

Fecha: 25/10/2022

Nombre:

Parcial 2

Ejercicio 1 (20%):

Dada la siguiente secuencia de código en assembler LEGv8:

ADDI X9, X6, #8
ADD X10, X6, XZR
STUR X10, [X9, #0]
LDUR X9, [X9, #0]
ADD X0, X9, X10

Y asumiendo que las etapas individuales del procesador LEGv8 tienen las siguientes latencias:

Instruction Fetch	Instruction Decode	Execute	Memory	Write Back
100 ns	60 ns	80 ns	120 ns	70 ns

a) Completar la tabla:

	Sin pipeline	Con pipeline
Frecuencia del reloj mínima	100 = 213 MHZ	120= 813M
Latencia de una instrucción	430 05	600 ns
Tiempo que tarda en ejecutar la secuencia dada(*)	5.100 AS = 2150	6004/200 4 = 1080
Mejora en el tiempo de ejecución de la secuencia	Gorancia - 1.9907	

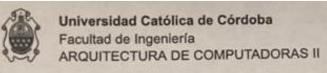
(*) asumir que el procesador con pipeline tiene implementado forwarding-stall

b) Asumiendo que es posible rediseñar una de las etapas del procesador dado y disminuir su latencia a la mitad, responder:

ii) ¿Cuál sería la frecuencia de reloj del procesador con pipeline modificado?

Ejercicio 2 (45%):

La siguiente secuencia de código se ejecuta en un procesador LEGv8 con pipeline. Considerando que antes de la ejecución del código, el contenido de la posición de memoria apuntada por X5 es: 0b1101.



Fecha: 25/10/2022

Nombre:

 a) Analizar en el código las dependencias de datos y de control, y luego completar la siguiente tabla:

Tipo de dependencia	Registro involucrado (si aplica)	N° instrucciones involucradas	Genera hazard sin forwarding stall?
***		***	***

b) Mostrar el orden de ejecución de las instrucciones (hasta que se termine de ejecutar la instrucción n° 9), considerando un procesador con forwarding-stall. Dejar correctamente indicados los caminos de forwarding.

Ejercicio 3 (35%):

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 que en cada issue packet una instrucción pueda ser cualquier tipo y la otra deba ser una instrucción aritmética/lógica. Para el siguiente fragmento de código LEGv8:

a) Sin alterar el orden de las instrucciones, mostrar en la siguiente tabla cómo organizaría 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.

Instrucción de cualquier tipo	Instrucción aritmética/lógica	

b) Mostrar el orden de ejecución del código del punto "a" en el procesador 2-issue (sólo hasta completar una iteración del bucle "LOOP", suponer que la posición de memoria 0x100 contiene un valor mayor que cero). Indicar los caminos de forwarding utilizados.