Práctico 1: Pipeline

Ejercicio 1

Considere el pipeline del procesador MIPS de cinco etapas visto en teórico, sin forwarding. Determine las dependencias de datos del siguiente fragmento de código, y establezca el forwarding necesario para ejecutar con las mínimas detenciones del pipeline posibles:

```
add $3, $4, $6 ; suma registros $4,$6 y los almacena en $3
sub $5, $3, $2 ; resta registros $3, $2 y los almacena en $5
lw $7, 100($5) ; carga la palabra de memoria $5+100 a $7
add $8, $7, $2 ; suma los registros $7, $2 y los almacena en
$8
```

Ejercicio 2

Considere el pipeline del procesador similar a MIPS visto en clase (etapas IF, ID, EX, MEM, WB).

```
loop: LD    R1, 0(R2)   ; carga en R1 el valor de la dirección 0 + R2
      DADDI R1, R1, #1  ; R1 = R1 + 1
      SD    R1, 0 (R2)  ; guarda el valor de R1 en la dirección 0+R2
      DADDI R2, R2, #4  ; R2 = R2 + 4
      DSUB  R4, R3, R2  ; R4 = R3 - R2
      BNEZ  R4, loop    ; saltar a loop si R4 != 0
```

Asuma que el valor inicial de R3 es R2 + 400

a) Realice un diagrama del pipeline, mostrando para cada ciclo de reloj, en qué etapa del pipeline se encuentra cada instrucción (al menos dos iteraciones). Considere que una escritura realizada en el banco de registros puede ser leída en el mismo ciclo de reloj por otra instrucción. Recuerde que no existe ningún mecanismo de forwarding y que la penalización por salto no tomado en dicho pipeline es de tres ciclos (ver diagrama de teórico) ¿Cuántos ciclos de reloj demora en ser ejecutado el loop?

b) Repita el ejercicio anterior, considerando una implementación completa de forwarding y manteniendo todo lo anterior. ¿Cuántos ciclos de reloj demora en ser ejecutado el loop?

c) Suponga ahora que se utiliza un predictor de saltos en la etapa de decodificación, de modo que al final de dicha etapa se carga el PC con la predicción. Considerando la política 'always taken'. ¿Cuántos ciclos de reloj demora en ser ejecutado el loop?

Ejercicio 3

Considere el pipeline MIPS de 5 etapas visto en teórico, con múltiples unidades funcionales y múltiples puertos de escritura en el banco de registros. Para el siguiente fragmento de código, realice un diagrama del pipeline, indicando en qué etapa se encuentra cada instrucción para cada ciclo de reloj, con las siguientes consideraciones:

a) Sin forwarding implementado.
b) Con forwarding completo.

```
add R3, R4, R6 ; suma registros R4,R6 y los almacena en R3
sub R5, R3, R2 ; resta registros R3 y R2 y los almacena en R5
mul.d F1, F2, F3 ; multiplica registros F2 y F3 y almacena en
F1
mul R6, R5, R1 ; multiplica registros R5 y R1 y almacena en R6
add.d F1, F4, F6 ; suma registros F4 y F6 y almacena en F1
add R3, R4, R6 ; suma registros R4,R6 y los almacena en R3
```

Ejercicio 4

Para el pipeline MIPS con varias unidades funcionales y un único puerto de escritura en banco de registros, indique la función lógica de la señal de control que detecta hazards estructurales en la etapa MEM (ver dibujo del pipeline en teórico).

Ejercicio 5

a. Para el pipeline MIPS con varias unidades funcionales visto en teórico, indique la condición de detección de hazards WAW en la etapa de decodificación. Indique la función lógica de la señal de control que la implementa.

b. Implemente sobre el pipeline la conexión necesaria para que una instrucción no impacte sus resultados en el banco de registros en el caso de una dependencia WAW. **Sugerencia**: Agregar un bit que se propague en el pipeline que evite que la instrucción impacte sus resultados en el banco de registros. Al detectar el hazard WAW (parte a), encender este bit para la instrucción que no debe impactar sus resultados.

Ejercicio 6

Para el pipeline MIPS de 5 etapas visto en teórico, extienda la señal que detecta excepciones de modo que también marque las instrucciones anteriores en el pipeline. Si se agrega la etapa de commit, ¿Sigue siendo necesaria esta señal? Justifique.

Ejercicio 7

El Reorder Buffer junto a la etapa de commit y la adición de un nuevo banco de registros permiten manejar excepciones en el contexto de finalización fuera de orden. Sin embargo, si una instrucción *store* impacta sus resultados fuera de orden y luego se produce una excepción el estado de la memoria no es correcto cuando se reanuda la ejecución.

a. Para el pipeline de múltiples unidades funcionales visto en teórico, brinde un ejemplo en MIPS donde se produzca una excepción luego de haber realizado una escritura en memoria.

b. Investigue y explique qué es un Store Buffer (o Finished Store Buffer - FSB) y cómo soluciona este problema.