## Ejercicio i

Considere para un procesador de arquitectura LEGv8 (Address bus de 64 bits) una CACHÉ L1 exclusiva para INSTRUCCIONES de 2K bytes de datos, de correspondencia DIRECTA, de 2 palabras (cada palabra es 1 instrucción de 32 bits) por bloque. La siguiente lista corresponde a DIRECCIONES de acceso a la memoria de instrucciones (PC) para la ejecución de un lazo (solo se muestran 32 bits de los 64 bits para abreviar):

1> 0x00...00 00 02 F4

<loop: asm\_instruction>

2> 0x00...00 00 02 F8

3> 0x00...00 00 02 FC

4> 0x00...00 00 1A FC

5> 0x00...00 00 1B 00

<cbz loop>

a) Calcular el formato de memoria principal para la CACHÉ dada y completar la siguiente tabla en formato HEXADECIMAL para 2 iteraciones del loop:

b) Suponiendo que el procesador opera con el pipeline en régimen a 1GHz, el tiempo de acceso a CACHE es de 1 ciclo y el Miss penalty es de 50 ciclos, calcular el tiempo total de ejecución (en seg) para las condiciones del punto a).

Tiempo de ejecución: 0,00000021 seg. X 21 nseg.

c) Ahora considere que puede cambiar el tamaño del bloque y la CACHE pasa a ser asociativa por conjunto de N-vías. Determine el tamaño del bloque y el número de vías ÓPTIMOS para minimizar la ocurrencia de MISS (se considera el número óptimo aquel arriba del cual no se generan beneficios adicionales).

Acc.	Hit / Miss	Tag (54) bits	Index (4) bits	Word (1) bits
1>	Miss	0 x O	OX 17A	OXO
2>	Miss	OxO	0x 17C	OXO
3>	Miss	OxO	0x 17E	OXO
4>	Hit	0x6	OX HE	6X0
5>	Miss	0x6	0x180	OXO
1>	Hit	ox6	OX17A	OXO
2>	#1+	0x6	OX17C	OXO
3>	Hit	6x6	0x17E	OXO
4>	Hit	0x6	0x17E	000
5>	Hit	0x6	OX/80	OXO

ALCOHOLD STATE OF THE PARTY OF		
Tamaño de bloque:	7	_ bytes.

Número de vías: vias.

d) Calcular el tiempo de ejecución de 2 iteraciones del lazo para las condiciones del punto c)

Tiempo total de ejecución:

Ejercicio 2

Dado un procesador de arquitectura LEGv8 2-issue, que predice los saltos perfectamente (de modo que los hazard de control son manejados por hardware), con una modificación que permite ejecutar dos instrucciones aritméticas/lógicas juntas (es decir, una instrucción

en un paquete puede ser de cualquier tipo -IssueA-, pero la otra debe ser una instrucción aritmética - La códica I EGVB donde Va aritmética o lógica -IssueB-). Para el siguiente fragmento de código LEGv8, donde X0=A (dirección base del arreglo A, de elementos DoubleWord), X1=N (cantidad de elementos del arreglo) y los demás registros se encuentran inicializados con "0":

3 / 4	14.010.50	
1>		LSL X3, X1, #3
2>		ADD X3, X0, X3
3>		ORRI X2, XZR, #1
4>	loop:	SUBS X12, X0, X3
5>		B.GE end //>= signado
6>		LDUR (29, [X0, #0]
7>		LDUR \$10, [X3, #0]
8>		SUB X12, (X9, X2
9>		ADD X11, X10, X2
10>		STUR X12, [X3, #0]
11>		STUR X11,[X0, #0]
12>		ADDI X0, X0, #8
13>		SUBI X3, X3, #8
14>		B loop
	end:	

C

ción I
_
_
_
Felicit
13 17

a) Analizar en el código las dependencias de datos que NO sean condicionales y luego

completar la si	guiente tabla	a:
Tipo	Registro	instrucciones
WAR	X3	1,2
WAR	X9	6,8
WAR	X 10	7,9
RAW	XO	11 122
		(E)
	4/101	Dell'In
	Old The said	

Tipo	Registro	instrucciones
		JIS CONTRACTOR
and the same is	- a F	Villege of the control
		Wind memer
		A Commission
CO IO- ROLLS		
T. Inches Tale	4	
Personnel in	0 00	
		1
omist was visit o		The state of the s

b) Sin alterar el orden de las instrucciones, mostrar en la tabla junto al código cómo organizaria los issue packets para ejecutar el programa en la menor cantidad posible de ciclos de clock (cada instrucción sólo puede agruparse con la inmediata anterior, la inmediata posterior o una nop). El compilador asume toda la responsabilidad de insertar instrucciones nop para que el código se ejecute sin necesidad de generación de stalls. Usar los números correspondientes para referirse a las instrucciones del código.

c) Calcular la cantidad de ciclos de reloj que toma la ejecución del código dado en un procesador LEGv8 de 1-issue con forwarding-stall y en el procesador LEGv8 2-issue descrito. Suponer que en ambos procesadores los saltos son perfectamente predichos y los hazard de control son manejados por hardware (no hay flush de instrucciones). Suponer que N=5 y considerar la totalidad de las iteraciones realizadas. Luego completar:

Tiempo de ejecución en 1-issue=\_-

ii. Tiempo de ejecución en 2-issue= 13.

iii. Ganancia en velocidad = \_\_\_

## Ejercicio 3

Considerando un microprocesador out-of-order execution implementado mediante el algoritmo de Tomasulo, el código en assembler en el ejercicio 2, muestre el contenido de las tablas de estado de las Reservation stations y registros para el 5to clock de ejecución (inclusive). El procesador cuenta con el siguiente hardware:

Hardware	9	
El salto se predice correctamente		
Issue = 4 instrucciones		To
Load = 4 RS / 1 clk		I
Store = 4 RS / 1 clk		1
ALU entera = 4 RS / 1 clk		
ALU punto flotante = 4 RS / 2 clk		10
Multiplicación punto flotante = 2 RS	S / 3 clk	

## Ejercicio 4

Se cuenta con un predictor global con un GR de 8 bits, considerando el siguiente segmento de código en LEGv8:

.org 0x	00	
0x1> L:	add x1, xzr, #2	
0x2>	udiv x2, x0, x1	//Se verifica
0x3>	mul x3, x1, x0	//si x0 es
0x4>	sub x3, x3, x2	//múltiplo de 2
0x5>	cbnz x3, s1	
	•	
0x6>s1:	add x1, xzr, #4	
0x7>	udiv x2, x0, x1	//Se verifica
0x8>	mul x3, x1, x0	//si x0 es
0x9>	sub x3, x3, x2	//múltiplo de 4
0xA>	cbnz x3, s2	
	•••	
0xB>s2:	add x0, x0, #1	
	cmp x0, 1000	
ØxD>	The state of the s	

Contenido
AND IN THE STATE OF THE STATE O
Saring times on the se
The second
The state of the s

## Parcial 2 - Arquitectura de computadoras 2023

a) Completar la tabla en hexadecimal en las posiciones de la PHT del predictor global que se modifican en dos iteraciones del salto de la posición 0xD, considerando que está completamente inicializada en cero y que X0=10 y el GR=0x34 antes de comenzar a ejecutarse primer instrucción del código.

b) Considerando que inicialmente X0=0 y que existe un patrón repetitivo (cada dos iteraciones del código aparece un número par, y cada 4 aparece un número múltiplo de 4). ¿Qué cantidad de bits debería tener como mínimo el GR para almacenar el patrón del salto de la posición 0x5? ¿Y para el salto de la posición 0xA?

0x5>	0xA>	=

Ejercicio 5

Completar el siguiente kernel de openCL que multiplique una matriz cuadrada de NxN elementos por una constante alpha.

El kernel debe corresponderse con el siguiente código de host:

```
plataform_list = cl.get_platforms()
devices = plataform_list[0].get_devices(device_type = cl.device_type.GPU)
context = cl.Context(devices=devices)
queue = cl.CommandQueue(context)
a_np = np.arange(N*N).astype(np.float32)
a_g = cl.Buffer(context, cl.mem_flags.READ_WRITE |
cl.mem_flags.COPY_HOST_PTR, hostbuf = a_np)
program = cl.Program(context, KernelSource).build()
kernel = program.parallelMMult
kernel.set_scalar_arg_dtypes([np.int32, np.float32, None])
kernel.set_args(N, alpha, a_g)
globalRange = (N,N)
localRange = (N,N)
ev = cl.enqueue_nd_range_kernel(queue, kernel, globalRange, localRange)
cl.enqueue_copy(queue, a_np, a_g)
```