Universidad Católica de Córdoba
Facultad de Ingeniería Fecha: 02/11/2021
ARQUITECTURA DE COMPUTADORAS II

Parcial 2

Ejercicio 1 (15%):

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

Etapa:	**IF**	**ID**	**EX**	**MEM**	**WB**
Latencia:	20 ns	30 ns	40 ns	80 ns	15 ns

a) ¿Cuál es la latencia de una instrucción y el menor periodo posible de reloj para un microprocesador sin pipeline?

b) ¿Cuál es la latencia de una instrucción y el menor periodo posible de reloj para un microprocesador con pipeline?

c) ¿Cuántas instrucciones se deben ejecutar seguidas para que el rendimiento del microprocesador con pipeline sea mejor que el mismo sin pipeline?

Ejercicio 2 (35%):

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

```
        1> ADD X9, X9, #8
        2> LDUR X0, [X9, #0]
        3> SUB X10, X9, X1
        4> CBNZ X10, L1
        5> ADDI X2, XZR, #1
L1:     6> SUB X3, X2, X0
        7> OR X4, X2, X0
```

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: X1=17 y X9=9.

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: X1=15 y X9=23.

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>  ADD X10, XZR, XZR
      2>  ADDI X11, XZR, #64
L1:   3>  SUBIS X2, X10, #64
      4>  B.EQ end
      5>  LDUR X0, [X10, #0]
      6>  ADDI X11, X11, #8
      7>  LDUR X1, [X11, #0]
      8>  SUBS X3, X0, X1
      9>  STUR X0, [X11, #0]
      10> B.LT L1
      11> STUR X1, [X10, #0]
      12> ADDI X10, X10, #8
      13> B L1
end:  ...
```

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. Mostrar sólo hasta una iteración del loop L1.

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 de L1, suponer que X0>X1). 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 1.5GHz. 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 siempre X0>X1. 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 “L1” 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
...	...