

Práctica 1. Repaso de conceptos relacionados con jerarquía de memoria.

Grado en Ingeniería Informática
Universidade de Santiago de Compostela

Arquitectura de computadores. Práctica 1.

citus.usc.es

Práctica 1

Es una de las dos prácticas que habrá que entregar de manera obligatoria.

- Puntúa un 40 % de la nota total de prácticas. (2 puntos sobre los 10 de la nota final de la asignatura).
- Realización en equipos de 2 personas. Se proporcionará una cuenta del CESGA por equipo.
- No es obligatorio asistir a las sesiones de prácticas pero si asistes participa.
- Se entrega una memoria junto con el código (ver enunciado).

Práctica 1

Dada una estructura de datos queremos comprobar que dos patrones de acceso a los datos con diferente localidad consumen un tiempo de acceso a memoria diferente.

- Programa en lenguaje C y como SO Linux en el FinisTerra III del CESGA.
- Compilamos con gcc opción -O0 para evitar que el compilador realice optimizaciones (reestructurar código, precargar datos, evitar hacer operaciones,...).
- Medimos ciclos de reloj por cada acceso a datos realizado en el código e interpretamos resultados.
- Interpretamos resultados basándonos en la localidad de los accesos y la arquitectura de cachés del procesador (IMPORTANTE).

Práctica 1. Uso del CESGA - I



Acceder a través de la web <https://portalusuarios.cesga.es> utilizando los siguientes datos:

- Username: cursoaNN
- Password: a.d.C.NN

Siendo NN el número asociado a cada equipo.

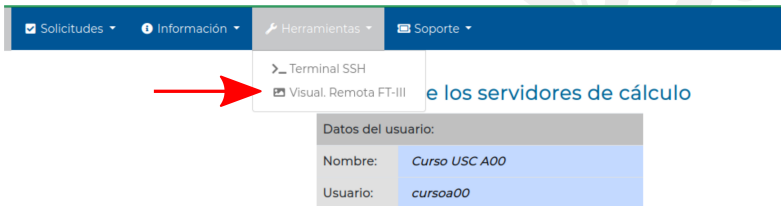
La primera vez que abráis un terminal os solicitará cambiar la contraseña. SI OLVIDÁIS LA CONTRASEÑA PERDERÉIS EL ACCESO.

Práctica 1. Uso del CESGA - II



Como conectarse:

- Conexión mediante escritorio remoto (desde cualquier red)



- Conexión por ssh (sólo desde eduroam):
`ssh cursoaNN@ft3.cesga.es`

Práctica 1. Instrucciones para ejecución en el CESGA - I



Existen dos opciones para ejecutar el código: sbatch o compute.

- Para la primera opción debemos ejecutar el comando:
`sbatch script_experimentos.sh`
Esto generará un archivo de salida: `slurm-job_id.out`
Para ver el estado de mi ejecución podemos usar el comando
'squeue'
- Para la segunda opción pediremos un nodo de computación con el comando `'compute -c 64 --mem 64G'` y después ejecutaremos `./script_experimentos.sh`

Práctica 1. Instrucciones para ejecución en el CESGA - II

Listing 1: Script Bash para la ejecución de los experimentos

```
#!/bin/bash
# Solicitamos un nodo con 64 cores y 256 GB de memoria durante 2 horas
#SBATCH -n 1 -c 64 -t 02:00:00 --mem=256G
# Ponemos nombre a nuestro trabajo para poder identificarlo.
# ATENCIÓN - Debes sustituir el NN por el número de equipo.
#SBATCH --job-name placgNN

# Sustituir los valores de Di y Li por los calculados para la realización de la práctica.

gcc acp1.c -o acp1 -msse3 -O0

for i in {1..10}
do
    for D in {D1,D2,D3,D4,D5}
    do
        for L in {L1,L2,L3,L4,L5,L6,L7}
        do
            ./acp1 $D $L
        done
    done
done
```

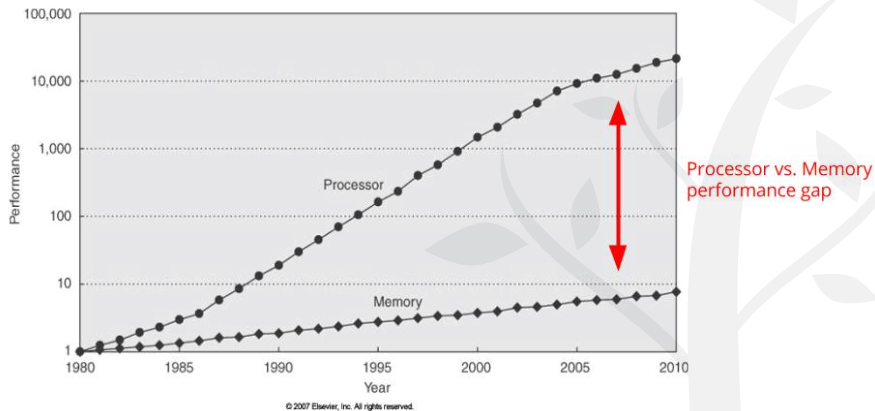
Practica 1. PAUTAS DE EVALUACIÓN

Condición para aceptar la entrega: **que el código sea compilable con el script.**

- Estructura y legibilidad del informe: **hasta 3 puntos.**
- Calidad y legibilidad del código entregado. **hasta 1 punto.**
- Que se respeten las especificaciones del enunciado. Elección correcta de los parámetros. **hasta 1 punto.**
- Experimentos adecuados, bien presentados y explicados: **hasta 5 puntos.**
 - ▷ Versión base: **hasta 2,5 puntos.**
 - ▷ Experimentos con tipo de dato int: **hasta 1,25 puntos.**
 - ▷ Comparar accesos indirectos con accesos directos: **hasta 1,25 puntos.**

El gap con la CPU

Todos los conceptos presentados a continuación fueron estudiados en el Tema 6 de Fundamentos de Computadores.



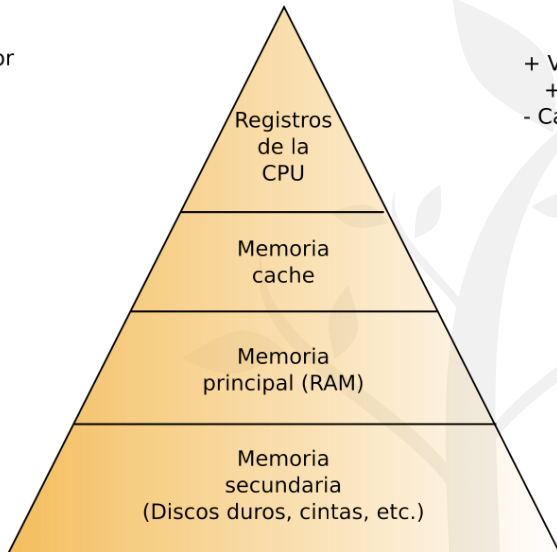
La jerarquía de memoria

Registros del procesador
Hasta 2 KiB
Hasta 57 GB/seg

Caches
32 KiB - 4 MiB
25 - 45 GB/seg

Memoria Principal
> 4 GiB
> 5 GB/seg

Disco duro
Miles de GiB
Hasta 150 MB/seg
Cintas de back-up
Varios TiB, 30 MB/seg



+ Velocidad
+ Coste
- Capacidad



Tiempos de acceso

Supongamos un ciclo de reloj de 0,3 ns (frecuencia 3,3 GHz)

Tecnología	Tiempo típico	Tiempo equivalente
Registro	0,3 ns	1 s
Cache L1	0,9 ns	3 s
Cache L2	2,8 ns	9 s
Memoria	60 ns	3,3 min
Disco SD	50 μ s	2 días
Disco magnético	5–20 ms	6–24 meses

Gestión de la jerarquía de memoria

¿Quién gestiona la jerarquía de memoria?

- Procesador: los niveles superiores (registros y cache)
- Sistema operativo: los niveles inferiores (memoria principal y disco)

La eficiencia en la gestión se basa en el **principio de localidad**

El principio de localidad

En programas reales se accede en cada intervalo de tiempo a un conjunto pequeño de direcciones.

Tipos de localidad:

- **Temporal**: si usamos un dato (o instrucción) seguramente lo volveremos a usar en breve
- **Espacial**: si usamos un dato o (instrucción) seguramente usaremos pronto otro situado en una dirección de memoria próxima
- **Secuencial** (caso particular de la localidad espacial): si usamos un dato (o instrucción) seguramente usaremos pronto el siguiente en memoria

Ejemplo de localidad

PROGRAMA

```
for(i=0; i<1000; i++) {  
    b[i] = d[i] * e[i];  
    a[i] = b[i] + c[i];  
    f[i]  = i * b[i];  
}
```

Instrucciones:
Localidad temporal,
espacial y secuencial

Las mismas instrucciones
se repiten 1000 veces

b[i] se usa tres veces
seguidas en cada
iteración del lazo

Datos:
Localidad temporal

Se accede a los elementos
de cada array de forma
consecutiva

Datos:
Localidad espacial
y secuencial

La jerarquía de memoria y la localidad

El contenido de un nivel es un subconjunto del contenido del nivel inferior de la pirámide:

- En principio, si un dato está en un nivel dado, tendrá copia en todos los niveles inferiores
- Los datos se mueven entre niveles adyacentes
- La unidad mínima de información que se transfiere entre dos niveles es un **bloque**, formado por una o más palabras
 - ▷ También se denomina **línea** si se refiere a movimiento cache ↔ memoria y **página** si se trata de memoria ↔ disco

Acierto o fallo

Cuando pedimos un datos a un nivel de la jerarquía, este puede estar o no:

- **Acierto** (*hit*): el dato requerido se encuentra ya en el nivel
- **Fallo** (*miss*): el dato requerido no se encuentra en el nivel

Acciones en caso de fallo:

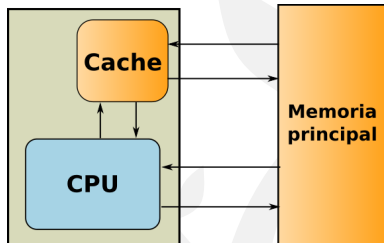
- Buscarlo en el nivel inferior de la jerarquía de memoria
- Asignarle un lugar en el nivel actual
- Es posible que sea necesario desalojar otro dato para hacer sitio al nuevo

Los fallos influyen de forma muy negativa en el rendimiento

La memoria cache

Memoria pequeña y rápida situada entre la CPU y la memoria principal

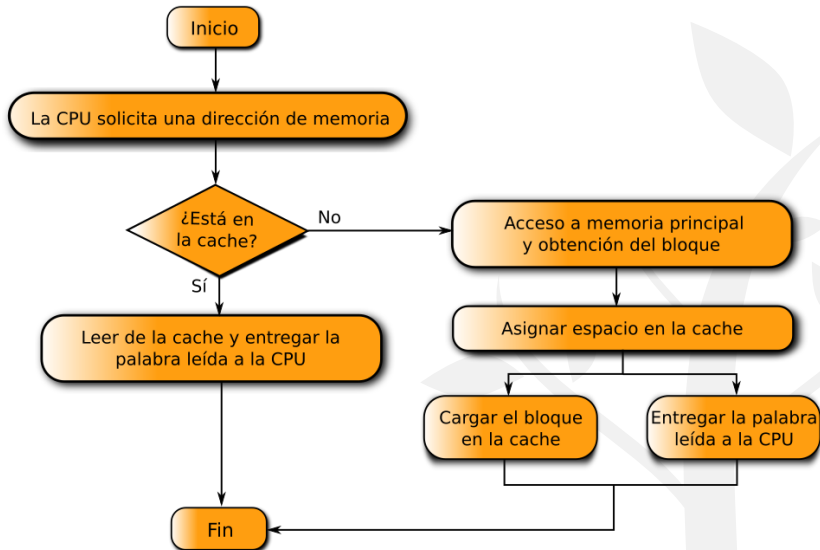
- Almacena la información en uso en un momento dado: instrucciones, datos y resultados
- El contenido de la cache cambia para adecuarse a los cambios en la ejecución del programa
- Su funcionamiento se basa en el principio de localidad
- El programador no tiene que intervenir (tampoco el SO): es el procesador (hardware) quien la gestiona



Conceptos básicos

- **Línea** (o bloque): cada una de las posiciones de la cache
 - ▷ Cada línea puede contener 1 ó más palabras
- **Tamaño de línea**: número de palabras contenidas en una línea de la cache
- La línea es la unidad indivisible dentro de una cache
 - ▷ No es posible cargar ni quitar porciones de una línea
- Cuando se carga una palabra en la cache, se carga una línea completa
 - ▷ Permite aprovechar la localidad espacial
 - ▷ A mayor tamaño de línea mejor aprovechamiento de la localidad espacial, pero... a mayor tamaño de línea menos líneas en la cache

Lectura de una línea cache



Tipos de cache

Según su organización, clasificamos la cache en tres tipos

- Caches de asignación directa
 - ▷ Cada bloque de memoria se coloca en una línea concreta en la cache
- Caches totalmente asociativas
 - ▷ Un bloque de memoria puede colocarse en cualquier línea en la cache
- Caches asociativas por conjuntos
 - ▷ Las líneas cache se agrupan en **conjuntos**
 - ▷ Cada bloque de memoria se coloca en un conjunto concreto en la cache
 - ▷ Dentro del conjunto, el bloque se coloca en cualquier línea

Tipos de fallos: clasificación 3C

Obligatorios (**Compulsory**)

- Se producen la primera vez que se referencia un dato/instrucción
- Se pueden reducir **aumentando el tamaño de línea**

De capacidad (**Capacity**)

- Cuando se referencia un dato que fue reemplazado porque la cache estaba llena
- Se pueden reducir aumentando el tamaño de la cache

De conflicto (**Conflict**)

- Se producen cuando 2 líneas compiten por la misma posición en la cache
- Se reducen con **caches separadas de datos e instrucciones** y aumentando la **asociatividad** de la cache

Las caches de asignación directa pueden ser poco eficientes ya que:

- En caso de reemplazo, la línea a sustituir está predefinida de antemano
- Esto hace que se pueda sustituir una línea que produce muchos aciertos, mientras que otras líneas poco utilizadas podrían no ser sustituidas nunca

Flexibilizar la política de colocación de las líneas:

- Cache totalmente asociativa
- Cache asociativa por conjuntos

Cache asociativa por conjuntos

La cache se divide en conjuntos de varias líneas

- **Número de vías:** número de líneas por conjunto

Cada línea de memoria tiene asignado un conjunto de forma directa, pero dentro de ese conjunto puede ir a cualquier sitio

- Dentro de cada conjunto la asociatividad es total
- Cuando el conjunto está lleno es necesario reemplazar una de las líneas: **algoritmo de reemplazo**

Solución intermedia entre la asociatividad total y la asignación directa

- Resultados próximos a la asociatividad total con una implementación más sencilla y menor coste

Caches en varios niveles

Se aumenta la jerarquía de memoria añadiendo niveles de cache entre el procesador y la memoria principal

- **Cache multinivel**
- El tamaño de la cache aumenta con el nivel, mientras que su velocidad se reduce
- Si un dato no se encuentra en el nivel, se busca en el nivel superior
- Cada nivel resuelve los fallos del nivel inferior
- Normalmente, 2 o 3 niveles de cache

Niveles de cache

Cache de primer nivel o L1

- La más cercana a la CPU
- La más rápida y pequeña
- Normalmente separada en datos e instrucciones
- Centrada en minimizar el tiempo de acierto

Cache de segundo nivel o L2

- Atiende los fallos de la L1
- Más grande y lenta que la L1, pero mucho más rápida que la memoria
- Normalmente unificada para datos e instrucciones
- Centrada en tener una razón de fallos baja para evitar accesos a memoria principal

Muchos de los procesadores para PC actuales incluyen una cache de tercer nivel (L3)

- En procesadores multicore, L1 y L2 suelen ser individuales (una por core) y L3 compartida entre los cores