Práctico - Superescalares

Ejercicio 1

Considere el siguiente código:

```
I1: Move R3,R7        /R3<-(R7)/
I2: Load R8,(R3)           /R8<-Memory(R3)/
I3: Add R3,R3,4       /R3<-(R3)+4/
I4: Load R9,(R3)           /R9<-Memory(R3)/
I5: BLE R8,R9,L3           /Branch if (R9)>(R8)/
```

Este programa tiene dependencias WAW, RAW y WAR. Muéstrelas.

Ejercicio 2

Considere el siguiente código MIPS:

```
loop:
      LD          F0,0(R1)
      MULTD       F0,F0,F2
      LD          F4,0(R2)
      ADDD        F0,F0,F4
      SD          0(R2),F0
      SUBI        R1,R1,8
      SUBI        R2,R2,8
      BNEQZ       R1,loop
```

El loop calcula Y[i] = a . X[i] + Y[i]. Suponga que se ejecuta en el pipeline MIPS (múltiples unidades funcionales, con forwarding, etapas IF, ID, EX, MEM, WB, C y múltiples puertos de escritura en el banco de registros). Recuerde que en este pipeline la penalización por saltos es de tres ciclos.

a. Calcule el tiempo de ejecución del loop para R1 = 256.
b. Desenvuelva (unroll) el siguiente loop una vez y calcule nuevamente el tiempo de ejecución. Calcule la aceleración lograda con esta técnica.
c. Reordene las instrucciones del código desenrollado para minimizar las detenciones. Calcule la aceleración total obtenida con respecto al código original.

Ejercicio 3

Repita las partes a), b) y c) del ejercicio 2 con el siguiente código:

```
loop: LD          F0,0(R1)
      LD          F4,0(R2)
      MULTD       F0,F0,F4
      ADDD        F2,F0,F2
      SUBI        R1,R1,8
```

```
SUBI        R2,R2,8
BNEQZ       R1,loop
```

Ejercicio 4

Para el siguiente pseudocódigo, renombre los registros para prevenir problemas de dependencia.

```
I1: R1=100
I2: R1=R2+R4
I3: R2=R4-25
I4: R4=R1+R3
I5: R1=R1+30
```

Ejercicio 5

Considere un procesador MIPS con etapas IF, ID, I, EX, MEM, WB, C, con emisión fuera de orden, ejecución especulativa, múltiples unidades funcionales, forwarding completo y múltiples puertos de escritura en el banco de registros. Para el siguiente código, realice un diagrama del pipeline, indicando, para cada instrucción, en qué etapa del pipeline se encuentra. Resuelva hazards WAR mediante detención del pipeline.

```
LD          F4,0(Rx)
MUL.D       F2,F0,F2
MUL.D       F8,F4,F2
LD          F3,0(Ry)
ADD.D       F6,F8,F5
SUB.D       F8,F3,F6
SD          F8,0(Ry)
```

Ejercicio 6

Considere un procesador superescalar MIPS de ancho 2, con etapas IF, ID, I, EX, MEM, WB, C, emisión fuera de orden, ejecución especulativa, register renaming, múltiples unidades funcionales y forwarding completo.

Sea el siguiente código:

```
LD F2,0(Rx)
MULT.D F8,F2,F0
MULT.D F2,F6,F2
LD F4,0(Ry)
ADD.D F4,F0,F4
ADD.D F10,F8,F2
ADDI   Rx,Rx,#8
```

```
ADDI Ry,Ry,#8
SD F4,0(Ry)
SUB R20,R4,Rx
```

a. Realice un diagrama del pipeline indicando, para cada ciclo, en qué etapa del pipeline se encuentra cada instrucción.
b. Indique en qué ciclos se producirían hazards WAR y WAW si no se utilizara *register renaming*.

Ejercicio 7

Compare el costo en instrucciones de un hazard de control, entre el pipeline MIPS de 5 etapas y su versión superescalar de ancho dos.

Ejercicio 8

Considere el pipeline de etapas IF,ID,EX,MEM,WB,C con múltiples unidades funcionales, un ùnico puerto de escritura en el banco de registros y scoreboard, explique cómo se puede utilizar el scoreboard para:

a. Detectar hazards estructurales en el banco de registros
b. Detectar hazards WAW.

Ejercicio 9

Considere el pipeline de etapas IF,ID,EX,MEM,WB,C con múltiples unidades funcionales y scoreboard. Explique cómo se debe actualizar el scoreboard si son emitidas dos instrucciones con diferentes unidades funcionales que deben escribir al mismo registro.

Ejercicio 10

Considere la técnica de *register renaming.* Explique en qué momento se puede marcar como ‘libre’ un registro físico en la *free list* (lista de registros físicos libres).