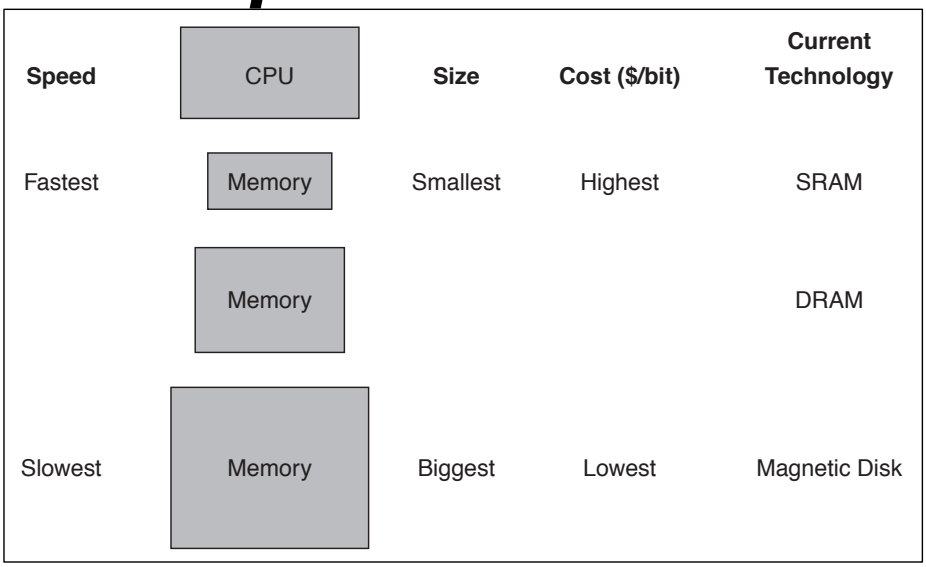
Jerarquía de memoria - Motivación

- Idealmente uno podría desear una capacidad de memoria infinitamente grande, tal que cualquier …. palabra podría estar inmediatamente disponible … Estamos forzados a reconocer la posibilidad de construir una jerarquía de memorias en la que cada nivel tenga mayor capacidad que el que le precede pero su acceso no es tan veloz.

A.W. Burks, H.H. Goldsteind y J. von Neumann
(Discusión preliminar del diseño lógico de un instrumento electrónico de computación-1946)

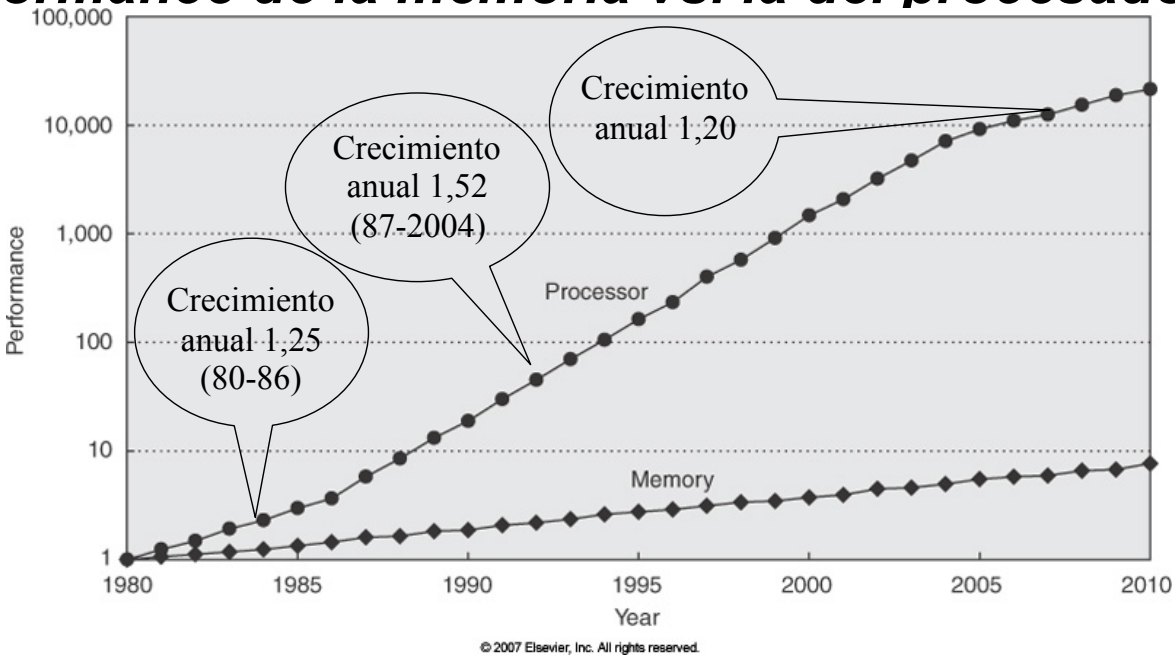
Jerarquía de memoria - Idea

Jerarquía de memoria



- Disminuir el tiempo efectivo de acceso a los elementos en memoria.
- Relación velocidad - capacidad.
- Jerarquía de memoria basada en la localidad de referencia.
- Replicación de instrucciones y datos almacenados en niveles más bajos de la jerarquía.

Performance de la memoria vs. la del procesador



- Una forma de disminuir la brecha es usar una jerarquía de memoria.
- El uso de una jerarquía de memoria crea la ilusión de contar con una memoria de tamaño suficiente y con alta velocidad.

Principio de localidad

- Los programas tienden a repetir el uso de datos e instrucciones que usaron recientemente. Regla 90/10.
- Localidad temporal:
  - Las referencias probablemente se repitan en un futuro cercano.
  - Un ítem referenciado tenderá a ser referenciado pronto.
- Localidad espacial:
  - Las referencias probablemente sean cercanas a la última.
  - Los ítems cercanos a una referencia tenderán a ser referenciados.

Localidad temporal

- Las instrucciones de un loop se ejecutan varias veces.
- Si en un loop hay *n* referencias ⇒
  - La primer referencia es a memoria y
  - Las demás se resuelven en cache.

- Tiempo promedio de acceso = (n * Tc + Tm) / n
  = Tc + Tm / n

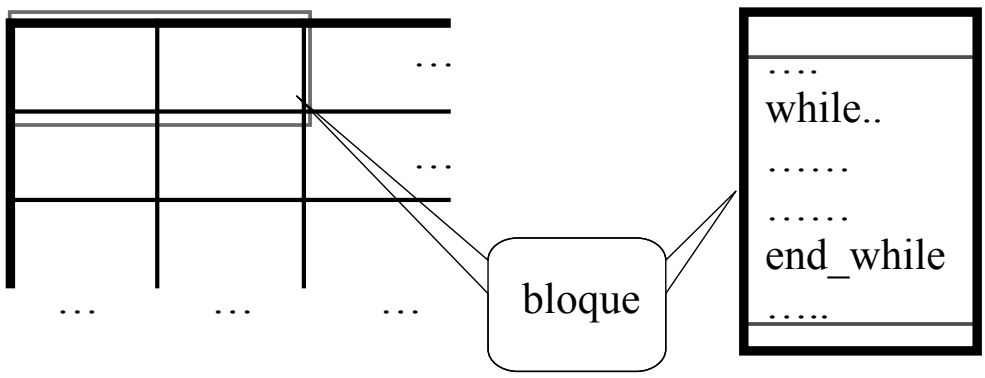
n = nº de accesos, Tc = tmpo de cache, Tm = tmpo de mem

Localidad temporal - ejemplo

- Tiempo promedio de acceso = Tc + Tm / n
- Si n = 10, Tc = 25 ns, Tm = 200 ns
- Tiempo promedio = 45 ns
- Sin el uso de cache el tiempo promedio es de 200 ns

Localidad espacial

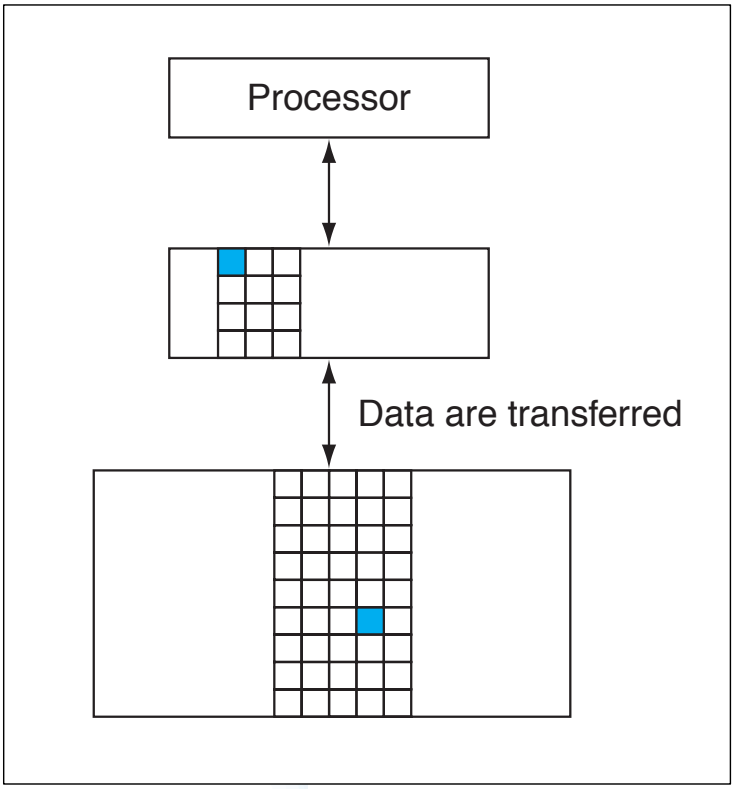
- En un programa secuencial
  una instrucción se encuentra a continuación de la última ejecutada.
- Los datos en arreglos, tablas y registros muestran localidad espacial.



Jerarquía de memoria - ideas importantes

- La localidad de referencia puede aplicarse en cualquier nivel de la jerarquía.
- El nivel inferior es la memoria virtual.
- Los datos se copian entre niveles adyacentes.
- El nivel superior está más cerca de la velocidad del procesador y tiene menor capacidad.
- La unidad mínima transferida entre niveles es el bloque (o línea).

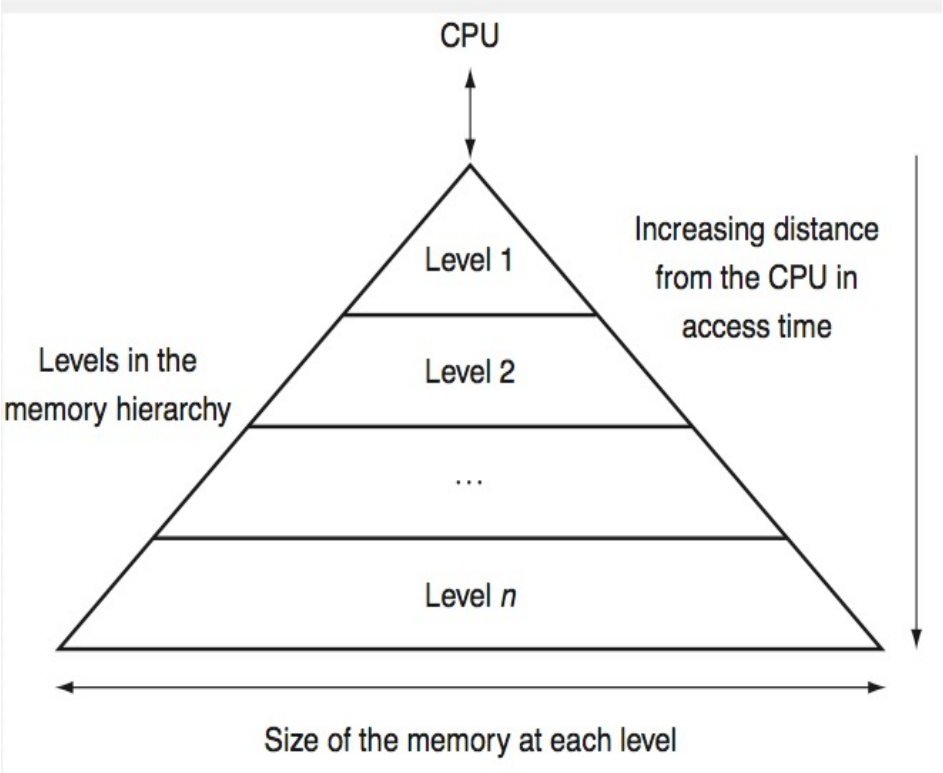
La unidad de información en cada nivel es el bloque o línea.



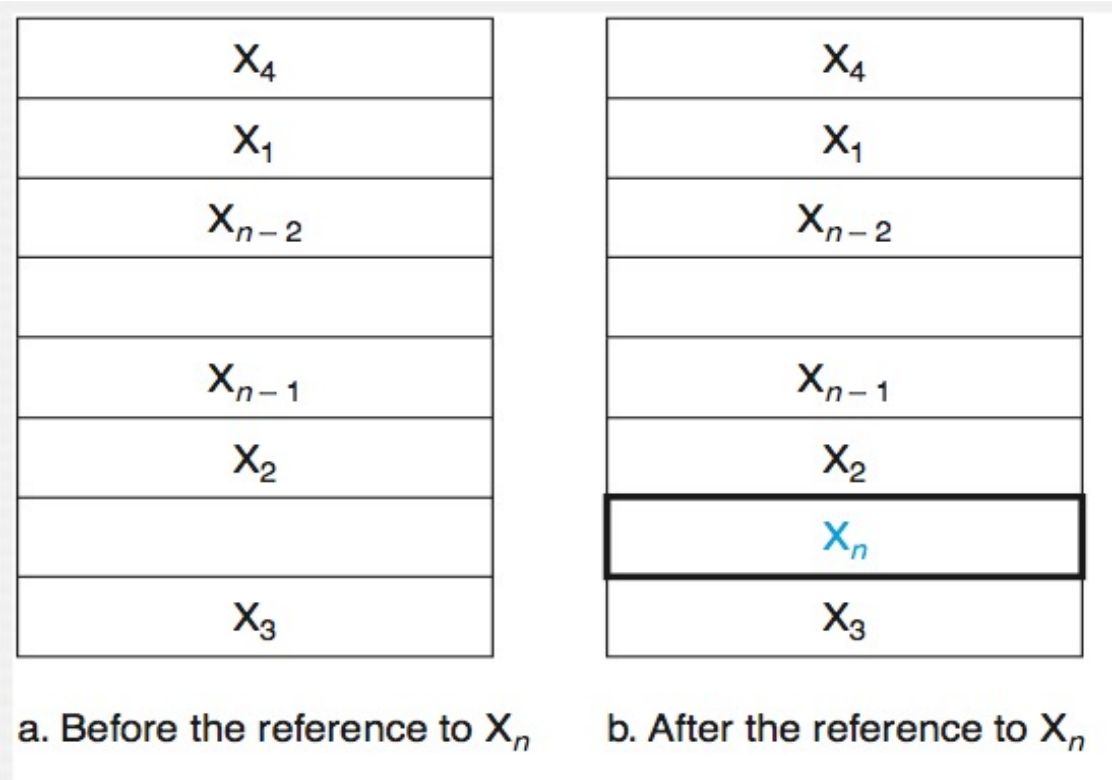
Jerarquía de memoria: algunas definiciones

- Hit: el dato requerido está en algún bloque del nivel más alto
- Si el dato no es encontrado en el nivel superior, se produce un miss.
- Tasa de hit: fracción de accesos a memoria encontrados en el nivel superior.
- Tiempo de hit: es el tiempo para acceder a la memoria de nivel superior. Incluye determinar si fue hit o miss.
- Penalidad por miss: tiempo para reemplazar un bloque en el nivel superior con el correspondiente bloque del nivel inferior, más el tiempo para proveer lo requerido.

Estructura de la jerarquía de memoria

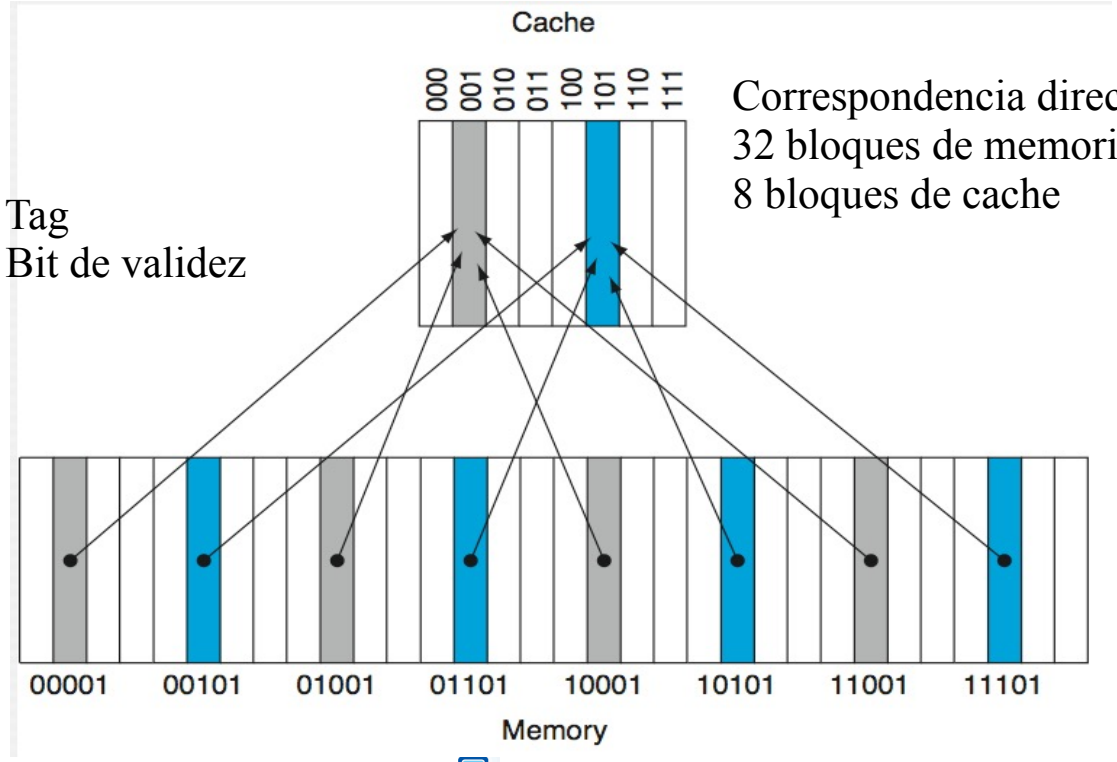


Fetch de un bloque desde el nivel inferior



Ubicación de los bloques en cache

Bloque cache=(dirección del bloque en mem) modulo (# bloques en cache)



Ejemplo: secuencia de nueve accesos

- Cache de 8 entradas y memoria de 32 lugares.
- Campos de la cache: índice, validez, tag, dato.
- Estado inicial: vacía y todos los bloques no válidos.

Dirección decimal de referencia	Dirección binaria de referencia	Hit o miss en cache	Bloque asignado (donde encontrar o ubicar)
22	10110_{dos}	miss (3.b)	$(10110_{dos}\ mod\ 8) = 110_{dos}$
26	11010_{dos}	miss (3.c)	$(11010_{dos}\ mod\ 8) = 010_{dos}$
22	10110_{dos}	hit	$(10110_{dos}\ mod\ 8) = 110_{dos}$
26	11010_{dos}	hit	$(11010_{dos}\ mod\ 8) = 010_{dos}$
16	10000_{dos}	miss (3.d)	$(10000_{dos}\ mod\ 8) = 000_{dos}$
3	00011_{dos}	miss (3.e)	$(00011_{dos}\ mod\ 8) = 011_{dos}$
16	10000_{dos}	hit	$(10000_{dos}\ mod\ 8) = 000_{dos}$
18	10010_{dos}	miss (3.f)	$(10010_{dos}\ mod\ 8) = 010_{dos}$
16	10000_{dos}	hit	$(10000_{dos}\ mod\ 8) = 000_{dos}$

Actualización después de un fallo

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

a. The initial state of the cache after power-on

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

b. After handling a miss of address (10110_{two})

Actualización después de fallos

Index	V	Tag	Data
000	N		
001	N		
010	Y	11_{two}	Memory (11010_{two})
011	N		
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

c. After handling a miss of address (11010_{two})

Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})
001	N		
010	Y	11_{two}	Memory (11010_{two})
011	N		
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

d. After handling a miss of address (10000_{two})

Actualización después de fallos

Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})
001	N		
010	Y	11_{two}	Memory (11010_{two})
011	Y	00_{two}	Memory (00011_{two})
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

e. After handling a miss of address (00011_{two})

Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})
001	N		
010	Y	10_{two}	Memory (10010_{two})
011	Y	00_{two}	Memory (00011_{two})
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

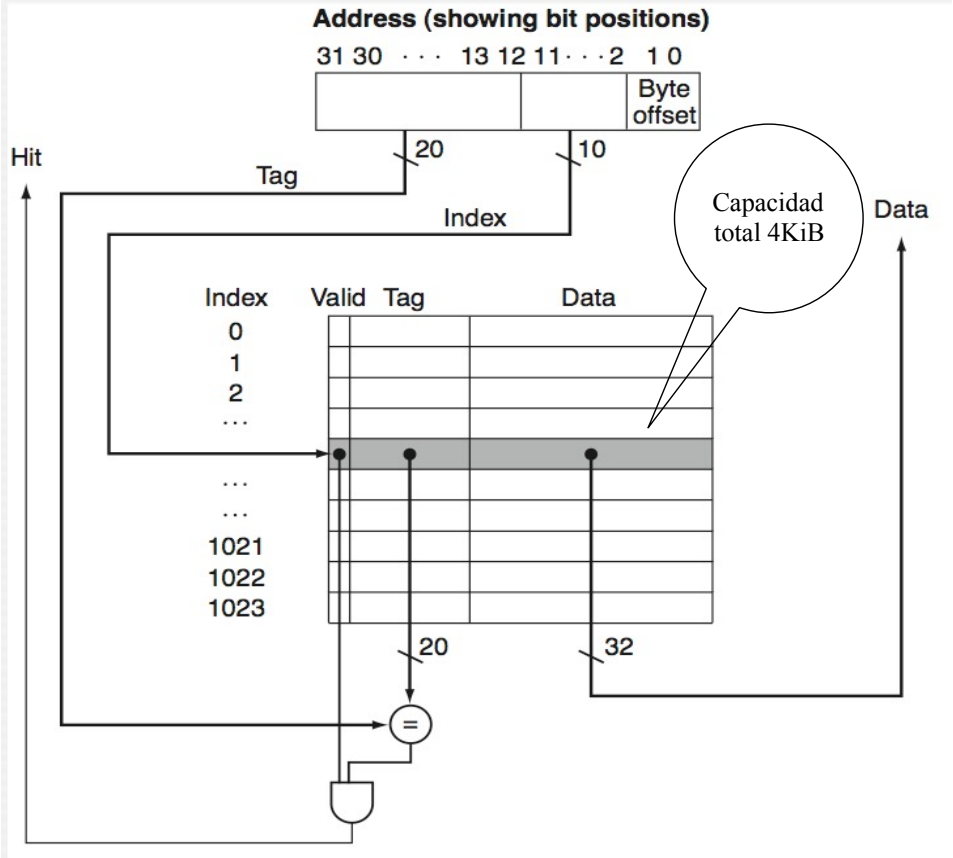
f. After handling a miss of address (10010_{two})

Campos en una referencia

- **Campo etiqueta** (tag), usado para comparar con el valor del campo tag de la cache.
- **Índice de cache**, usado para seleccionar un bloque.
- Tamaño del campo tag:
  - Direcciones de 32 bits.
  - Cache de correspondencia directa.
  - Tamaño de la cache es $\mathbf{2^n}$ bloques, **n** bits de índice
  - Tamaño de bloque $\mathbf{2^m}$ palabras (2^{m+2} bytes), **m** bits usados para la palabra dentro del bloque, dos bits usados para el byte.

$$32-(n+m+2)$$

Campos en una dirección



Ejemplo: bits en la cache

- ¿Cuántos bits requiere una cache de 16 KiB (16 X 2^{10} bytes) de datos y bloques de 4 palabras, con direcciones de 32 bits?

- Total de datos 4096 (2^{12}) palabras.
- Total de bloques 1024 (2^{10}).
- Bytes por bloque 4 X 32 =128 bits.
- Tag por bloque 32-10-2-2=18
- Vit de validez

$$2^{10} \text{ X } (4 \text{ X } 32 + (18) + 1) = 2^{10} \text{ X } 147 = 147 \text{ Kibibits}$$

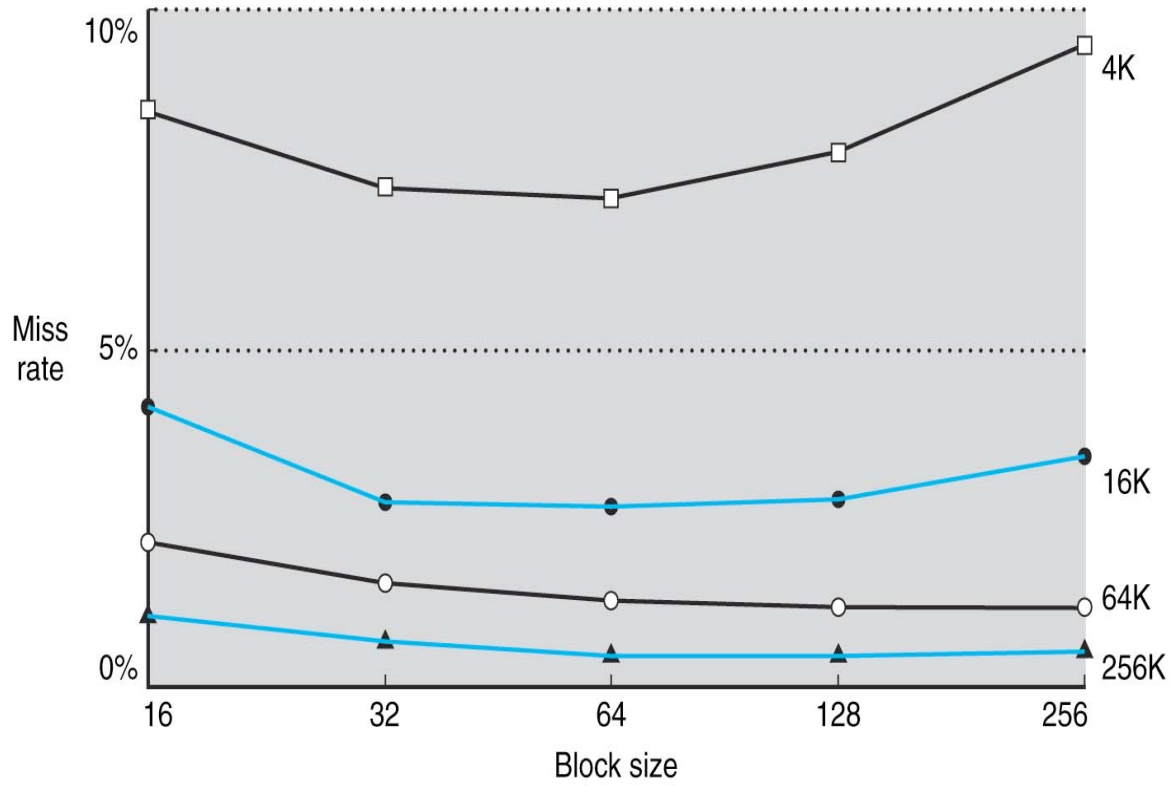
Ejemplo: correspondencia en multipalabra

- Considere una cache con 64 bolques y 16 bytes por bloque.
- ¿A qué bloque le corresponde el byte 1200?
- Bloque en cache=(Dir Bloque) modulo (Nro de bloques)

$$Dirección\ del\ bloque\ en\ memoria = \left[\frac{1200}{16}\right] = 75$$

- (75 modulo 64)=11

Tasa de fallos vs. tamaño del bloque



Manejo de los fallos de cache

Colaboración entre la unidad de control y
un controlador separado que inicia
la lectura de memoria y llena la cache.

Se atasca el procesador y se congelan los registros.

Si hay miss de instrucción el RI es inválido.
Solicitud de lectura al nivel inferior.
Acceso a la memoria en PC-4.

Hay una espera de varios ciclos y
luego se escriben las palabras en la cache.

Pasos ante un fallo de cache

Envío del PC original (NPC-4) a la memoria.

Solicitar a la memoria la lectura,

luego esperar a que se complete el acceso.

Escritura en la entrada de cache: copia del dato de memoria hacia la zona de datos, escribir el campo tag, poner el bit de validez en ON.

Recomenzar la ejecución de la instrucción desde el primer paso, ahora cargando la instrucción correcta.

Manejo de escrituras

- La copia en cache debe ser actualizada en memoria.
- Escritura directa (write through): la actualización se hace en cache y memoria.
- Escritura demorada (write back):se actualiza en cache y cuando se reemplaza el bloque se actualiza en memoria.

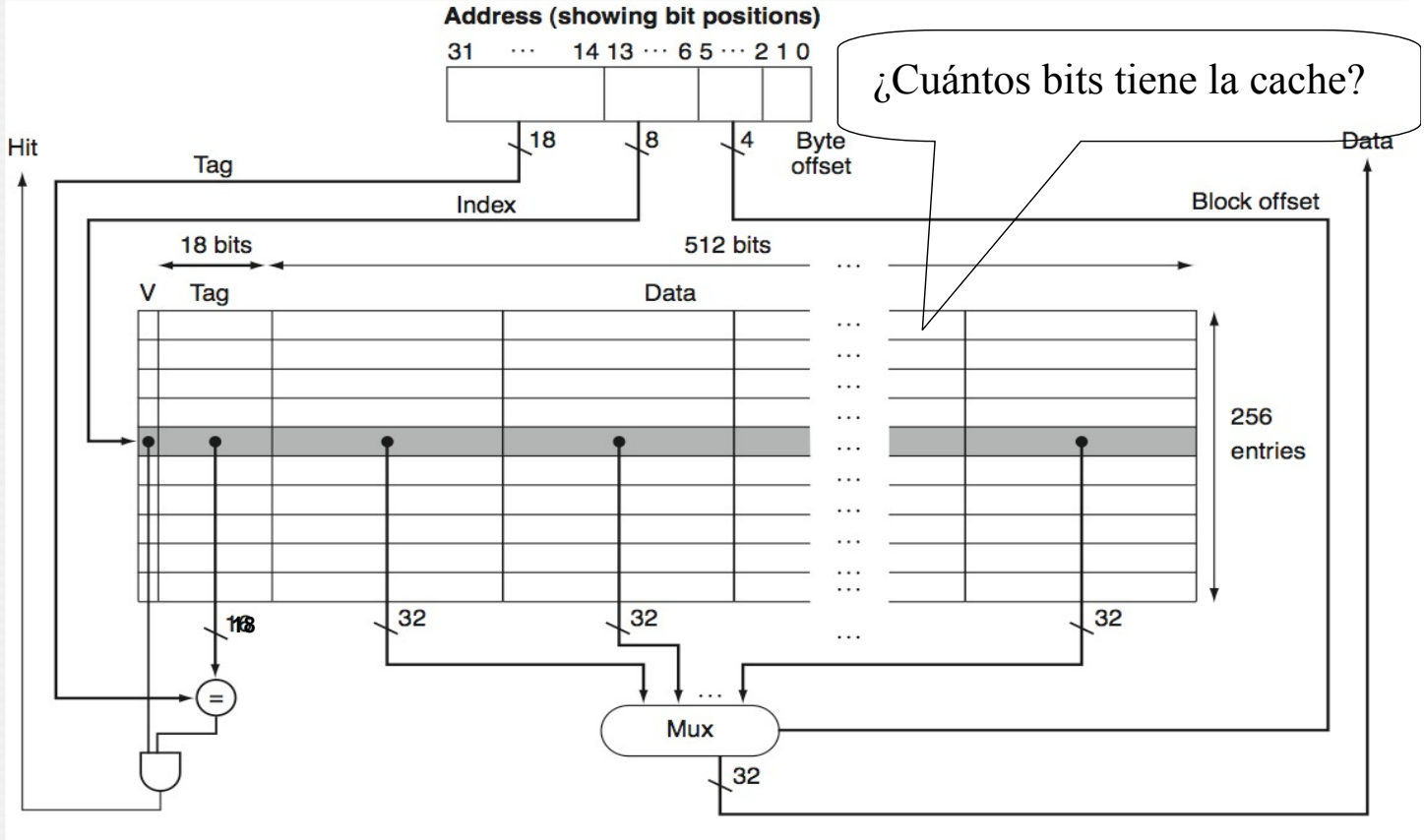
Buffer para Escritura Directa

- Un buffer mantiene los datos mientras aguardan ser escritos en memoria.
- El procesador continúa ejecutando luego de escribir el dato en cache y en el buffer de escritura.
- Si no hay lugar en el buffer es necesario esperar. Stall.
- La velocidad a la que se generan las escrituras no debe ser mayor que la velocidad de procesamiento de la memoria.

Complicaciones con las escrituras

- Ante un fallo ¿hay que cargar el bloque en cache?
  - Alojar para escribir (write-allocate)
  - No alojar para escribir (no write allocate)
- En write-back las escrituras:
  - Requieren un ciclo de comprobación y otro de escritura.
  - Usan el buffer de escritura y después copian en cache.
  - Pasan el bloque modificado al buffer, mientras se lee el nuevo.

Ejemplo de cache: procesador FastMATH



Las siguientes afirmaciones ¿Son siempre verdaderas?

- La jerarquía de memoria toma ventaja de la localidad temporal.
- En una lectura, el valor retornado depende de los bloques que se encuentren en cache.
- Casi todo el costo de la jerarquía de memoria está en el nivel más alto.
- Casi toda la capacidad de la jerarquía de memoria está en el nivel más bajo.

¿Qué vimos?

- Jerarquía de memoria.
- Localidad de referencia.
  - Localidad temporal
  - Localidad espacial
- Introducción a los conceptos de cache
- Ubicación de los bloques con correspondencia directa
- Cantidad total de bits de la cache
- Cache multi-palabra