


# Arquitectura de Computadores

The background of the slide is a collage of digital and technological motifs. On the left, a blue-tinted profile of a human head is formed by a complex network of circuit lines and binary digits (0s and 1s). The right side of the slide features a dark blue background with glowing white circuit traces and bright, starburst-like light effects. In the top right corner, there is a faint, stylized illustration of a neural network or a complex geometric structure.

2021 A

Andrés Larco





# MEMORIA CACHE

6.5 Medidas de Rendimiento de la  
cache

# Principios de funcionamiento

- El papel de la memoria caché está fundamentado en el denominado principio de **localidad**:

*“Todo aquello que se utiliza en un momento dado nos da una pista sobre lo que se va a utilizar en el futuro”*

- **Localidad Temporal:** Si un dato es utilizado por el procesador, es altamente probable que ese dato se vuelva a utilizar.
- **Localidad Espacial:** Si un dato es utilizado por el procesador, es altamente probable que se utilicen otros datos en las posiciones de memoria **cercanas**.
- Estos dos principios permiten identificar con una probabilidad alta el **conjunto de datos** con el que el procesador trabajará en el futuro.

# Interacción CPU-Caché

## Uso del principio de localidad temporal

- La memoria cache y el procesador **interaccionan** de la siguiente forma (se asume que, al comenzar, la memoria caché está vacía):
  - ✓ El procesador accede a memoria principal.
  - ✓ La memoria principal proporciona el dato, pero se almacena también en la caché.
  - ✓ El procesador accede de nuevo al mismo dato.
  - ✓ La memoria caché detecta este acceso y proporciona el dato. Por tanto se evita el acceso a la memoria principal.
- Esta mejora está basada en el principio de **localidad temporal**.

# Interacción CPU-Caché

## Uso del principio de localidad temporal ....

- Esta mejora contribuye a aumentar el **rendimiento** del procesador.
- Supongamos que el micro accede 10 **veces a un dato**, que el acceso a la memoria principal es de 100 ns, y que el acceso a la caché son 10 ns.
- **Sin Caché:** 10 accesos a 100 ns significan 1000 ns que el procesador dedica a esta transferencia de datos.
- **Con Caché:** El primer acceso tarda 100 ns pero los siguientes 9 se realizan a la cache, por tanto tardamos en total 190 ns, o lo que es lo mismo un 19% del tiempo empleado sin la cache.

# Interacción CPU-Caché

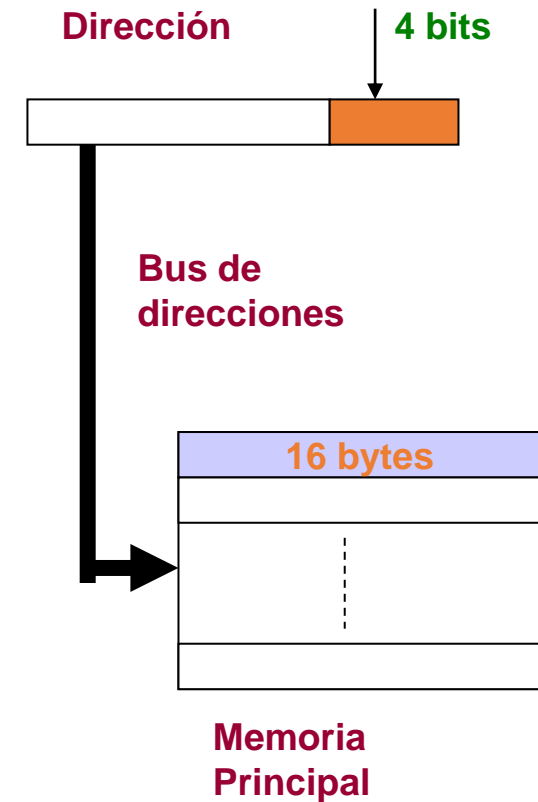
## Uso del principio de localidad espacial

- Lo anterior puede mejorarse con **localidad espacial**, es decir si se utiliza un dato es altamente probable que se utilicen los datos adyacentes en memoria.
- De memoria se trae un conjunto de datos consecutivos en lugar de un único dato.
- **¡PROBLEMA!**
- **¿Por qué?** Para traer cada dato contiguo se tiene que emplear un tiempo de acceso.

# Interacción CPU-Caché

## Uso del principio de localidad espacial ..

- **¡SOLUCIÓN!** Se modifica la estructura de la memoria para que sus operaciones de lectura y escritura se hagan en bloques de cierto tamaño.
- **Ejemplo:** Supongamos que la transferencia de memoria se hace en bloques de 16 bytes.
- Cuando se pide datos con una dirección de memoria se proporcionan los 16 bytes consecutivos (se necesiten o no).



# Interacción CPU-Caché

## Uso del principio de localidad espacial ...

- Lo anterior prácticamente implica que puedan combinarse el principio de localidad temporal y espacial .
- ¿Por qué?

### Ejemplo de acceso:

- Pensemos que se utiliza la estructura de memoria del ejm anterior (100 ns de acceso a memoria principal y 10 ns de acceso a la caché).
- Y un programa accede a 8 datos de 4 bytes consecutivos en memoria, cada uno de ellos 10 veces.
- La secuencia de accesos es:

**Primera vez:** dato1, dato2, ..., dato 8,

**Segunda vez:** dato1, dato2, ..., dato 8,

...

**Décima vez:** dato1, dato2, ..., dato 8



# Interacción CPU-Caché

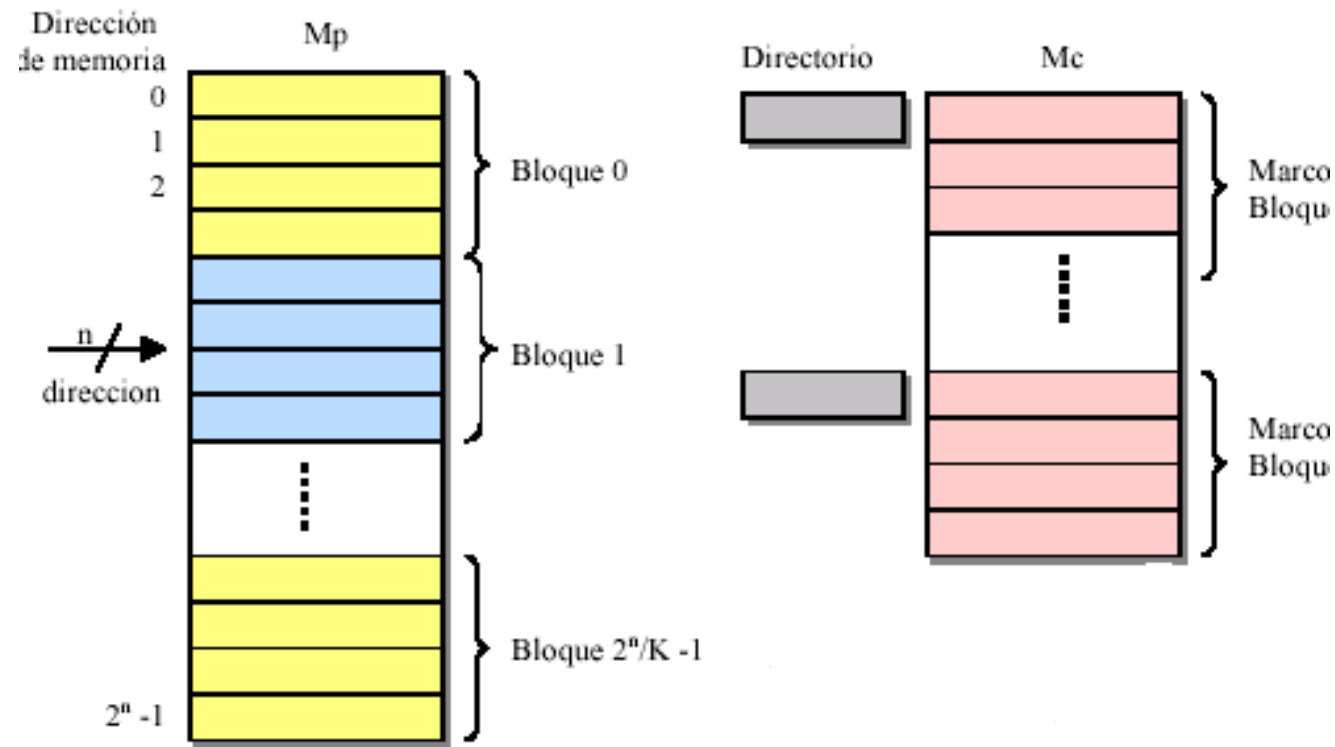
## Uso del principio de localidad espacial ...

- **Sin Caché:** 8x10 accesos a 100 ns cada uno totalizan un retardo de 8000 ns (cada acceso es a un solo dato).
- **Con Caché:**
  - El primer acceso tarda 100 ns pero proporciona 4 datos (16 bytes).
  - Los siguientes accesos hasta el cuarto dato están ya en la caché, 10 ns cada uno.
  - El quinto dato tarda 100 ns porque no está en la caché.
  - De ahí en adelante se tarda 10 ns para cada acceso.
  - En total se tardan 980 ns, o 12,25% del tiempo sin caché (el procesador va 8.16 veces más rápido).

# Estructura de la Caché

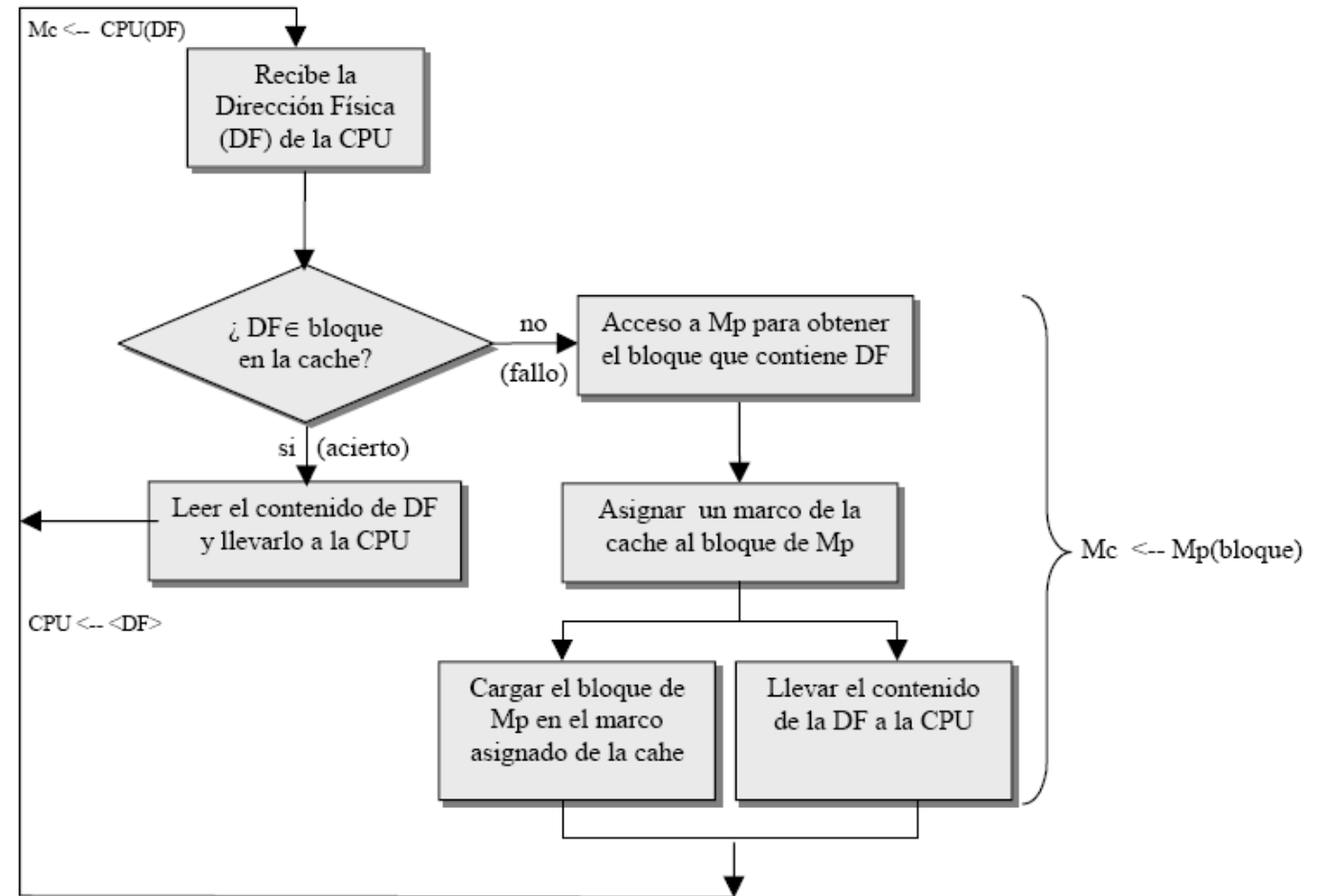
- Para implementar el mecanismo de actualización de la caché con los datos *que pueden ser referenciados* con mayor probabilidad se utilizan políticas de estructuración tanto de la memoria caché como de la main memory.
- A la main memory se lo divide en bloques de cierto número de bytes (4, 8, 16, etc.).
- Y a la caché en los denominados marcos de bloque de igual tamaño, un marco puede albergar varios bloques de la main memory.
- Un bloque de memoria principal se almacena en un **único bloque** en la memoria cache.
- El directorio contiene la información de los bloques de la m.m. que se encuentran ubicados en la caché.

# Estructura de la Caché



- $m$ : número de marcos de bloque
- $n$ : ancho del bus de direcciones
- $K$ : número de bytes por bloque

# Funcionamiento del sistema



# Funcionamiento del sistema

- Bajo este esquema de funcionamiento pueden presentarse 2 casos:
  - ✓ Acierto (hit): el dato es encontrado en la dirección denotada
  - ✓ Fallo (miss): dato no es encontrado en la dirección denotada

## Medidas de Rendimiento

- Se supone:
  - N: número de accesos a la caché
  - $N_h$ : número de aciertos
  - $N_m$ : número de fallos



# Funcionamiento del sistema

## Medidas de Rendimiento ...

- Se puede calcular los siguientes parámetros:
  - ✓ el número total de accesos ( **$N$** )
  - ✓ la tasa de aciertos ( **$\tau_h$** )
  - ✓ la tasa de Fallos ( **$\tau_m$** )
  - ✓ la relación entre las tasas

$$N = N_h + N_m \quad (1)$$

$$\tau_h = \frac{N_h}{N} \quad (2)$$

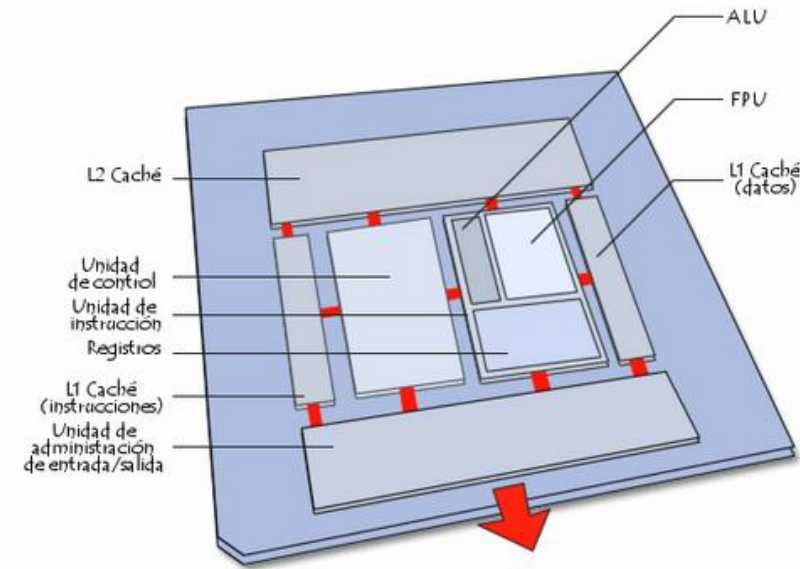
$$\tau_m = \frac{N_m}{N} \quad (3)$$

$$\tau_h = 1 - \tau_m \quad (4)$$

# Datos

## Valores Típicos

- L1 o interna: La caché de primer nivel contiene muy pocos kilobytes (unos 32, 64, 128 o 256 Kb).
- L2 o externa: Los tamaños típicos de la memoria caché L2 oscilan en la actualidad entre 256 kb y 8 Mb.



# Datos

- Valores Típicos

Parámetro	Valor típico
Tamaño del bloque	4-128 bytes
Tiempo de acierto	1-4 ciclos
Penalidad de desacierto	8-32 ciclos
Tiempo de acceso	6-10 ciclos
Tiempo de transferencia	2-22 ciclos
Razón de desacierto	0,01-0,2
Tamaño del cache	1Kbyte - 256Kbyte

# Datos

- Valores Típicos

## Terminología

Bloque : unidad mínima de almacenamiento en cache

Acierto : palabra buscada pertenece a bloque presente en cache

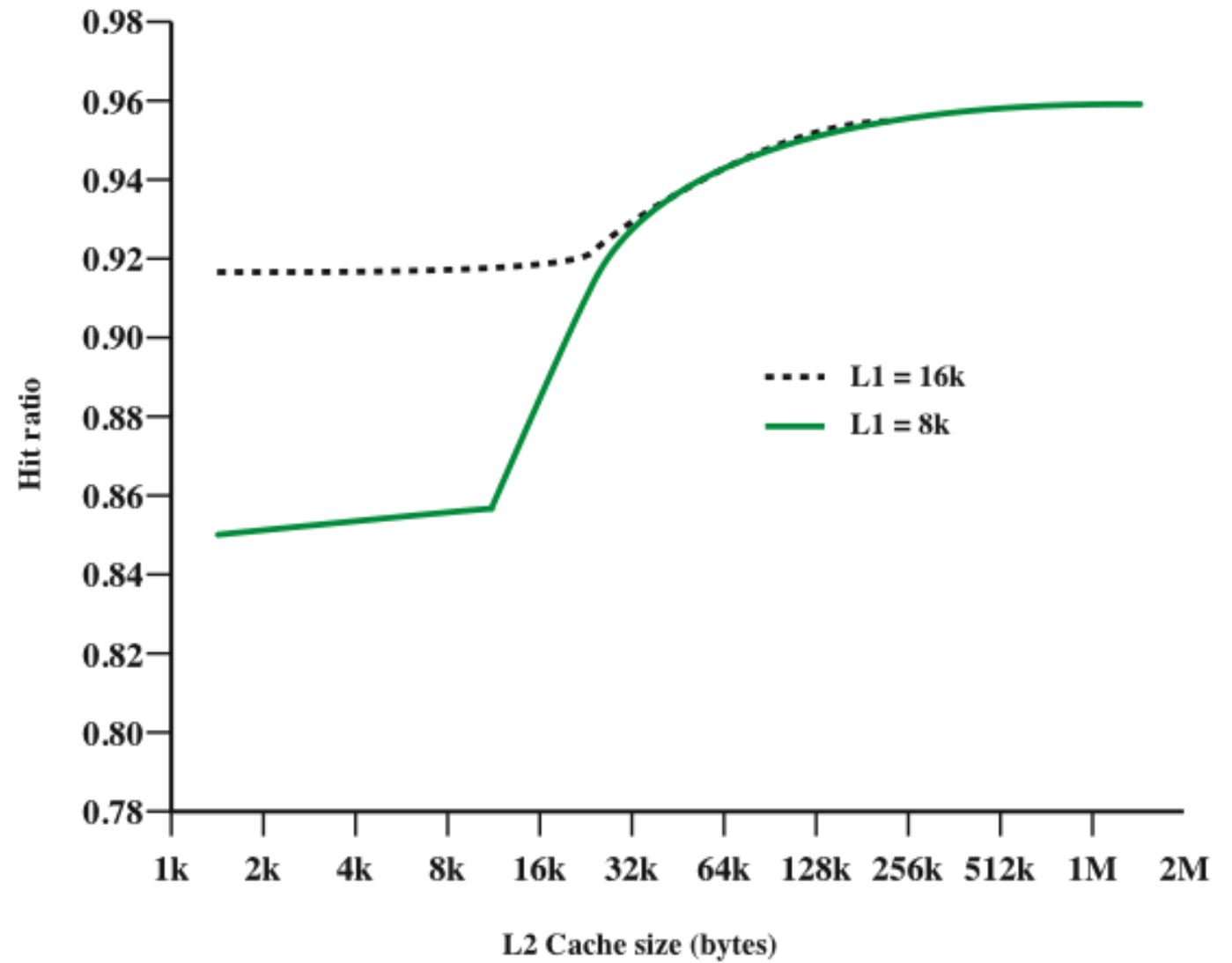
Desacierto : palabra buscada pertenece a bloque ausente de cache

Razón de acierto : fracción de referencias a memoria que producen aciertos

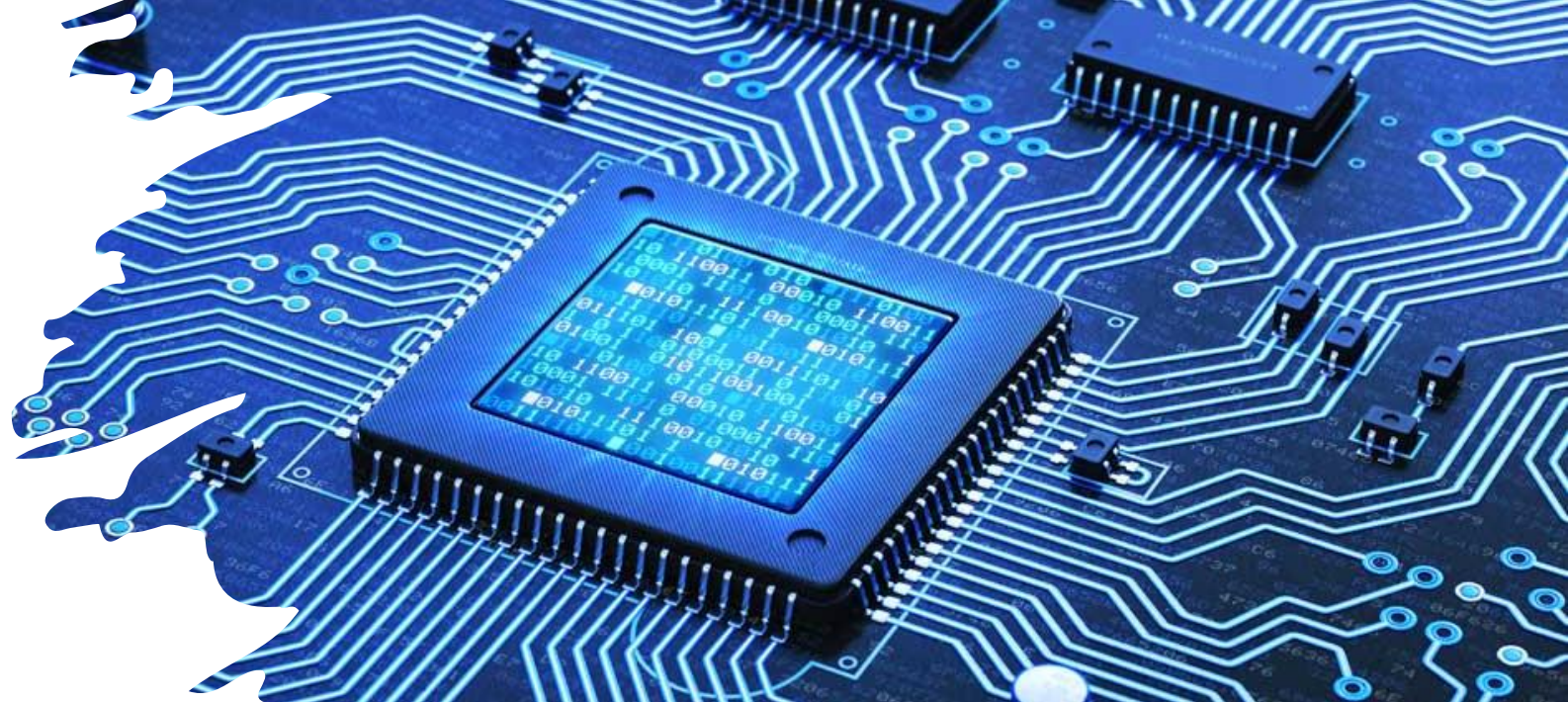
Razón de desacierto :  $1 - (\text{razón de acierto})$

Tiempo de acierto : tiempo en leer un dato del cache

Radio de  
aciertos (L1 &  
L2)  
Para 8 Kbyte y  
16 Kbyte L1







Gracias !!!

