Auxiliar 9

Arquitecturas y Archivos

Profesor: Luis Mateu

Auxiliares: Gerard Cathalifaud
Vicente González
Joaquín López
Rodrigo Urrea

Semestre: Primavera 2023

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	
linea	etiqueta	contenido	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:

(i) ¿Cuál es la porción de la dirección que se usa como **etiqueta**?
(ii) ¿Cuál es la porción de la dirección que se usa para indexar el cache?
(iii) ¿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.

Solución:

(i) Una dirección de memoria D se almacena dentro de la linea $\frac{D}{16}$ de la DRAM, que es la que conocemos como "Etiqueta de linea", esta se utiliza en la memoria caché para representar a las 16 direcciones de memoria que se almacenan en dicha linea en la DRAM. (Por ejemplo: la etiqueta 04f2 es la linea de la DRAM que almacena las direcciones desde la 04f20 hasta la 04f2f). En la memoria caché se almacena el contenido de estas 16 direcciones, bajo la etiqueta de linea.

(ii) Teniendo la Etiqueta de Linea L, esta se indexa en la memoria caché en la linea $L\%C$, siendo C la cantidad de lineas de la memoria caché, en este caso 4096. Tenemos entonces que, siguiendo el ejemplo anterior, 0x04f2 % 0x1000 = 4f2 es la linea del caché donde se almacenar á el contenido de las direcciones 04f20 hasta la 04f2f, bajo la etiqueta 04f2.

(iii) Dicho todo esto, para saber si la consulta de una dirección es un éxito o no en la memoria caché debemos proceder de forma inversa.

- a4f28, de estar almacenada en la memoria caché, solo puede estar en la linea 4f2, vamos a esa linea en la tabla y revisamos las etiquetas de ambos bancos, si alguna de ellas a42f, la consulta es un acierto, en este caso, lo es, pues está en el Banco 2.
- dc360 es un acierto, linea c36 Banco 1.
- 53014 es un desacierto, no se encuentra en ningún banco de la linea 301, por lo tanto se guarda el contenido de las 16 direcciones que van desde la 53010 hasta la 5301f bajo la etiqueta 5301 en alguno de los 2 bancos sin preferencia particular (ustedes pueden reemplazar cualquiera), supongamos que en el 2. Nos queda el caché de la siguiente forma:

	Banco 1		Banco 2	
linea	etiqueta	contenido	etiqueta	contenido
301	4301		5301	
4f2	04f2		a4f2	
c36	dc36		1c36	

- 2301c en nuestro caso particular es un desacierto porque justo lo quitamos en la consulta anterior, en caso de haber reemplazado el Banco 1 antes, esta consulta sería un acierto. Dejaremos la linea esta vez en el Banco 1:

	Banco 1		Banco 2	
linea	etiqueta	contenido	etiqueta	contenido
301	2301		5301	
4f2	04f2		a4f2	
c36	dc36		1c36	

- 1c360 es un acierto.

- ec368 es un desacierto, lo almacenaremos en el Banco 2 (de nuevo, ustedes pueden elegir el 1 sin problemas).

	Banco	1	Banco	2
linea	etiqueta	contenido	etiqueta	contenido
301	2301		5301	
4f2	04f2		a4f2	
c36	dc36		ec36	

- 84f20 es un desacierto, almacenaremos la linea en el banco 1:

	Banco	1	Banco	2
linea	etiqueta	contenido	etiqueta	contenido
301	2301		5301	
4f2	84f2		a4f2	
c36	dc36		ec36	

- dc36c es un acierto. La tabla anterior muestra un posible resultado final del caché.

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.

Solución:

- En una arquitectura **microprogramada**, tenemos una sola componente que se encarga de hacer *fetch*, *decode* y *execute* de cada instrucción, por lo que en cada ciclo del reloj una instrucción pasa por una de las 3 fases secuencialmente y solo puede haber una en cualquiera de las 3. Nos queda entonces:

Ciclo	Fetch	Decode	Execute
1	a		
2		a	
3			a
4	b		
5		b	
6			b
7	c		
8		c	
9			c
10	p		
11		p	
12			p
13	q		
14		q	
15			q
16	r		
17		r	
18			r
19	s		
20		s	
21			s

  La ejecución de la instrucción c nos hace saltar a la instrucción p, se hace fetch de p en lugar de d, que sería la siguiente instrucción en caos de no haber salto.

- En una arquitectura en **pipeline**, tenemos componentes dedicadas a hacer fetch, decode y execute de las instrucciones por separado. Por lo tanto, es posible hacer una de cada una en un solo ciclo del reloj. Recordemos que en este caso asumiremos que trabajamos bajo una arquitectura predictiva, por lo que nos queda la tabla:

Ciclo	Fetch	Decode	Execute
1	a		
2	b	a	
3	c	b	a
4	p	c	b
5	q	p	c
6	r	q	p
7	s	r	q
8		s	r
9			s

En este caso, luego del fetch de c se hace fetch de p, pues es posible que c realice el salto hacia allá, bajo nuestros supuestos efectivamente ocurre, y para cuando se ejecuta c es posible ejecutar p inmediatamente después.

- Para la arquitectura superescalar de 2 pipelines, tenemos 2 componentes para cada una de las fases, por lo que es posible procesar hasta 6 instrucciones a la vez. En este caso, es importante tener cuidado con los saltos y las dependencias, es decir, cuando una instrucción altera un registro que otra instrucción va a usar en el mismo ciclo del reloj. Nos queda entonces:

Ciclo	Fetch	Decode	Execute
1	ab		
2	cd	ab	
3	pq	cd	ab
4	rs	pq	c
5	q	(q)	p
6		rs	q
7			rs

En este caso, luego de hacer fetch de c y d, se hace fetch de pq, pues se predice el salto a p por la ejecución de c, luego, cuando toca hacer execute de c y d (linea 4) solo se ejecuta c, pues al saltar, no debe ejecutarse d. Para el siguiente ciclo, gracias a la predicción, puede ejecutarse p. Ojo con que p y q no se ejecutan a la vez porque p altera el registro x5, que es usado por q. El resto de la ejecución ocurre normalmente