



Auxiliar 9

Arquitecturas y Archivos

Profesor: Luis Mateu

Auxiliares: Gerard Cathalifaud

Vicente González

Joaquín López

Rodrigo Urrea

Semestre: Primavera 2023

1. Resumen

1.1. Memorias

1.1.1. Memoria DRAM

La memoria **DRAM** es de rápido acceso, pero no lo suficiente para alimentar a la CPU (lectura puede tomar 100 ciclos del reloj).

Esta se descompone en líneas de un cierto tamaño (usualmente 16 bytes). Se suele anotar a la **L**-ésima línea de la **DRAM** como **M[L]**.

Para una memoria con líneas de tamaño **T**, se dice que **L** es la **etiqueta** de la dirección **D** si $L = D / T$.

1.1.2. Memoria SRAM

La memoria **SRAM** es de más rápido acceso que la memoria **DRAM**. Usada en caches.

Su tamaño debe ser una potencia de 2, por ejemplo $4096 = 2^{12}$.

Al igual que la **DRAM** se descompone en líneas de un cierto tamaño (usualmente 16 bytes). Se suele anotar a la **LC**-ésima línea de la **SRAM** como **C[LC]**.

Si la línea **L** esta en el caché (de tamaño **C**), solo puede estar en **C[LC]**, con $LC = L \% C$. Si **C** es de tamaño 4096 entonces **LC** es los 12 bytes menos significativos de **L**.

Además del caché se tiene también una **SRAM** del mismo tamaño que el cache donde se almacenan las etiquetas de las líneas almacenadas en el caché, se dice que en **E[LC]** se encuentra la etiqueta de línea almacenada en **C[LC]**.

También se tiene un bit adicional donde se almacena si la línea en el caché a sido modificada, se dice que **A[LC]** indica si la linea almacenada en **C[LC]** fue o no modificada.

1.1.3. Accesos a memoria

Para acceder a un dato en la línea **L** de la **DRAM** no se consulta directamente esta, eso seria muy costoso, si no que se consulta primero en el caché.

Acceder a un dato en la línea **L** será un acierto si $E[LC] = L$, con $LC = L \% C$.

Si es un desacierto, **C[LC]** contendra una línea **L'** distinta, por lo cuál habrá que ir a buscar a la memoria por la línea consultada.



En caso de que $A[L']$ se encuentre en 1, antes de recuperar $M[L]$ se deberá escribir $C[LC]$ en $M[L']$.

Luego se recupera $M[L]$ de la memoria DRAM, se guarda en $C[LC]$, se escribe L en $E[LC]$ y se deja $A[LC]$ en 1.

1.2. Grados de asociatividad

Se dice que un cache tiene n grados de asociatividad si se compone de n caches de acceso directo (los descritos anteriormente).

Una etiqueta solo puede estar guardada en 1 de los n caches, y la búsqueda se realiza de forma paralela en ellos.

Frente a un desacuerdo se elige aleatoriamente uno de los n caches para almacenar la etiqueta.

1.3. Arquitecturas

1.3.1. Arquitectura microprogramada

- Las instrucciones se procesan de a 1 a la vez.
- Se espera a que se termine de ejecutar una para ejecutar la siguiente.
- Cada instrucción requiere de varios ciclos del reloj.

1.3.2. Arquitectura en pipeline

- Las instrucciones pasan por etapas, por ejemplo **fetch**, **decode** y **execute**.
- Se tiene una instrucción en cada etapa por ciclo del reloj, por lo que cada ciclo del reloj se completa una instrucción (en el mejor caso).
- En caso de haber un salto, se pierde el trabajo hecho en **fetch** y **decode** de las líneas inmediatamente siguientes al salto.
- Operaciones costosas (división o lectura de memoria) toman varios ciclos y bloquean el pipeline.

1.3.3. Arquitectura superescalares

- Se ejecutan varios pipelines que avanzan de manera síncrona.
- Con n pipelines se pueden ejecutar n instrucciones por cada ciclo del reloj (como máximo).
- Todavía se tiene el problema de las operaciones costosas, que bloquean el pipeline.
- Las instrucciones con dependencias (por ejemplo lectura de un registro escrito por la instrucción anterior) deben esperar a que la bloqueante termine, ralentizando el pipeline.

1.3.4. Arquitectura con ejecución especulativa y fuera de orden

- Se tienen múltiples unidades de ejecución.
- Las instrucciones se ejecutan cuando sus operandos se encuentran listos.
- En caso de que estos no estén listos (por la ejecución de una operación costosa) se prosigue con las operaciones siguientes.
- Los saltos se especulan (predicen).
- Complejos de implementar y costosos en tamaño y energía.



1.4. Archivos

1.4.1. Lectura y escritura

En Unix:

```
int open(char* path, int flags);  
size_t read(int fd, void* buf, size_t nbytes);  
size_t write(int fd, void* buf, size_t nbytes);  
int close(int fd);
```

En C:

```
FILE* fopen(const char* filename, const char* mode);  
size_t fread(char* buffer, size_t size, size_t count, FILE* stream);  
size_t fwrite(const void* buffer, size_t size, size_t count, FILE* stream);  
int fclose(FILE* stream);
```

1.4.2. Macros (Dentro de `sys/stat.h`)

- Link simbólico?

`S_ISLINK(m)`

- Archivo normal (registro)?

`S_ISREG(m)`

- Directorio?

`S_ISDIR(m)`

- Dispositivo de caracteres?

`S_ISCHR(m)`

- Dispositivo de bloques de bytes?

`S_ISBLK(m)`



1.4.3. Directorios

- Estructura de un directorio (dirent)

```
struct dirent {  
    char d_name[NAME_MAX];  
    ino_t d_ino;  
}
```

- Abrir un directorio

```
DIR* opendir(char* nom);
```

- Obtener el archivo de un directorio

```
struct dirent* readdir(DIR* dir);
```

- Reiniciar la posición al principio del directorio

```
void rewinddir(DIR* dir);
```

- Cerrar un directorio

```
int closedir(DIR* dir);
```

- Obtener la ruta de directorio de trabajo actual

```
char* getcwd(char* nom, int maxlen);
```

- Cambia el directorio de trabajo

```
int chdir(char* nom);
```



2. Preguntas

P1. La figura muestra un extracto del contenido de un cache de **128KB** de 2 grados de asociatividad y líneas de 16 bytes. El computador posee un bus de direcciones de 20 bits.

El cache se organiza en 2 bancos, cada uno con 4096 líneas.

Por ejemplo en la línea **4f2** (en hexadecimal) del banco izquierdo se almacena la línea de memoria que tiene como etiqueta **04f2** (es decir, la línea que va de la dirección **04f20** en hexadecimal a la dirección **04f2f**)

Banco 1		Banco 2	
línea	etiqueta	etiqueta	contenido
301	4301	2301	
4f2	04f2	a4f2	
c36	dc36	1c36	

Un programa accede a las siguientes direcciones de memoria:

- **a4f28**
- **dc360**
- **53014**
- **2301c**
- **1c360**
- **ec368**
- **84f20**
- **dc36c**

Conteste:

- ¿Cuál es la porción de la dirección que se usa como **etiqueta**?
- ¿Cuál es la porción de la dirección que se usa para indexar el cache?
- ¿Qué accesos a la memoria son aciertos y cuáles son desaciertos?, en caso de un desacierto muestre también el estado del cache luego del acceso.



P2. Considere el siguiente programa en **assembly**

```
a. lw x7, 4(x5)
b. add x9, x2, x3
c. blt x0, x7, p
d. slli x9, x3, 1
...
p. ori x5, x9, 7
q. srli x1, x5, 2
r. and x10, x7, x9
s. xori x8, -1, x1
```

Haga una tabla de la ejecución de las instrucciones del programa anterior en:

- Una arquitectura **microprogramada**.
- Una arquitectura en **pipeline** con etapas fetch, decode y execute.
- Una arquitectura **superescalar**, con 2 pipelines con las etapas de la arquitectura anterior.

P3. Hace poco fue la tarea 5 de PSS, y usted la hizo a altas horas de la noche. Una vez que la terminó, se fue a dormir, y al día siguiente no se acuerda de en qué carpeta dejó la tarea.

Naturalmente, usted decide crear un programa para encontrarla, que recorra recursivamente todas las carpetas (también conocidas como directorios) hasta encontrarla.

El programa debe imprimir todos los archivos que encuentre que tengan el nombre “t3.c”, sin incluir carpetas que tengan ese nombre. No debe seguir links simbólicos.

Para hacer el recorrido utilice la función `chdir`.