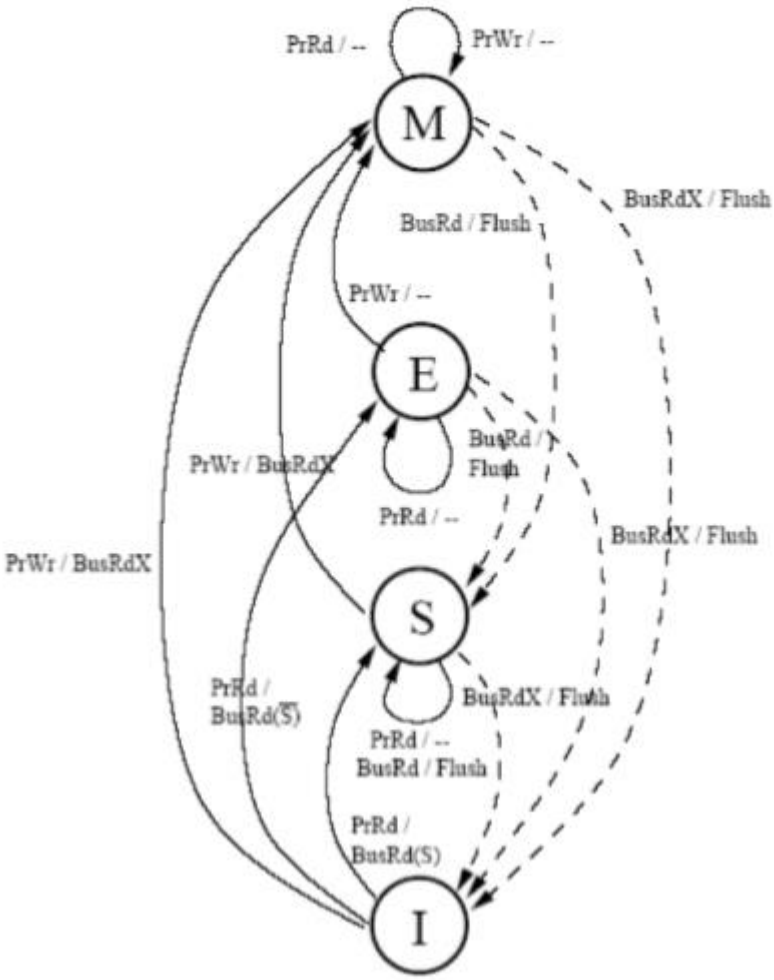
**6. Explica en qué consiste una arquitectura vectorial.**

Una arquitectura vectorial es una arquitectura orientada al procesamiento de vectores.

Esta arquitectura utiliza un repertorio de instrucciones especializado y se caracteriza por:

- Cálculo de los componentes del vector de forma independiente, obteniendo buenos rendimientos.
- Cada operación vectorial codifica gran cantidad de cálculos, reduciendo el número de instrucciones y evitando riesgos de control.
- Se optimiza el uso de memoria, usando entrelazado de memoria y organizaciones S y C.

**7. Dibuja el diagrama de transiciones del protocolo de coherencia de caché MESI.**




**1. Explica la utilidad de la ventana de instrucciones y explica brevemente cómo puede ser el orden de emisión y el alineamiento de la ventana.**

La ventana de instrucciones se emplea para almacenar las instrucciones pendientes. Las instrucciones se cargan en la ventana una vez decodificadas y se utiliza un bit para indicar si un operando está disponible o no. Una instrucción puede ser emitida cuando tiene todos sus operandos disponibles y la unidad funcional donde se procesará.

El orden de emisión de las instrucciones puede ser:

- **Emisión Ordenada.** Las instrucciones se emiten en el mismo orden en el que entran a la ventana. Se espera hasta que la instrucción esté preparada para ser emitida.
- **Emisión Desordenada.** Las instrucciones se emiten sin orden alguno. Se emite la que esté preparada. Sin esperar a una en particular.

El alineamiento de la venta de instrucciones puede ser:

- **Emisión Alineada**. No se reciben nuevas instrucciones hasta que no se vacía la ventana de instrucciones.
- **Emisión Desalineada.** Se pueden recibir nuevas instrucciones siempre que haya sitio.

**2. Explica brevemente las estructuras que permiten la gestión de predicción de los saltos dinámica explícita.**

**Branch Target Buffer (BTB): bits acoplados.**

Almacena la dirección de destino de los últimos saltos tomados y los bits de predicción de ese salto. Los campos de la BTB se actualizan después de ejecutar el salto, cuando se conoce si el salto fue tomado o no y la dirección de destino del salto. La predicción implícita no emplea bits de predicción, el problema de esto es que sólo se pueden predecir aquellas instrucciones de salto que están en la BTB.

**Tabla de historia de saltos (BHT): bits desacoplados.**

- BTAC. Almacena la dirección de destino de los últimos saltos tomados.
- BHT. Almacena los bits de predicción de todas las instrucciones de salto condicional.

La ventaja de esta estructura es que puede predecir instrucciones que no están en la BTAC, sim embargo, necesita más hardware.

**Bits de predicción en la caché.**

Cuando se capta una instrucción de la caché, si se trata de una instrucción de salto condicional, accede en paralelo a los bits de predicción y si el salto se predice como tomado se accede a la instrucción destino del salto. Para acceder a la instrucción destino del salto se usa una BTB independiente a la que se añade el índice sucesor a la l-cache.

Las ventajas de esta estructura son que se puede predecir instrucciones que no están en la BTB y no añade una cantidad extra de hardware excesiva.

**3. Describe ejemplos de problemas, cuya solución consista en un conjunto de procesos o hebras que se comunican mediante cada uno de los siguientes patrones:**

La **difusión o broadcast** consiste en un ordenador maestro que envía un mismo mensaje a todas las máquinas hijo. Un ejemplo de esto sería las actualizaciones del sistema operativo que un ordenador maestro envía a los ordenadores de sus clientes por medio de una OTA (Over-The-Air).

La **dispersión o scatter** consiste en un ordenador maestro que divide un mensaje en partes y envía una parte diferente a cada una de las máquinas hijo. Un ejemplo de esto sería la suma de las componentes de dos vectores, de esta forma, cada máquina hijo tomaría una componente de cada vector y realizaría la suma.

**4. ¿Cuál es la característica distintiva de las redes de interconexión directas frente a los otros tipos de redes de interconexión (indirectas y de medio compartido)?**

A diferencia de otros tipos de redes de interconexión, las redes de interconexión directas emplean enlaces directos fijos entre los nodos mientras que las redes de interconexión indirectas y de medio compartido están formadas por enlaces no permanentes que se reconfiguran en función de la demanda.

**6. Explica brevemente qué tipos de paralelismo existen y en qué niveles puede aplicarse.**

**Paralelismo de datos.** La misma función, instrucción, etc. se ejecuta en paralelo, pero en cada una de esas ejecuciones se aplica sobre un conjunto de datos distinto.

**Paralelismo funcional.** Varias funciones, tareas, instrucciones, etc (iguales o distintas) se ejecutan en paralelo.

- **Nivel de instrucción (ILP).** Se ejecutan en paralelo las instrucciones de un programa. Granularidad fina.
- **Nivel de bucle o hebra (Thread).** Se ejecutan en paralelo distintas iteraciones de un bucle o secuencias de instrucciones de un programa. Granularidad fina/media.
- **Nivel de procedimiento (Proceso).** Distintos procedimientos que constituyen un programa se ejecutan simultáneamente. Granularidad media.
- **Nivel de programa.** La plataforma ejecuta en paralelo programas diferentes que pueden corresponder, o no, a una misma aplicación. Granularidad gruesa.

**7. ¿Qué diferencia existe entre el protocolo de coherencia de caché MESI y MSI? ¿Qué ventajas tiene uno de ellos sobre el otro?**

**MSI.** Protocolo de invalidación básico que usa tres estados necesarios en cualquier caché post-escritura para distinguir bloques válidos que no han sido modificados de aquellos que han sido modificados. Está compuesto por 3 estados: Inválido (I), Compartido (S) y Modificado (M).

**MESI.** Protocolo de invalidación de 4 estados. Refinado para aplicaciones “secuenciales” que corren en multiprocesadores. MESI añade el estado Exclusivo (E), que indica que el bloque es la única copia (exclusiva) del sistema multiprocesador y que no está modificado, ningún otro procesador tiene el bloque en la caché y la memoria principal está actualizada.

La ventaja de MESI sobre MSI es que, al ser exclusivo, es posible realizar una escritura o pasar al estado modificado sin ninguna transición en el bus, al contrario que en el caso de estar en el estado compartido; pero no implica pertenencia, así que al contrario que en el estado modificado la caché no necesita responder al observar una petición de dicho bloque.

En el MSI el programa cuando lee y modifica un dato tiene que generar 2 transacciones incluso en el caso de que no exista compartición (sólo presente en una caché) del dato.

**1. Explica brevemente cuáles son los tipos de paralelismo que podemos encontrar en un sistema informático.**

**Paralelismo funcional.** Se obtiene a través de la reorganización de la estructura lógica de una aplicación. Existen diferentes niveles de paralelismo funcional según las estructuras que se reorganicen.

**Paralelismo de datos.** Implícito en operaciones con estructuras de datos tipo vector o matriz. Está relacionado con operaciones realizadas sobre grandes volúmenes de datos que sean independientes entre sí.

- **Paralelismo explícito.** Paralelismo no presente de forma inherente en las estructuras de programación y que se debe indicar expresamente.
- **Paralelismo implícito.** Paralelismo presente (aunque puede no estar ejecutándose de forma paralela) debido a la propia estructura de los datos (vectores) o de la aplicación (bucle).

**2. ¿Existe alguna diferencia entre el acceso a memoria concurrente y el acceso simultáneo en una máquina vectorial? Si es así, explícala muy brevemente.**

La diferencia entre el acceso a memoria concurrente y el acceso simultáneo radica en que en el acceso concurrente cada módulo puede trabajar por separado, de forma asíncrona, leyendo direcciones diferentes y cada módulo tiene su propio registro de dirección y puede ir trabajando paralelamente a los demás.

El acceso simultáneo se basa en simultanear los accesos en todos los módulos y, tratar de superponer ese tiempo con la salida de los datos de memoria.

**3. ¿Cuáles son las similitudes y las diferencias entre el buffer de renombrado y el buffer de reorden?**

**Similitudes:**

En los procesadores con finalización desordenada, se pueden producir riesgos de dependencias WAR o WAW (Write After Read o Write after Write), y para minimizar el impacto que provocarían en la ejecución del programa haremos uso del renombrado de registros y reorden de los mismos, y dentro de ellos existen dos tipos:

- **Implementación Dinámica.** Se realizan durante la ejecución y requieren circuitería adicional.
- **Implementación Estática.** Se realiza durante la compilación.

**Diferencias:**

El buffer de renombrado se encarga de modificar el nombre de las instrucciones, en cambio, el buffer de reorden se encarga de modificar el orden de ejecución de las instrucciones.

**4. Explica en qué consiste el procesamiento especulativo de los saltos.**

El procesamiento especulativo es una manera de gestionar los saltos condicionales no resueltos. Se realiza una tarea que podría no ser necesaria; la idea consiste en llevar a cabo un trabajo antes de saber si será necesario con la intención de evitar el retraso que supondría realizarlo después de saber que sí es necesario. Si el trabajo en cuestión resulta ser innecesario, se revierten los cambios realizados por ese trabajo y se ignoran los resultados obtenidos.

**5. Defina y enumere las diferencias entre proceso y una hebra.**

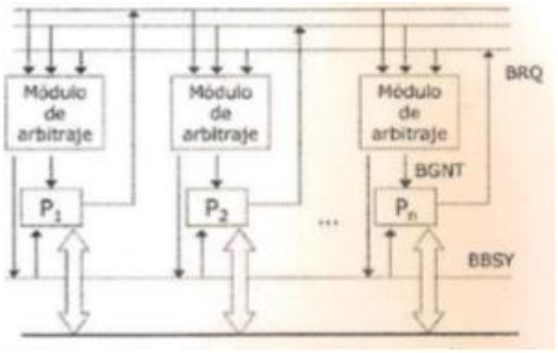
**Proceso.** Es una unidad de actividad que se caracteriza por la ejecución de una secuencia de instrucciones, un estado actual, y un conjunto de recursos del sistema asociados.

Cada proceso tiene un espacio de direcciones virtuales propio.

**Hebra.** Secuencia de tareas encadenadas muy pequeña que puede ser ejecutada por un sistema operativo.

Las hebras comparten direcciones virtuales, se crean y destruyen más rápido y la comunicación también es más rápida que en los procesos.

**6. La imagen de la figura indica un tipo de arbitraje para un cierto tipo de redes de interconexión.**




**a) ¿De qué tipo de redes se trata? ¿La prioridad es estática o dinámica? ¿Por qué?**

Se trata de una red de medio compartido (arbitraje del bus), en concreto, se trata de autoarbitraje (distribuido-paralelo). La prioridad de la red es estática.

**1. Explica brevemente qué es el paralelismo en computación. ¿Siempre mejora el rendimiento? (Explícalo poniendo ejemplos).**

El paralelismo es una forma de computación en la cual varios cálculos pueden realizarse simultáneamente, basado en el principio de dividir los problemas grandes para obtener varios problemas pequeños, que son posteriormente solucionados en paralelo.

El paralelismo en computación no siempre mejora el rendimiento, cuando hay cargas computacionales de bajo coste, realizar la paralelización ralentiza el tiempo respecto a la ejecución del programa en secuencial.

**2. ¿Para qué sirve el buffer de renombrado? ¿En qué etapa o etapas se usa? Pon un ejemplo descriptivo sencillo de su funcionamiento.**

El buffer de renombrado sirve para evitar el efecto de las dependencias WAR, o Antidependencias (en la emisión desordenada) y WAW, o Dependencias de Salida (en la ejecución desordenada).

El buffer de renombrado se usa en la etapa de decodificación para leer los registros fuente que estén renombrados y para renombrar el registro destino. Al acabar la ejecución se escriben los resultados en el buffer de renombrado, y al acabar se escriben los resultados en los registros correspondientes.

**3. ¿En qué consiste la predicción dinámica implícita? Explica cómo funciona poniendo un ejemplo.**

La **predicción dinámica** consiste en que cada vez que se predice una misma instrucción de salto la predicción puede ser diferente según la historia previa (si se han tomado o no las anteriores ejecuciones).

La **predicción implícita** no contiene información en forma de bits de historia, sino que almacena la dirección destino de salto de la instrucción que se ejecutó después del salto.

**4. ¿Cuál es la principal diferencia entre multicomputadores y multiprocesadores?**

La principal diferencia entre un multiprocesador y un multicomputador es el número de equipos implicados en cada uno.

Un **multiprocesador** es una máquina que opera con varias CPU que comparten el mismo espacio de memoria.

Un **multicomputador** es un conjunto de ordenadores interconectados entre sí para funcionar como un único equipo. Cada procesador tiene su propio espacio de memoria.

**5. Explica brevemente los diferentes tipos de paralelismo que se pueden encontrar.**

**Paralelismo de datos.** La misma función, instrucción, etc. se ejecuta en paralelo, pero en cada una de esas ejecuciones se aplica sobre un conjunto de datos distinto.

**Paralelismo funcional.** Varias funciones, tareas, instrucciones, etc (iguales o distintas) se ejecutan en paralelo.

- **Nivel de instrucción (ILP).** Se ejecutan en paralelo las instrucciones de un programa. Granularidad fina.
- **Nivel de bucle o hebra (Thread).** Se ejecutan en paralelo distintas iteraciones de un bucle o secuencias de instrucciones de un programa. Granularidad fina/media.
- **Nivel de procedimiento (Proceso).** Distintos procedimientos que constituyen un programa se ejecutan simultáneamente. Granularidad media.
- **Nivel de programa.** La plataforma ejecuta en paralelo programas diferentes que pueden corresponder, o no, a una misma aplicación. Granularidad gruesa.

**7. Explica cómo se distribuyen los bits necesarios para mantener la coherencia utilizando un protocolo basado en directorios. ¿Para qué sirve cada bit?**

**En el directorio:** cada entrada de bloque tiene unos bits de presencia por cada caché y un bit de inconsistencia única:

- **Bits de presencia.** Especifican la presencia en las cachés de copias del bloque de memoria.
- **Bit de inconsistencia única.** Cuando este bit está activado, sólo uno de los bits de presencia está a uno, es decir, sólo existe una caché con la copia de ese bloque o línea, con lo que sólo esa caché tiene permiso para actualizar la línea.

**Por cada caché:**

- **Bits de validación (v).** Indica si la copia es válida o no.
- **Bit de privacidad (p).** Indica si la copia tiene permiso de escritura, es decir, cuando este bit es uno entonces es la única copia que existe de esta línea en las cachés y, por tanto, tiene permiso para escribir.

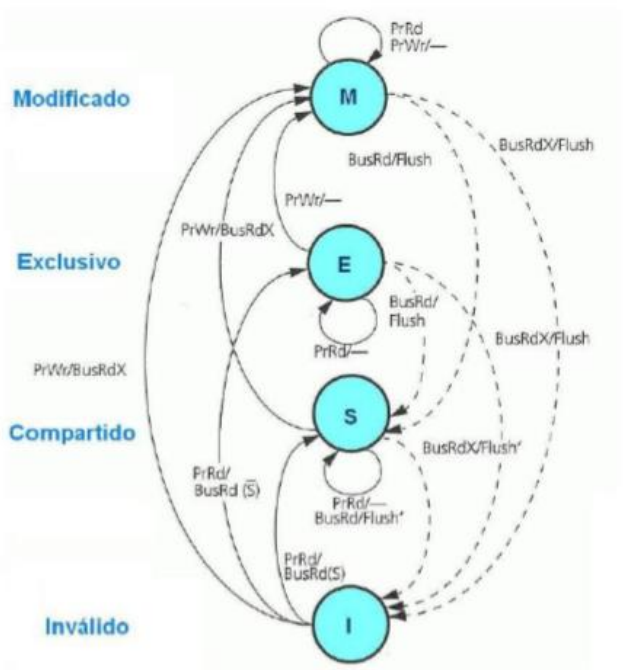
# MESI

▪ Modificado (M). Significa que es el único componente con una copia válida del bloque. El resto tienen una copia no actualizada.

▪ Exclusivo (E). Significa que es el único componente con una copia válida del bloque. El resto tienen una copia no válida y la memoria tiene una copia actualizada del bloque.

▪ Compartido (S). Supone que el bloque es válido en esta caché en memoria y al menos en otra caché.

▪ Inválido (I). Supone que el bloque no está físicamente en la caché, o si se encuentra ha sido invalidado por el contador como consecuencia de la escritura en la copia del bloque situada en la otra caché.



## Latencia

Tamaño-paquete → m B de datos ⇒ $L = m \cdot 8$ bits
n B de cabecera ⇒ $W = n \cdot 8$ bits

$t_r \equiv$ Tiempo de enrutamiento (routing)
$t_w = \frac{W}{V_{TN}} \equiv$ Velocidad de transmisión entre nodos.

⚠ Tener ambos en las mismas unidades.

Toro/Malla k D de N nodos → $N = r^{dimensiones} \rightarrow r = \sqrt[d]{N}$

Nodo_origen = $N_o$ nodos = $(x_o, y_o, z_o)$
Nodo_destino = $N_d$ nodos = $(x_d, y_d, z_d)$

$N_o$ $\lfloor r$ → $z_o$ // c $\lfloor r$ → $y_o$ // $x_o$ //
$N_d$ $\lfloor r$ → $z_d$ // c $\lfloor r$ → $y_d$ // $x_d$ //

Esto dependerá de las dimensiones, ya que es una coordenada con k dimensiones.

$D = |Origen - Destino| = (|x_o - x_d|, |y_o - y_d|, |z_o - z_d|) = (x, y, z) = x + y + z$

Adaptar dependiendo de la tipología.

**Store & Forward**

$$T_{AR} = D\left[t_r + t_w\left(\frac{L}{W} + 1\right)\right]$$

**Wormhole**

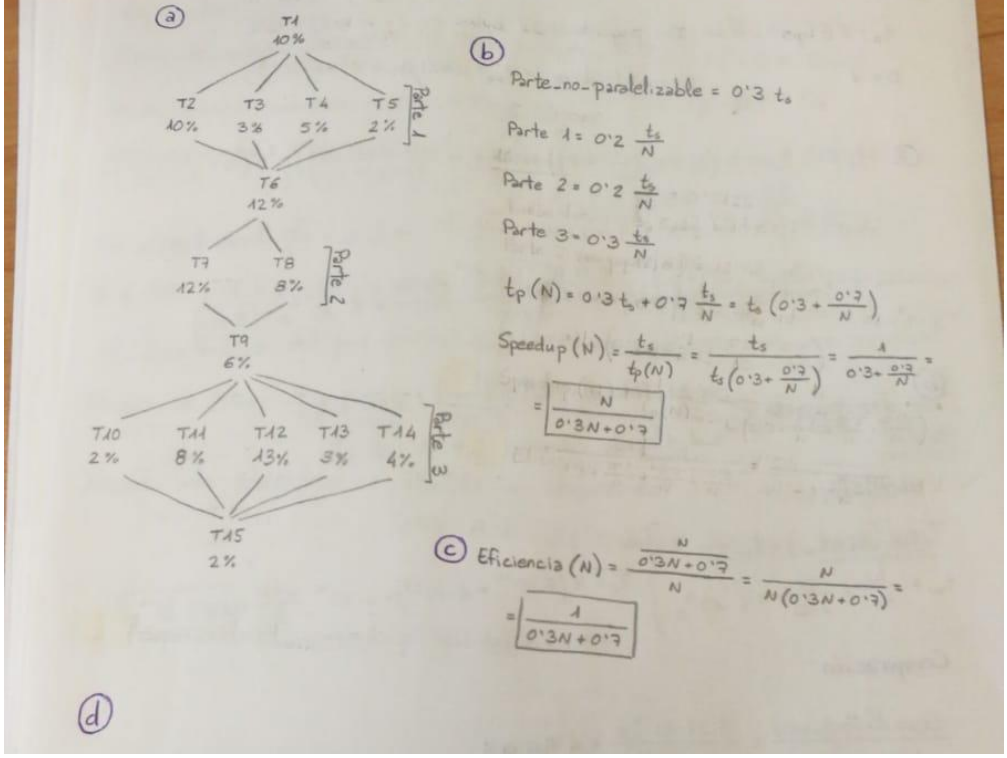
→ Buffer solo de entrada

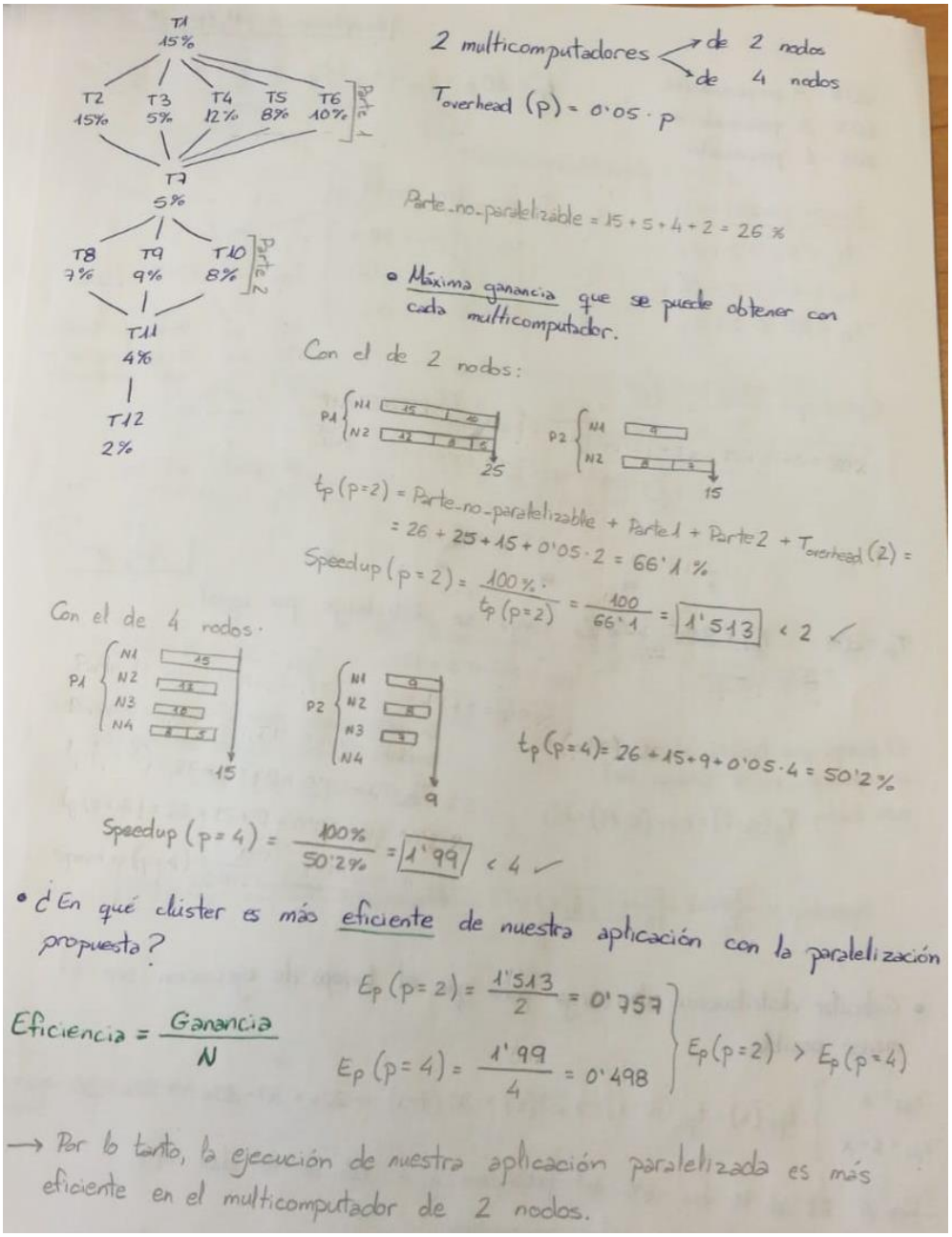
$$T_v = D(t_r + t_w) + t_w \frac{L}{W}$$

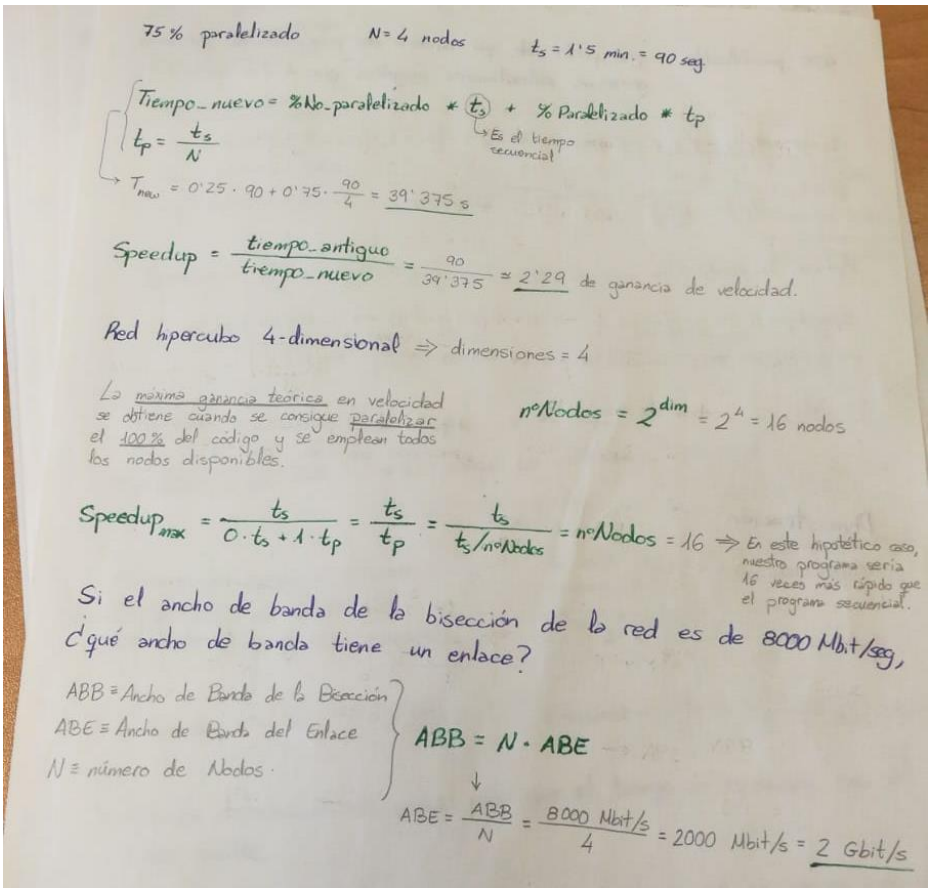
→ Buffer en entrada y salida

$$T_v = D(t_r + t_s + t_w) + \max(t_s, t_w)\frac{L}{W}$$

## Enero 2018, ej. 4




T1 15%
T2 15%
T3 5%
T4 12%
T5 8%
T6 10%
Parte 1
T7 5%
T8 7%
T9 9%
T10 8%
Parte 2
T11 4%
T12 2%
2 multicomputadores de 2 nodos de 4 nodos
Toverhead (p) = 0'05 · p
Parte_no_paralelizable = 15 + 5 + 4 + 2 = 26 %
Máxima ganancia que se puede obtener con cada multicomputador.
Con el de 2 nodos:
P1 N1 N2 25
P2 N1 N2 15
tp(p=2) = Parte_no_paralelizable + Parte1 + Parte2 + Toverhead(2) = 26 + 25 + 15 + 0'05 · 2 = 66'1 %
Speedup(p=2) = 100 % / tp(p=2) = 100 / 66'1 = 1'513 < 2
Con el de 4 nodos.
P1 N1 N2 N3 N4 15
P2 N1 N2 N3 N4 9
tp(p=4) = 26 + 15 + 9 + 0'05 · 4 = 50'2 %
Speedup(p=4) = 100 % / 50'2 % = 1'99 < 4
¿En qué clúster es más eficiente de nuestra aplicación con la paralelización propuesta?
Eficiencia = Ganancia / N
Ep(p=2) = 1'513 / 2 = 0'757
Ep(p=4) = 1'99 / 4 = 0'498
Ep(p=2) > Ep(p=4)
→ Por lo tanto, la ejecución de nuestra aplicación paralelizada es más eficiente en el multicomputador de 2 nodos.



75% paralelizado N = 4 nodos $t_s$ = 1'5 min. = 90 seg.

Tiempo_nuevo = %No_paralelizado * ($t_s$) + % Paralelizado * $t_p$

↳ Es el tiempo secuencial

$t_p = \frac{t_s}{N}$

$T_{new} = 0'25 \cdot 90 + 0'75 \cdot \frac{90}{4} = 39'375$ s

$Speedup = \frac{tiempo\_antiguo}{tiempo\_nuevo} = \frac{90}{39'375} \simeq 2'29$ de ganancia de velocidad.

Red hipercubo 4-dimensional ⇒ dimensiones = 4

La máxima ganancia teórica en velocidad se obtiene cuando se consigue paralelizar el 100% del código y se emplean todos los nodos disponibles.

$nºNodos = 2^{dim} = 2^4 = 16$ nodos

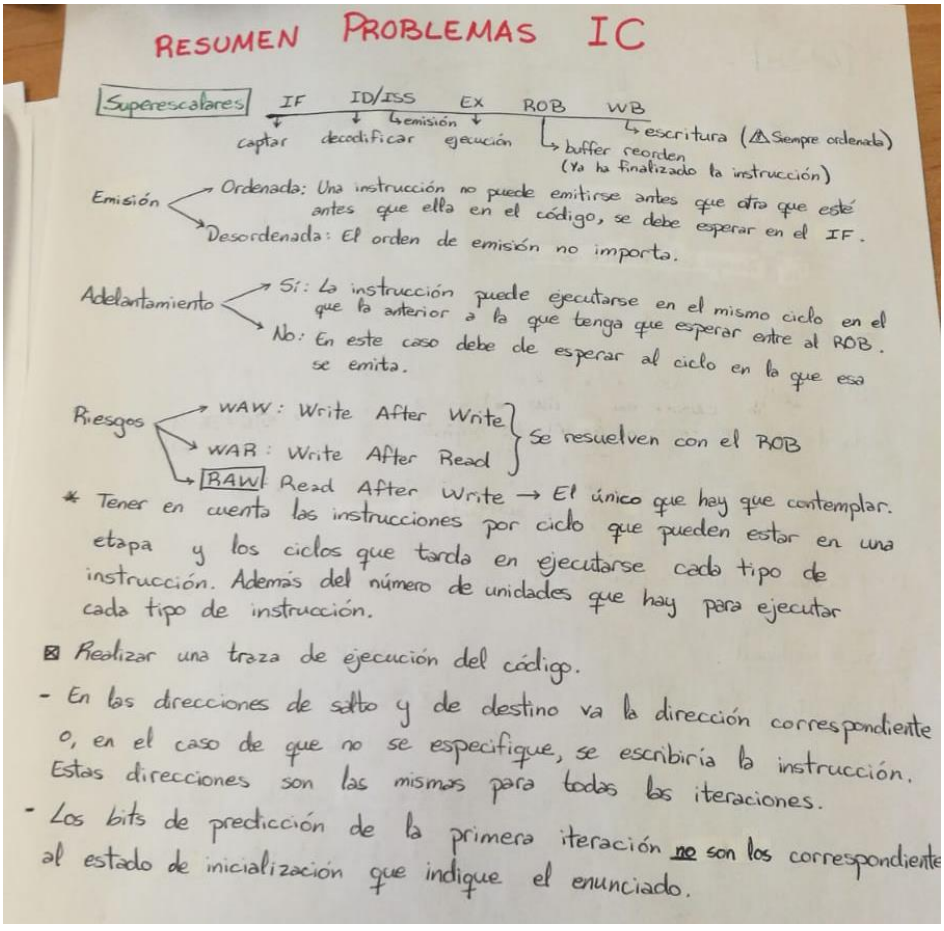
$Speedup_{max} = \frac{t_s}{0 \cdot t_s + 1 \cdot t_p} = \frac{t_s}{t_p} = \frac{t_s}{t_s / nºNodos} = nºNodos = 16$ ⇒ En este hipotético caso, nuestro programa sería 16 veces más rápido que el programa secuencial.

Si el ancho de banda de la bisección de la red es de 8000 Mbit/seg, ¿qué ancho de banda tiene un enlace?

ABB ≡ Ancho de Banda de la Bisección
ABE ≡ Ancho de Banda del Enlace
N ≡ número de Nodos

$ABB = N \cdot ABE$

$ABE = \frac{ABB}{N} = \frac{8000 \text{ Mbit/s}}{4} = 2000 \text{ Mbit/s} = 2 \text{ Gbit/s}$



# RESUMEN PROBLEMAS IC

**Superescalares** IF → ID/ISS → EX → ROB → WB

- IF: captar
- ID/ISS: decodificar; ↳ emisión
- EX: ejecución
- ROB: ↳ buffer reorden (Ya ha finalizado la instrucción)
- WB: ↳ escritura (⚠ Siempre ordenada)

Emisión
- Ordenada: Una instrucción no puede emitirse antes que otra que esté antes que ella en el código, se debe esperar en el IF.
- Desordenada: El orden de emisión no importa.

Adelantamiento
- Sí: La instrucción puede ejecutarse en el mismo ciclo en el que la anterior a la que tenga que esperar entre al ROB.
- No: En este caso debe de esperar al ciclo en la que esa se emita.

Riesgos
- WAW: Write After Write
- WAR: Write After Read

Se resuelven con el ROB

- RAW: Read After Write → El único que hay que contemplar.

* Tener en cuenta las instrucciones por ciclo que pueden estar en una etapa y los ciclos que tarda en ejecutarse cada tipo de instrucción. Además del número de unidades que hay para ejecutar cada tipo de instrucción.

☒ Realizar una traza de ejecución del código.

- En las direcciones de salto y de destino va la dirección correspondiente o, en el caso de que no se especifique, se escribiría la instrucción. Estas direcciones son las mismas para todas las iteraciones.
- Los bits de predicción de la primera iteración no son los correspondiente al estado de inicialización que indique el enunciado.