

9 (Nueve)

Nombre: EVAN REINSON

**Ejercicio 1**

Considere una caché de 1Mbyte, ASOCIATIVA por conjuntos de 4 vías, dispuesta en un procesador de 64 bits con una capacidad de direccionamiento de 4Gbytes (cada byte es directamente direccionable en memoria). Se sabe que la memoria principal posee 16M bloques. Se pide

a) Completar cada casillero con el número de bits de cada campo del formato de dirección de memoria principal:

Tag	Set	Word	Offset byte(*)
14	70	5	3

\* Llenar con 0 en el caso que no corresponda

b) Suponga que cada LÍNEA de la caché contiene además un bit de validación (V) y 4 bits de contador de accesos. Cual es el tamaño completo de una LÍNEA (expresado en bits) de la caché, considerando datos, tags y los bits de status antes mencionados?

Respuesta: 2067 bits.

c) Cuál es el tamaño total del área de TAG de la caché completa, expresada en bits?

Respuesta: 57344 bits.

d) Si la memoria caché y la memoria principal tienen tiempos de acceso de 3ns y 250ns respectivamente, ¿qué hit rate (tasa de acierto) se necesitaría para obtener un tiempo promedio de acceso a memoria (AMAT) de 40.5ns?

Respuesta: 85% s

**Ejercicio 2**

Considere la ejecución del siguiente segmento de código LEGv8 para  $N > 0$ , donde  $N \rightarrow X3$ , y  $X6$  contiene la dirección base del arreglo  $A[]$  del tipo `uint64_t`.

ADDRESS	<LABEL>	OPCODE	ASSEMBLY
		0x8b1f03e0	add x0, xzr, xzr
0x03394834	<COUNT_Z>:	0x8b0303e9	add x9, xzr, x3
0x03394838	<LOOP_FOR>:	0xb40000e9	cbz x9, NEXT
		0xf84000ca	ldur x10, [x6]
		0xb500004a	cbnz x10, NOT_Z
		0x9410b5bc	bl ZERO
0x03394848	<NOT_Z>:	0xd1000529	subi x9, x9, #0x1
		0x910020c6	addi x6, x6, #0x8
		0x17ffffffa	b LOOP_FOR
0x03394854	<NEXT>:	0x8b1f03ff	add xzr, xzr, xzr
			...
0x037c1f38	<ZERO>:	0x91000400	addi x0, x0, #0x1
		0xd61f03c0	br x30

Nombre: Iván Fernández

El sistema posee una CACHE exclusiva para INSTRUCCIONES de correspondencia ASOCIATIVA POR CONJUNTOS de 2 VÍAS de 128 bytes y 2 palabras de 32 bits por línea, sobre un procesador de 64bits, CPI = 1, que resuelve todos los data y control hazard sin necesidad de stalls, tiene una memoria principal de 4G palabras de 1 byte cada una. Considerar que la CACHE contiene los datos mostrados a continuación al inicio de la ejecución del segmento:

Set	Vía #0			Vía #1		
	Tag	V	Data	Tag	V	Data
000	00ce520	1	8b1f03ff_8b1f03ff	00ce521	0	8b1f03ff_8b1f03ff
001	3fffffff	0	ffffffff_ffffffff	3fffffff	0	ffffffff_ffffffff
010	01f5580	1	8b1f03ff_17ffffffa	00ce581	0	00000000_00000000
011	3fffffff	1	301f030f_87f00ffa	3fffffff	1	9b1f03ff_570fffff8
100	3fffffff	0	ffffffff_ffffffff	3fffffff	0	ffffffff_ffffffff
101	3fffffff	0	00000000_00000000	3fffffff	0	ffffffff_ffffffff
110	3fffffff	0	ffffffff_ffffffff	01f5580	1	9b1f0300_870f65f8
111	00000f1	1	9410b5bc_b500004a	3fffffff	0	00000000_00000000

a) Determinar el estado de la CACHE al final de la ejecución del segmento para N = 100. Para esto completar la siguiente tabla SOLO con el contenido de las líneas de CACHE que hayan sufrido modificaciones. El tamaño de la tabla no representa la cantidad de líneas necesarias.

Set	Tag	V	Data	Tag	V	Data
110	00ce520	1	9b1f0300_870f65f8	01f5580	1	9b1f0300_870f65f8
111	00ce520	1	b40000e9_fb4000ca	3fffffff	0	00000000_00000000
000	00ce521	1	b5000042_9410b5bc	00ce520	0	9b1f03ff_8b1f03ff
001	00ce521	1	d7000529_710020c6	3fffffff	0	ffffffff_ff+fffff
010	00ce521	1	17ffffff2_9b1f03ff	00ce581	0	00000000_00000000

b) Determine la cantidad de ciclos de clk necesarios para su ejecución en las mismas condiciones del punto a), considerando que cada MISS de caché tiene un tiempo de ejecución de 10 ciclos. Suponer que el pipeline ya se encuentra en régimen y que la CACHE de DATOS solo produce aciertos en los accesos, por lo que no se tiene penalidades por acceso a datos.

Expresión:  $5 * (10 + 1) + 100 * 6$  Respuesta: 655 ciclos de clk.



Nombre: Iván R. Ríos

**Ejercicio 4**

Se cuenta con un predictor por torneos que está compuesto por dos predictores más simples: un predictor global con un GR de 4 bits y un predictor de dos bits clásico. Considerando el siguiente segmento de código en LEGv8 y que los registros implicados están inicializados con los siguientes valores: X0=0, X1=99, X3=138, GR=0011:

```

1: 1: addi x0, #0, #0
2:   b.eq F1
3:   add x1, xzr, xzr
4: F1: cmpi x1, #0
5:   b.eq F2
6:   addi x1, x1, #1
7: F2: cmpi x1, 100
8:   b.neq L
9:   subis x3, x3, #1
10:  ebr L
      (bne x3, L)
    
```

PHT predictor global

Dirección	Contenido
0011 ✓	01 (T)
0111 ✓	00 (N)
1110 ✓	010 (T)
1100 ✓	01 (N)

2 bits.

- a) Indicar que posiciones de la PHT del predictor global se modifican y considerando que esta completamente inicializada en cero, que valores quedarían almacenados. (hasta el primer branch)
- b) Si este segmento de código se ejecuta muchas veces, indicar cuál de los dos predictores obtendrá mejores resultados para cada uno de los saltos:
- 8 > D2 igual
  - 10 > Solo se ejecuta en la primer iteración, ella ambos predicen mal.

**Ejercicio 5**

Se muestra a continuación el estado de los registros y las reservation station en un determinado momento de un procesador out-of-order. Deducir el segmento de código que se está ejecutando

Hardware	
Issue	= 4 instrucciones
Load	= 3 RS / 2 clk
Store	= 3 RS / 2 clk
ALU entero	= 3 RS / 1 clk
Multiplicación enteros	= 3 RS / 2 clk
ALU punto flotante	= 3 RS / 2 clk
Multiplicación punto flotante	= 3 RS / 4 clk

Name	Reservation stations						
	Busy	Op	Vj	Vk	Qj	Qk	A
load 1	<input type="checkbox"/>	load	-	-	alu int 1	0	#0
load 2	<input type="checkbox"/>	load	-	-	alu int 1	0	#8
store 1	<input type="checkbox"/>	store	-	[X0]	alu fp 1	0	#0
store 2	<input type="checkbox"/>	store	-	-	0	0	-
alu int 1	<input checked="" type="checkbox"/>	subi	[X2]	16	alu int 1	0	-
alu int 2	<input type="checkbox"/>	sub	-	[X1]	-	-	-
mult int 1	<input type="checkbox"/>	-	-	-	-	-	-
mult int 2	<input type="checkbox"/>	-	-	-	load 2	mult fp 1	-
alu fp 1	<input type="checkbox"/>	add	-	-	-	-	-
alu fp 2	<input type="checkbox"/>	-	-	[D4]	load 1	0	-
mult fp 1	<input type="checkbox"/>	div	-	-	-	-	-
mult fp 2	<input type="checkbox"/>	-	-	-	alu int 2	0	2
branch	<input type="checkbox"/>	cbz	-	-	-	-	-

Register Status								
	D0	D1	D2	D3	D4	D5	D6	D7
fp	alu fp 1	load 1	load 2	mult fp 1	-	-	-	-
int	x0	x1	x2	x3	x4	x5	x6	x7
fp	alu int 2	-	alu int 1	-	-	-	-	-

Nombre: Iván Ferrón

**Ejercicio 4**

Se cuenta con un predictor por tomness que está compuesto por dos predictores más simples: un predictor global con un GR de 4 bits y un predictor de dos bits clásicos. Considerando el siguiente segmento de código en LEGv8 y que los registros implicados están inicializados con los siguientes valores: X0=0, X1=99, X3=138, GR=0011:

- 1) L: ~~addi x0, #0~~ *addi x0, x0, #0*
- 2) b.eq E1
- 3) add x1, x2r, x2r
- 4) E1: cmpi X1, #0
- 5) b.eq E2
- 6) addi X1, X1, #1
- 7) E2: cmpi X1, 100
- 8) b.neq L
- 9) subis x3, X3, #1
- 10) ~~cbz~~ *cbz* x3, L

PHT predictor global

Dirección	Contenido
0011 ✓	01 (7) ✓
0111 ✓	00 (0) ✓
1110 ✓	01 (7) ✓
1100 <i>1 x</i>	01 (7) <i>2 bits</i> ✓

- a) Indicar que posiciones de la PHT del predictor global se modifican y considerando que esta completamente inicializada en cero, que valores quedarían almacenados. (hasta el primer branch a L)
- b) Si este segmento de código se ejecuta muchas veces, indicar cuál de los dos predictores obtendrá mejores resultados para cada uno de los saltos.

- 2) D2 igual
- 5) D2 igual

- 8) D2 igual
- 10) Solo se ejecuta en la primer iteración y el predictor mal.

**Ejercicio 5**

Se muestra a continuación el estado de los registros y las reservation station en un determinado momento de un procesador out-of-order. Deducir el segmento de código que se está ejecutando

Hardware	
Issue = 4 instrucciones	
Load = 3 RS / 2 clk	
Store = 3 RS / 2 clk	
ALU entero = 3 RS / 1 clk	
Multiplicación enteros = 3 RS / 2 clk	
ALU punto flotante = 3 RS / 2 clk	
Multiplicación punto flotante = 3 RS / 4 clk	

Name	Reservation stations						
	Busy	Op	Vj	Vk	Qj	Qk	A
load 1	<input type="checkbox"/>	load	-	-	alu int 1	0	#0
load 2	<input type="checkbox"/>	load	-	-	alu int 1	0	#8
store 1	<input type="checkbox"/>	store	-	[X0]	alu fp 1	0	#0
store 2	<input type="checkbox"/>						
alu int 1	<input checked="" type="checkbox"/>	subi	[X2]	16	0	0	
alu int 2	<input type="checkbox"/>	sub	-	[X1]	alu int 1	0	
mult int 1	<input type="checkbox"/>						
mult int 2	<input type="checkbox"/>						
alu fp 1	<input type="checkbox"/>	add	-	-	load 2	mult fp 1	
alu fp 2	<input type="checkbox"/>						
mult fp 1	<input type="checkbox"/>	div	-	[D4]	load 1	0	
mult fp 2	<input type="checkbox"/>						
branch	<input type="checkbox"/>	cbz	-	-	alu int 2	0	2

Register Status								
	D0	D1	D2	D3	D4	D5	D6	D7
Q1	alu fp 1	load 1	load 2	mult fp 1				
	X0	X1	X2	X3	X4	X5	X6	X7
Q1	alu int 2		alu int 1					