25/10/24 Apellido, Nombre:

Ejercicio 1

Considere la ejecución del siguiente segmento de código LEGv8 para i > 0, donde i \rightarrow X3, y X6 contiene la dirección base del arreglo A[] del tipo uint32_t.

El procesador tiene una CACHE exclusiva para **DATOS** de correspondencia **DIRECTA** de 64Kbyte y 4 palabras (w_p) por línea, sobre un procesador de 32bits, CPI = 1, que resuelve todos los data y control hazard sin necesidad de stalls, tiene una memoria principal de 4G palabras (w_m) de 1 byte cada una. Considerar que el resto de los registros están inicializados en 0 y la CACHÉ vacía al inicio de la ejecución del segmento.

LSL X9, X3, #2
ADD X9, X6, X9
SUB X11, X6, X9
CBZ X11, end
LDUR X10, [X9, #0]
ADD X0, x10, X0
SUBI x9, x9, #8
В 100р
STUR X0, [X9, #0]

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

Tag	Index	Word	Offset byte(*)
	t.	10, 10,10	

^{*} Llenar con 0 en el caso que no corresponda

b) Cuántos MISS de CACHE se produjeron en la ejecución del fragmento de código si i = 6 y la dirección base del arreglo A es 0x00F192B8?. Suponga que los accesos a memoria de escritura y lectura tienen IDÉNTICO comportamiento para la CACHE.

Respuesta: ___ Miss

c) Completar como queda la memoria CACHE al finalizar la ejecución del segmento para las condiciones del punto b). Llenar una fila por cada MISS producido. Se completó la primer línea sólo con fines ilustrativos.

Index (hex)	Tag (hex)	V	W ₃	W ₂	Data W ₁	W _o
0xC0CA	0xCAFE	1	A[11]	A[10]	A[9]	A[8]
	* 19					
3	100					
	<u> </u>					
					1	
			41			

En la figura se muestra un diagrama simplificado del procesador LEGv8 2-issue. Considerando la forma de funcionamiento de la implementación completa (con multiplexores, HDU, unidad de forwarding, los caminos de forwarding incluso a la etapa ID para la resolución de saltos, etc.) y asumiendo que los saltos son perfectamente predichos (en el ciclo de clock posterior al fetch de un salto se hace fetch del paquete correcto: PC+8 o PC+offset, según corresponda, no hay penalidad), responder:

a) Dado el siguiente fragmento de código en assembler LEGv8 (más abajo), analizar las dependencias de datos y completar con el primer caso que encuentre de cada una de las

siguientes dependencias:	Tipo de dependencia	Números de instrucciones	Operando en conflicto
	RAW		
	RAW condicional	The second	and the same of th
	WAW		

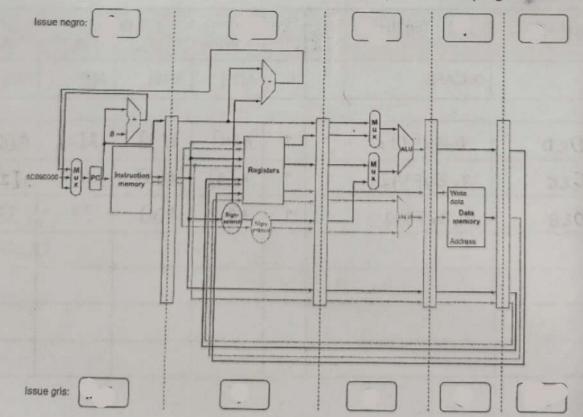
b) Mostrar en la tabla cómo organizaria los issue packets para ejecutar el programa en la menor cantidad posible de ciclos de clock sin alterar el orden de las instrucciones (cada instrucción sólo puede agruparse con la inmediata anterior, la inmediata posterior o una nop). El compilador asume toda la 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

Issue negro

Issue gris

1>	ORRI <u>X4</u> , XZR, #0x8	
2> loop:	CBZ X0, end	
3>	LDUR X2, [X1, #0]	1,000
4>	SUBI X0, X0, #1	
5>	LDUR X3, [X1, #8]	
6>	ADD X1, X1, X4	
7>	STUR X2, [X1, #0]	
8>	ADD X3, X3, X1	
9>	CBZ XZR, loop	
10> end:	the state of the s	
ALCOHOLOGICA CONTRACTOR CONTRACTO		

Completar en los recuadros vacíos del diagrama del procesador qué instrucción se está ejecutando en cada etapa del pipeline, tanto en el issue negro como en el gris, en el ciclo de clock número 6, considerando que antes de comenzar la ejecución del programa X0=2.



Ejercicio 3

Considerando un microprocesador out-of-order execution implementado mediante el algoritmo de Tomasulo que cuenta con las características del hardware descritas en la tabla y que ejecuta el siguiente código en assembler, muestre el contenido de las tablas de estado de las Reservation stations y registros (en la próxima página) para el 4to clock de ejecución (inclusive y considerando que se hace fetch en el clk 0):

1>	L1: SUB X0, X0, #8
2>	LDUR X10, [X0, #0]
3>	LDUR X11, [X1, #0]
4>	ADD X1, X0, #8
5>	ADD X12, X10, X12
6>	MULT X12, X10, X12
7>	STUR X12, [X0, #0]
8>	STUR X12, [X1, #0]
9>	CBNZ X0, L1
10>	ORR X0, X1, X2

Hardware	Ī
El salto se predice correctamente	
Issue = 4 instrucciones	Ī
Load = 4 RS / 1 clk	
Store = 4 RS / 1 clk	
ALU entera = 4 RS / 1 clk	
ALU punto flotante = 4 RS / 2 clk	ī
Multiplicación punto flotante = 2 RS / 3 d	ilk

Ejercicio 4

Considerar que un procesador que cuenta con un predictor global de 8 bits ejecuta el siguiente código en C y compilado como se muestra a la derecha:

```
A[3]={3,7,4};

for (i = 0; i; i; i++) { 7/x0<-i

if (A[i] < 5) { //x1<-A[i]; x2<-5;

...

0x34
```

	Dirección		
	0×14	L:	
ı			
			cmp X1, X2
	0x34		b.lt e_if /
	0x44		cbnz X0, L

Completar la siguiente tabla con todas las posiciones de la PHT que se modifican, considerar que todas las posiciones de memoria están inicializadas en 0 y el GR=0x3.

	Dirección	165	Contenido	16)
	1			
	<u> </u>		-	
_	1		-	
-	-		110	
-	T	11576	T I	
	THE RE		1 41 2	

Name	Busin		Re	servation statio	ns		
	Busy	Op	Vj	Vk	Qj	Qk	
				**************************************		Qn.	A
		-					
The second second							
0.0	10			2004			
						2 00 - 10	
	1 1 1 2 1						
				100			
**							
	4		-				
	574						
			1				
E 6 00 0	199						

Register Status

X6

X5

X4

X2

X1

XO

QI

хз

X12

X11

X10

X9

X8

X7