

Parte b)

Javier Kauer  
20806491-6

Cache 4KB ( $2^{12}$  bytes) de

1 grado de asociatividad

256 líneas de 16 bytes

Rellenar la tabla:

línea cache	etiqueta	contenido
0a	30a	
35	b35	
c1	1c1	

Acceso son:

b354, 30a5, 4c18, 30ac,

4c10, f0a0, b350

Al hacer las operaciones de / y de módulo se obtiene:

$$1) b354 \Rightarrow b35 \Rightarrow 35$$

$\downarrow$  etiqueta       $\hookrightarrow$  linea cache

$\hookrightarrow$  y al ver la tabla en

la linea 35, se encuentra b35, por lo tanto es un

acínto

línea cache	etiqueta	contenido
0a	30a	
35	b35	
c1	1c1	

$$2) 30a5 \Rightarrow 30a \Rightarrow 0a$$

$\downarrow$  etiqueta       $\hookrightarrow$  linea cache

$\hookrightarrow$  En la linea 0a se encuentra la etiqueta 30a

$\Rightarrow$  es un acínto

línea cache	etiqueta	contenido
0a	30a	
35	b35	
c1	1c1	

3)  $4c18 \Rightarrow 4c1 \Rightarrow c1$

↓  
etiqueta      ↓  
linea cache

↳ En la linea  $c1$  no se encuentra la etiqueta  $4c1$ , sino  $1c1$ , se tiene un **desacuerdo**

linea cache	etiqueta	contenido	↓ se tiene que ir a RAM
0a	30a		
35	b35		
c1	4c1		

4)  $30ac \Rightarrow 30a \Rightarrow 0a$

↓  
etiqueta      ↳ linea  
cache

↳ En la linea  $0a$ , se encuentra la etiqueta  $30a$ , se tiene un **acuerdo**

linea cache	etiqueta	contenido
0a	30a	
35	b35	
c1	4c1	

5)  $4c1 \Rightarrow 4c1 \Rightarrow c1$

↓  
etiqueta      ↳ linea  
cache

↳ En la linea  $c1$ , se encuentra la etiqueta  $4c1$ , se tiene un **acuerdo**

línea cache	etiqueta	contenido
0a	30a	
35	b35	
c1	4c1	

6)  $f0a0 \Rightarrow f0a \Rightarrow 0a$

↓                  ↓  
etiqueta      linea cache

↳ En la linea  $0a$ , se encuentra la etiqueta  $30a$  y no  $f0a$ , entonces se tiene un **desacuerdo**

línea cache	etiqueta	contenido
0a	f0a	
35	b35	
c1	4c1	

7)  $b350 \Rightarrow b35 \Rightarrow 35$

$\downarrow$  etiqueta       $\hookrightarrow$  linea  
cache

↳ En la linea 35, se tiene  $b35$ , entonces se tiene un **acínto**

linea cache	etiqueta	contenido
0a	f0a	
35	b35	
c1	4c1	

### c) Instrucciones:

- A and a5, s4, t0
- B addi a4, t1, 4
- C sub a4, t2, a5
- D ori a4, t3, 4
- E bgt a5, t4, L

F

G

...

- M sub a2, s0, a0
- N ori a1, a1, 4

### i) Pipeline:

Como se están ejecutando las instrucciones en una arquitectura pipeline entonces las componentes dedicadas Fetch, Decode, y Execute. Entonces se puede usar una de cada una en un solo ciclo del reloj. Además por enunciado, los saltos ocurren y se predice correctamente.

Ciclo Fetch Decode Execute

1	A		
2	B	A	
3	C	B	A
4	D	C	B
5	E	D	C
6	M	E	D
7	N	M	E
8		N	M
9			N

Despues de hacer fetch de E, se hace fetch de M porque predice correctamente.

2) Para una arquitectura superscalar, se tiene 2 pipelines con sus componentes Fetch, Decode, y Execute. Entonces se ejecutan en todos en cada ciclo del reloj con dos instrucciones. Ademas por encima predice. Importante notar si ocupan el mismo registro, entonces se espera a que se termine de usar.

Ciclo Fetch Decod. Ejecutar

1 AB

2 CD AB

3 EF CD AB

4 MN (D) C

5 EF D

6 MN E

7 MN

8

En este caso después de hacer fetch de CD, se predice que se va a hacer el salto, entonces se ejecuta EF. Además, cuando D, no puede ser ejecutado al mismo tiempo que C porque están usando el mismo registro.

Entonces tiene que esperar a que termine.