Universidad Católica de Córdoba
Facultad de Ingeniería Fecha: 12/11/2021
ARQUITECTURA DE COMPUTADORAS II

Recuperatorio parcial 2

Ejercicio 1 (15%):

Asumiendo que las etapas individuales de un procesador tienen las siguientes latencias:

	IF	ID	EX	MEM	WB
Latencia por etapa:	**250 ns**	**90 ns**	**130 ns**	**250 ns**	**70 ns**

a. Completar:

Tipo de procesador	Máxima frecuencia de reloj [Hz]	Latencia de una instrucción [ns]
Con pipeline		
Sin pipeline		

b. ¿Cuántas instrucciones seguidas se deben ejecutar para que el rendimiento del procesador con pipeline sea mejor que el mismo sin pipeline? Justificar.

Ejercicio 2 (35%):

Considerando un microprocesador LEGv8 con forwarding stall, dado el siguiente fragmento de código de instrucciones LEGv8:

```
        1> ADDI X23, XZR, #1
        2> ADDI X9, XZR, #16
        3> SUB X24, X9, X24
        4> CBZ X24, L1
        5> LSL X10, X23, #3
        6> ADD X10, X10, X24
        7> LDUR X9,[X10, #0]
L1:     8> ADD X10, X10, X9
```

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 gráficamente el orden de ejecución del programa en el cauce, indicando los caminos de forwarding utilizados. Considerar que antes de comenzar el programa X24 = 16.

c) Mostrar gráficamente el orden de ejecución del programa en el cauce, indicando los caminos de forwarding utilizados. Considerar que antes de comenzar el programa X24 = 32.

Ejercicio 3 (50%):

Dado un procesador de arquitectura LEGv8 2-issue, que predice los saltos perfectamente, de modo que los hazard de control son manejados por hardware. Para el siguiente fragmento de código LEGv8:

```
      1>  ADDI X0, XZR, #6
loop: 2>  CBZ X0, exit
      3>  LDUR X3, [X1, #0]
      4>  LDUR X4, [X1, #8]
      5>  ADD X3, X3, X3
      6>  ADD X4, X4, X4
      7>  ADD X5, X3, X4
      8>  STUR X5, [X2, #0]
      9>  ADDI X1, X1, #16
      10> ADDI X2, X2, #8
      11> SUBI X0, X0, #1
      12> B loop
exit: 13> ...
```

a) Mostrar en la siguiente tabla cómo organizaría los issue packets para ejecutar el programa en la menor cantidad posible de ciclos de clock (**si es necesario, reordenar las instrucciones**). El compilador asume toda la responsabilidad de insertar instrucciones nop para que el código se ejecute sin necesidad de generación de stalls. Mostrar sólo hasta una iteración del loop.

ALU or branch instruction	Data transfer instruction
...	...

b) Mostrar el orden de ejecución del código del punto “a” en el procesador 2-issue (sólo una iteración del loop). Indicar los caminos de forwarding utilizados.

c) Calcular el tiempo de ejecución del código dado en un procesador LEGv8 de 1-issue con forwarding-stall y en el procesador LEGv8 2-issue, si ambos trabajan a una frecuencia de 1GHz. Suponer que en ambos procesadores los saltos son perfectamente predichos y los hazard de control son manejados por hardware (no hay flush de instrucciones). Considerar la totalidad de las iteraciones realizadas por el código.

d) Aplicar la técnica de loop unrolling para que cada iteración del nuevo bucle se corresponda con dos iteraciones del bucle “loop” original. Está permitido hacer register renaming y reordenar las instrucciones. Mostrar el código resultante. Luego completar la siguiente tabla, mostrando cómo organizaría los issue packets para ejecutar su nuevo programa en la menor cantidad posible de ciclos de clock.

ALU or branch instruction	Data transfer instruction
...	...