

Cours 7

Sous Système Mémoire
ARCHI

25/10/23

Resumen arquitectura RISC

Esperar un $CPI = 1$

Aumentar el rendimiento con arquitectura **pipeline** al aumentar la frecuencia 

Superpipeline aumenta todavía más la frecuencia 

Entre más cortemos, más se degrada el CPI debido a las dependencias de datos (stall cycles)

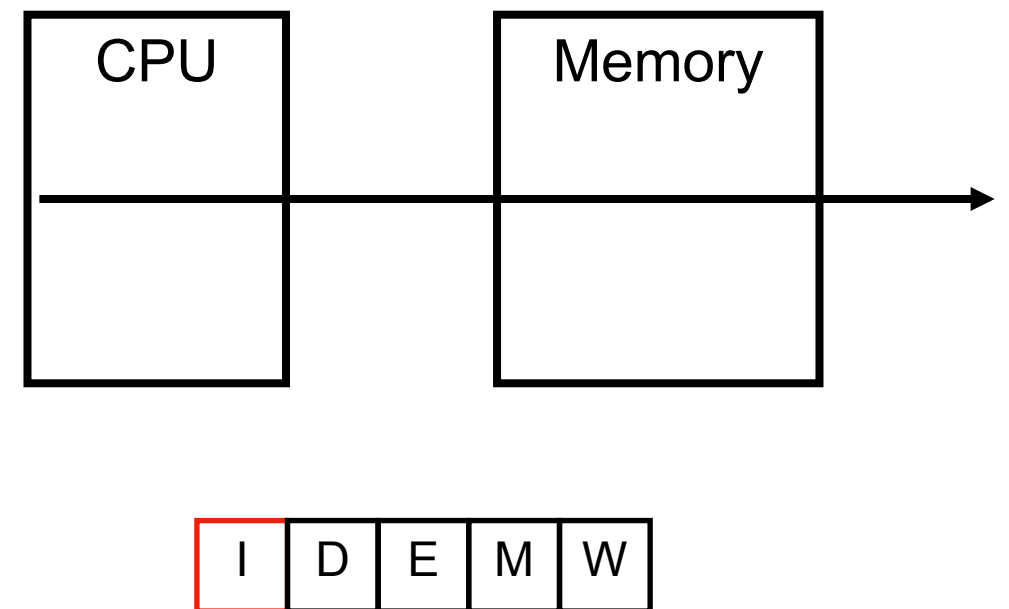
Superscalar es tener pipelines en paralelo

Ejecutar una instrucción en cada ciclo es ejecutar el IFC

En cada ciclo se llama a la memoria

No sólo se necesita un CPU eficaz, sino una **memoria eficaz** igualmente

Se requieren exigencias sobre la memoria



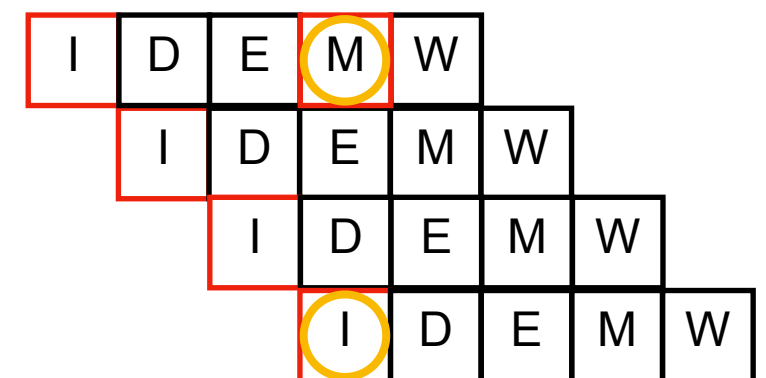
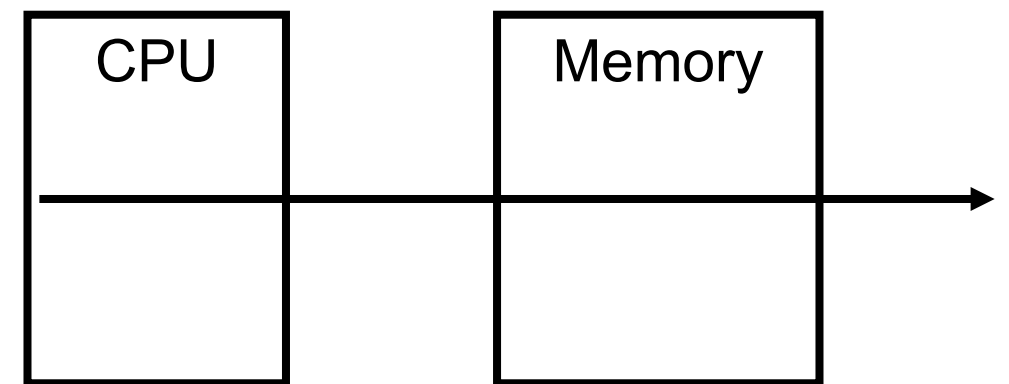
IFC accesa a la memoria

Ejecutar una instrucción en cada ciclo es ejecutar el IFC

En cada ciclo se llama a la memoria

No sólo se necesita un CPU eficaz, sino una **memoria eficaz** igualmente

Se requieren exigencias sobre la memoria



Dos accesos a la memoria al mismo tiempo ⚠

> 1 request par cycle

Load/Store -> 30%-40% des instructions

Fréquence du Processeur

Espace 4 GB par processus

Prix

Technologie Mémoire

Cómo podemos mejorar la tecnología de la memoria?

Fijamos un parámetro

Ej. **Precio \$**

Tiempo 1 ciclo, misma frecuencia que el procesador

Misma tecnología que el procesador

Static CMOS memory

Electronic (Silicium) drive

Static because there is an **active** process (e.g. transistors) maintaining the memory cell to '0' or '1'

Small memory capacity ↘

Some KB to MB (not that much)

Fast access times ↗ ↗

(1 cycle)

Expensive \$ \$ \$



SRAM
chip

Dynamic memory

Electronic (Silicium) drive

Dynamic because there is an **passive** process (e.g. capacitors) maintaining the memory cell to '0' or '1'

More memory capacity ↗

Some MB to GB

Slow access times ↘

(10 cycles, 10 times slower)

Less expensive \$ \$



DRAM
chip

Hard Disk Drive (HDD)

Mechanic drive

There is an **passive** process, in this case **magnetism** maintaining the memory cell to '0' or '1'

Much more memory capacity ↗ ↗

Some GB to TB

Slower access times ↘ ↘

(10^6 cycles, 10^6 times slower)

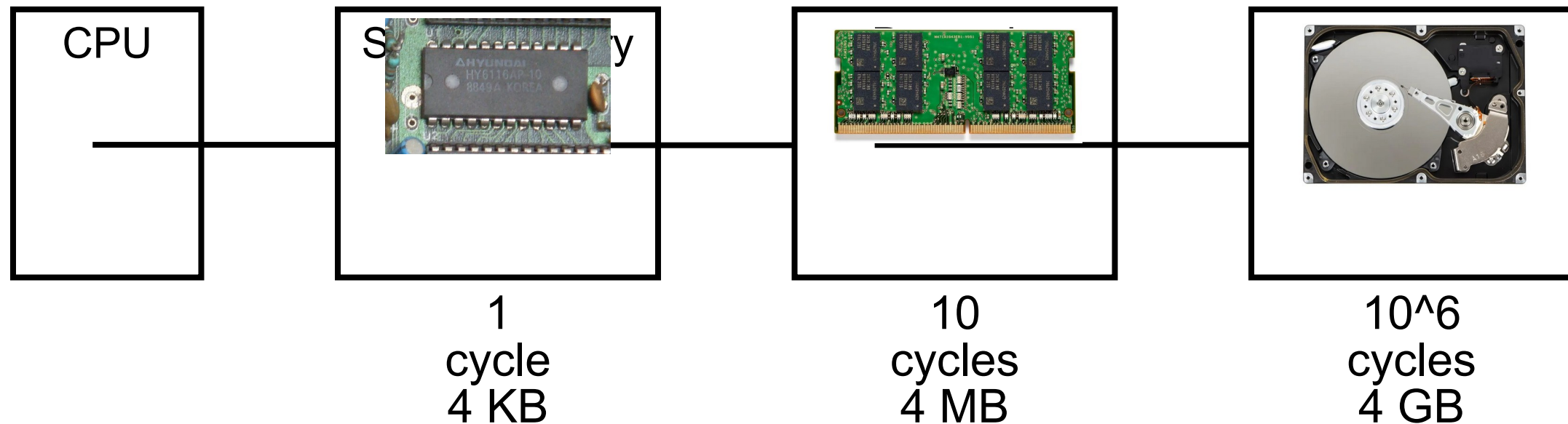
Much less expensive 💰



HDD

Hiérarchie de mémoire

Combining different memory technologies



Podemos **combinar diferentes tecnologías** de memoria que se adapten a nuestras necesidades.

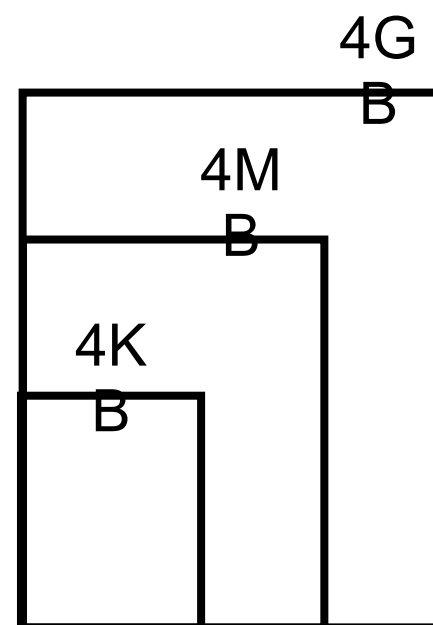
Memoria rápida para cubrir datos inmediatos y escasos

Memoria lenta para cubrir datos que necesiten ser guardados por mucho tiempo y que no se requieran acceder tan frecuentemente

Nombre moyen de cycles par access :

[OBJ]

[OBJ]



Ejemplo: probabilidad que un rayo caiga en mi, luego en mi coche, luego afuera, etc.

Probabilidad

Ejemplo supermercado

Todas las semanas voy al super y compro un chocolate

La semana que sigue voy al super y compro el mismo chocolate

Cómo es posible que siempre haya el mismo chocolate?

Porque el manager del super me observó con la carte de fidélité

Dijo: “La próxima vez que pase voy a stocker de chocolat”, porque sabe que yo lo voy a comprar



Fig
1

Localité spatiale

Espace d'adressage

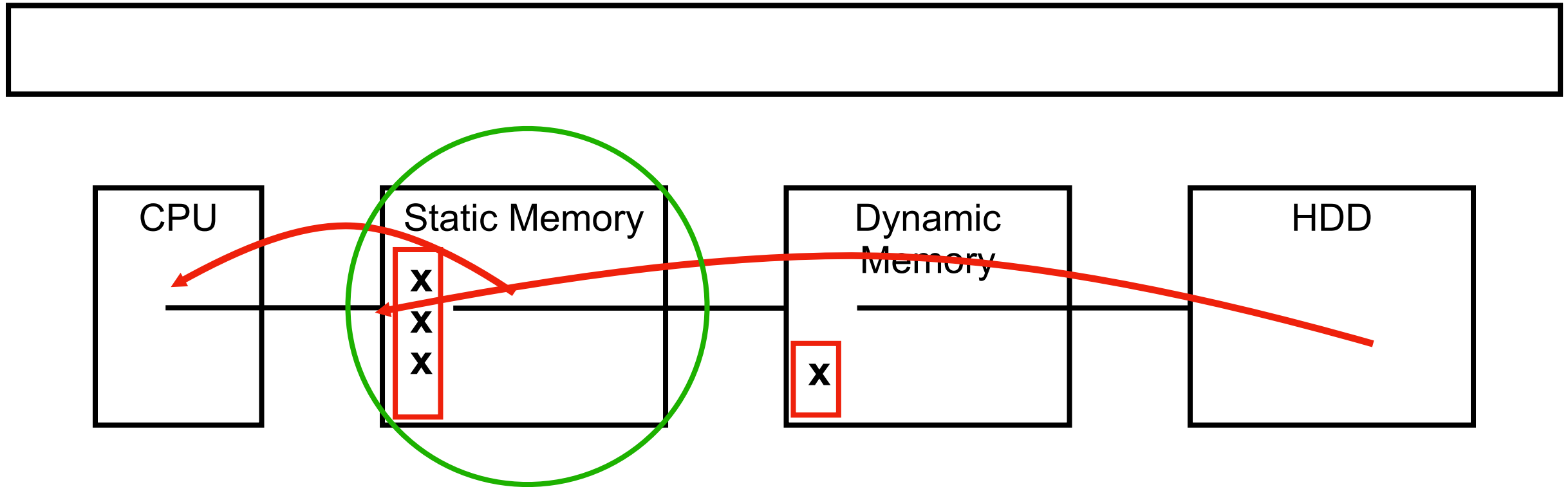
Il y a une forte probabilité que le prochain @ soit autour du dernière @ accédé

Más o menos 90% - 95% porque el compilador reserva memoria alrededor de la misma zona de memoria

Las variables locales son guardadas en el stack (mismo espacio memoria)

Las variables globales están en otra zona

Las variables dinámicas (alocación de memoria) se guardan en otra zona



Supongamos que el dato **x** esté en la memoria dinámica, pasa a la memoria estática y después al procesador.

La localidad espacial me dice que el acceso memoria estará alrededor de la memoria estática.

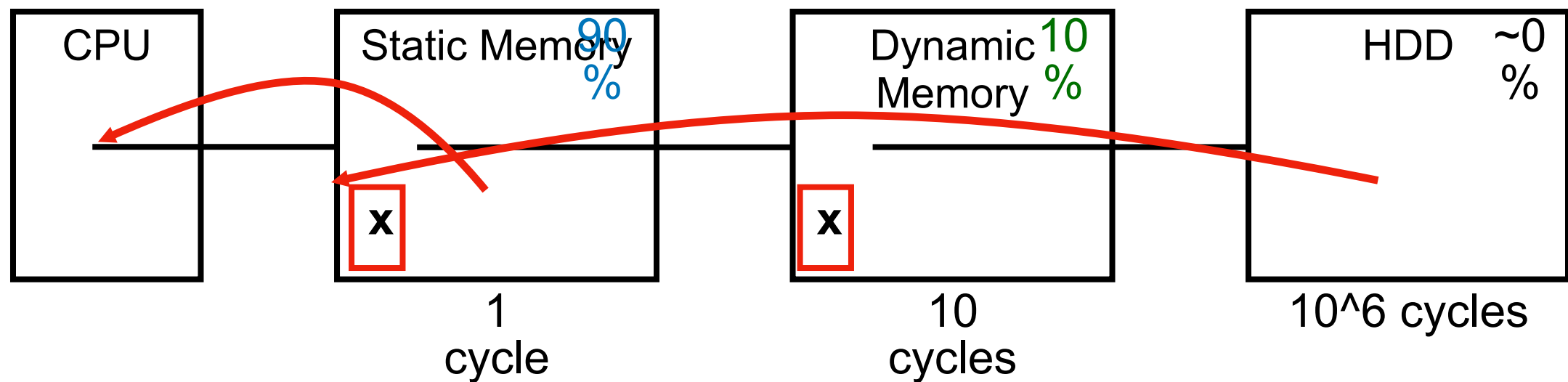
Ej. en lugar de tomar un chocolate solamente en la tienda, tomo un paquete de chocolates

Este bloque se llama **bloc mémoire**, en este caso son 8 words = 32 bits

90% de probabilité que
l'instruction soit dans la
mémoire statique

10% de probabilité que
l'instruction soit dans la
mémoire dynamique


$$= 1 \text{ cycle} * 0.9 + 10 \text{ cycles} * 0.1 = 1.9 \text{ cycle}$$

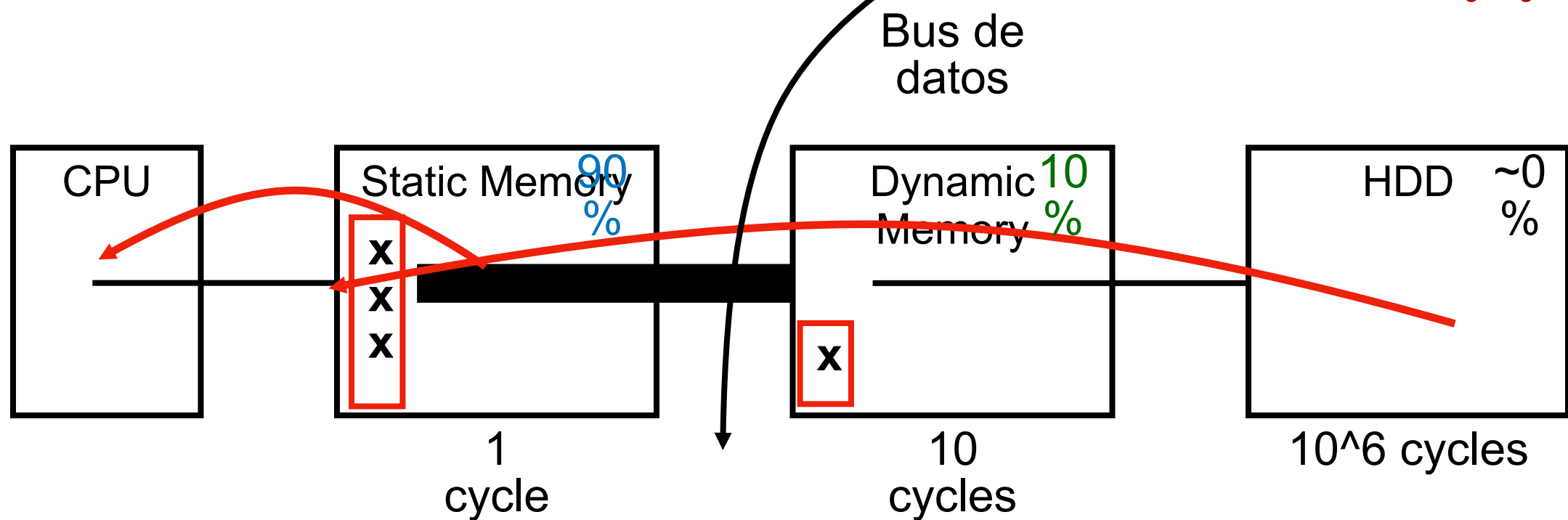


90% de probabilité que
l'instruction soit dans la
mémoire statique

10% de probabilité que
l'instruction soit dans la
mémoire dynamique

$$= 1 \text{ cycle} * 0.9 + 10 \text{ cycles} * 0.1 = 1.9 \text{ cycle}$$

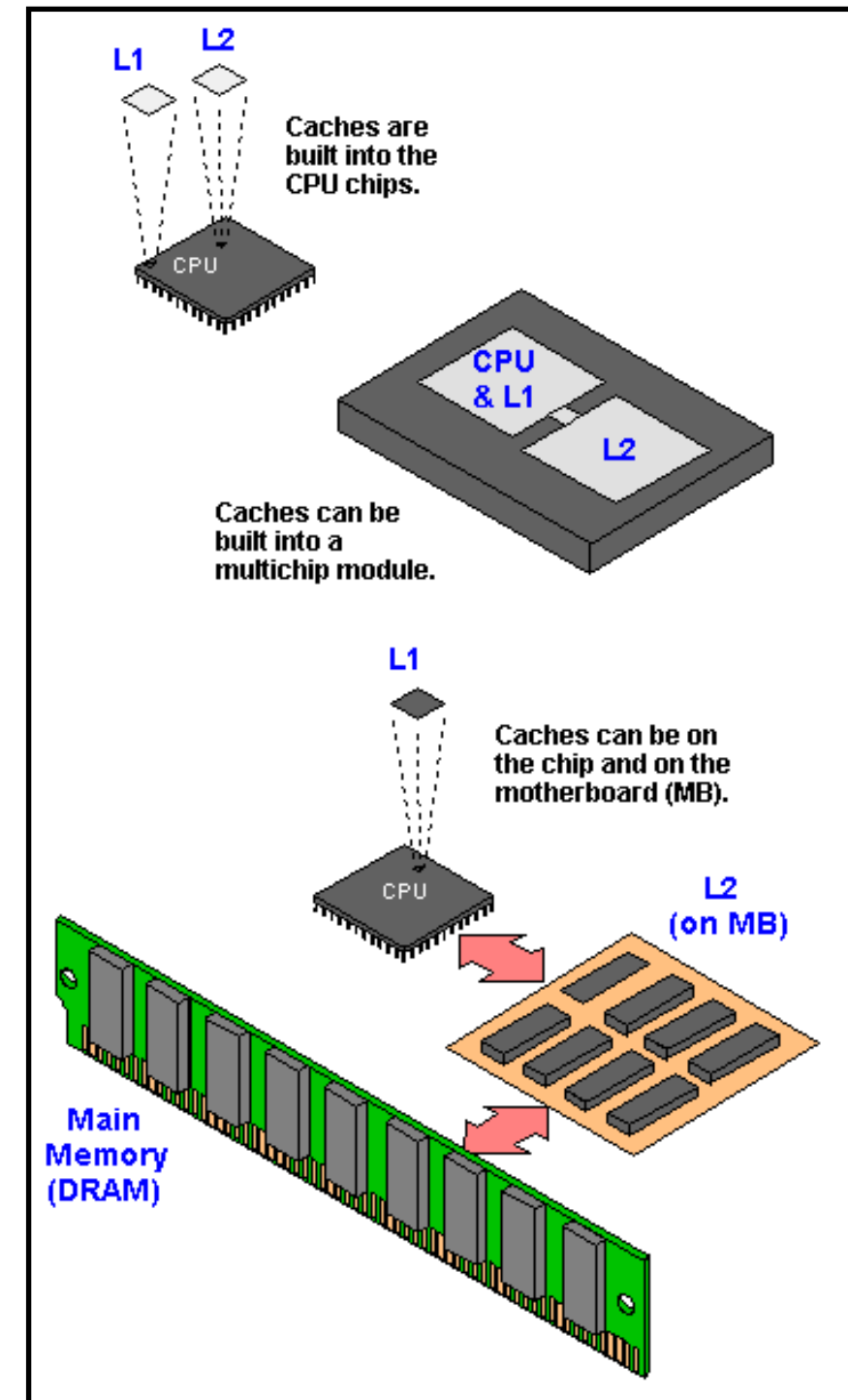
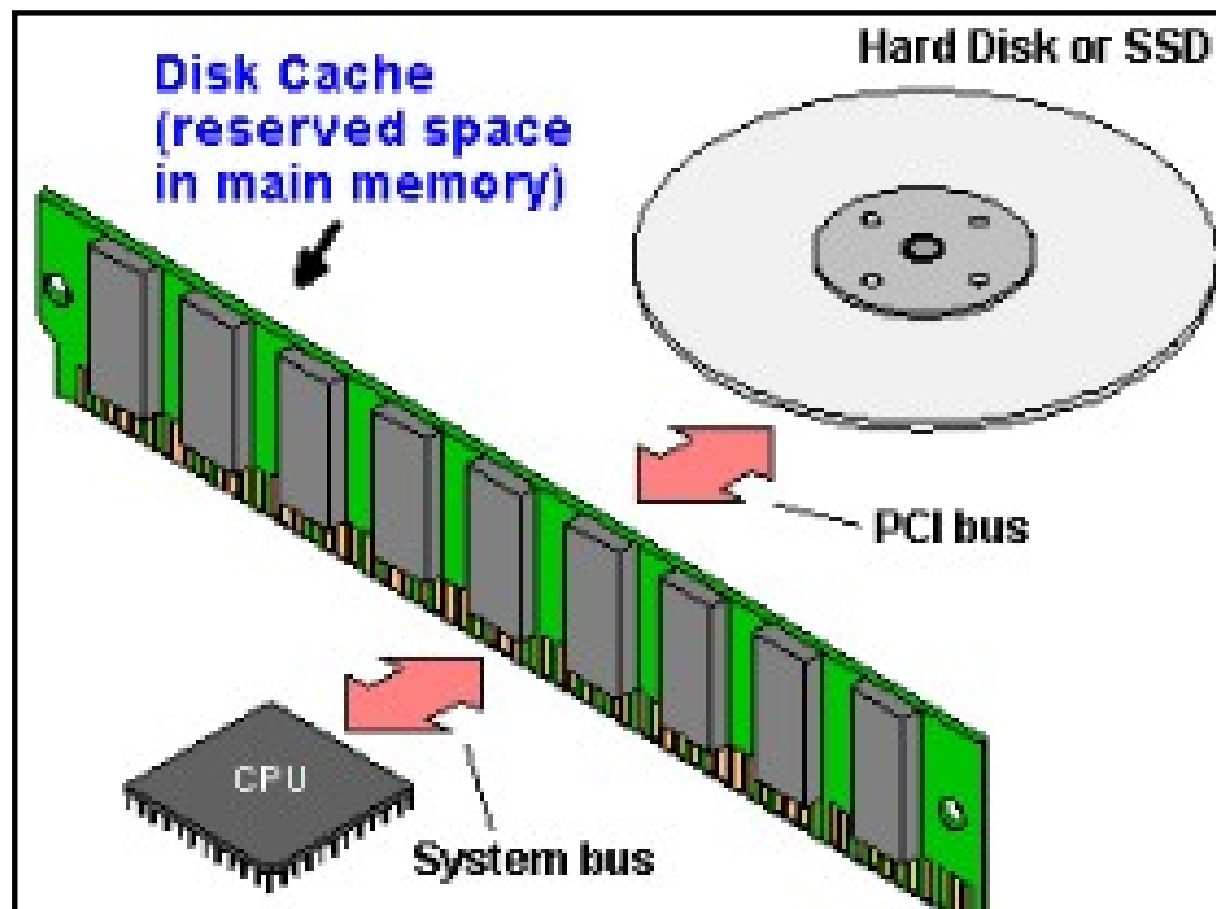
Faux 



$$= 1 \text{ cycle} * 0.9 + 8 * 10 \text{ cycles} * 0.1 = 8.9 \text{ cycle}$$

bloc de 32 bits = 8 mots

Bus de données

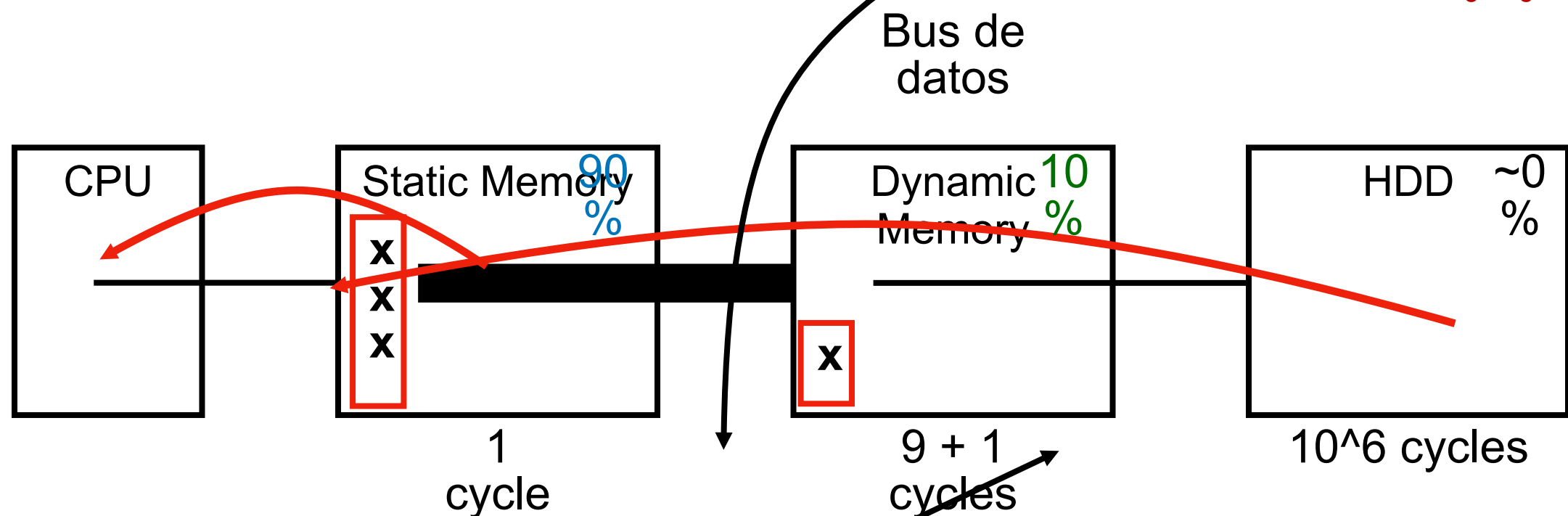


90% de probabilité que
l'instruction soit dans la
mémoire statique

10% de probabilité que
l'instruction soit dans la
mémoire dynamique

$$= 1 \text{ cycle} * 0.9 + 10 \text{ cycles} * 0.1 = 1.9 \text{ cycle}$$

Faux



En realidad se necesitan 9 ciclos para acceder al bus
de datos, + 1 ciclo para transferir los datos

$$= 1 \text{ cycle} * 0.9 + 8 * 10 \text{ cycles} * 0.1 = 8.9 \text{ cycle}$$

Faux

bloc de 32 bits = 8 mots

$$= 1 \text{ cycle} * 0.9 + 10 \text{ cycles} * 0.1 = 1.9 \text{ cycle}$$

Faux



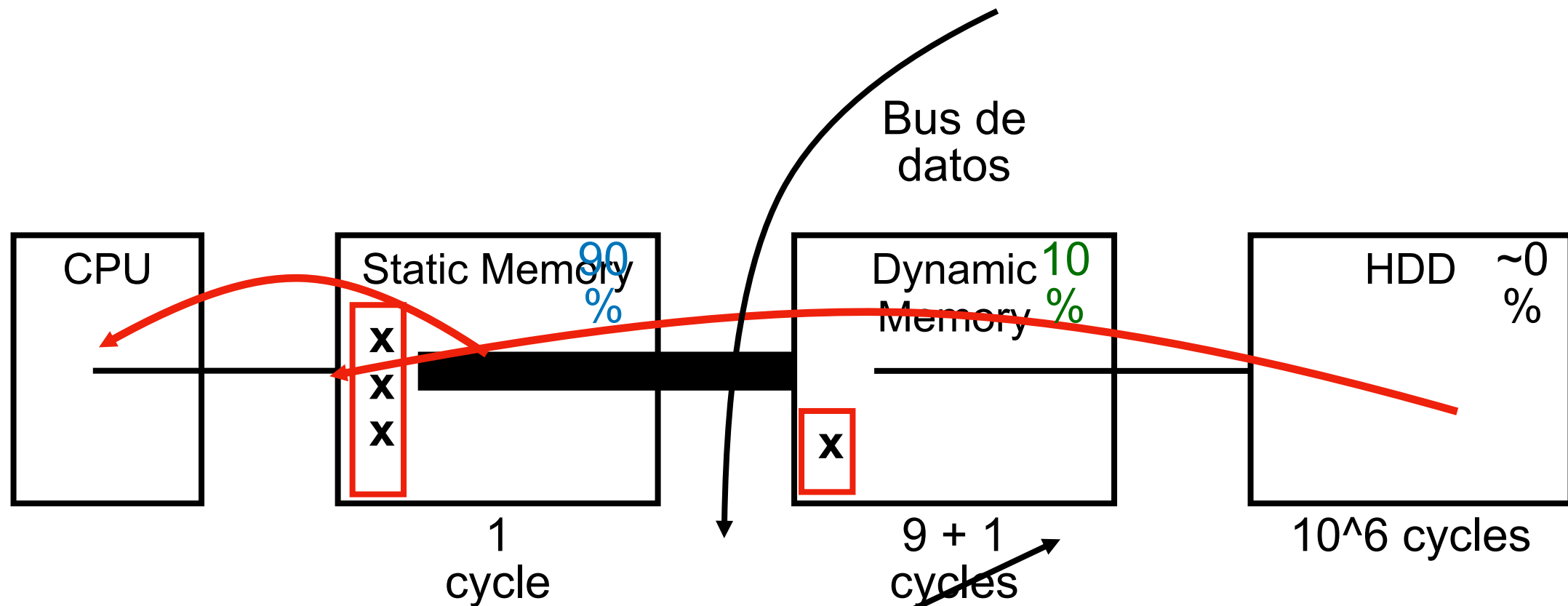
$$= 1 \text{ cycle} * 0.9 + 8 * 10 \text{ cycles} * 0.1 = 8.9 \text{ cycle}$$

Faux



$$= 1 \text{ cycle} * 0.9 + (9 \text{ cycles} + 8 \text{ mots} * 1) * 0.1 = 2.6 \text{ cycle}$$

Vrai



En realidad se necesitan 9 ciclos para acceder al bus de datos, + 1 ciclo para transferir los datos

Analogía

Quieres tirar pelotas en un tubo que no es grande, entonces no salen tan rapido porque están limitados por el tamaño del tubo.

Si queremos que salgan más rápido, se tiene que agrandar el tamaño del tubo.

En nuestro ejemplo, las pelotas son los datos. El tubo es el data bus

1. Temps pour acceder au bus
2. Nombre de cycles pour acceder au bus ne depends pas de la taille
3. le temps de transferer depend de la taille du bloc, es directement proporcional