Presentación *Práctica 2*

Arquitectura de Computadores

3º de grado en Ingeniería Informática y

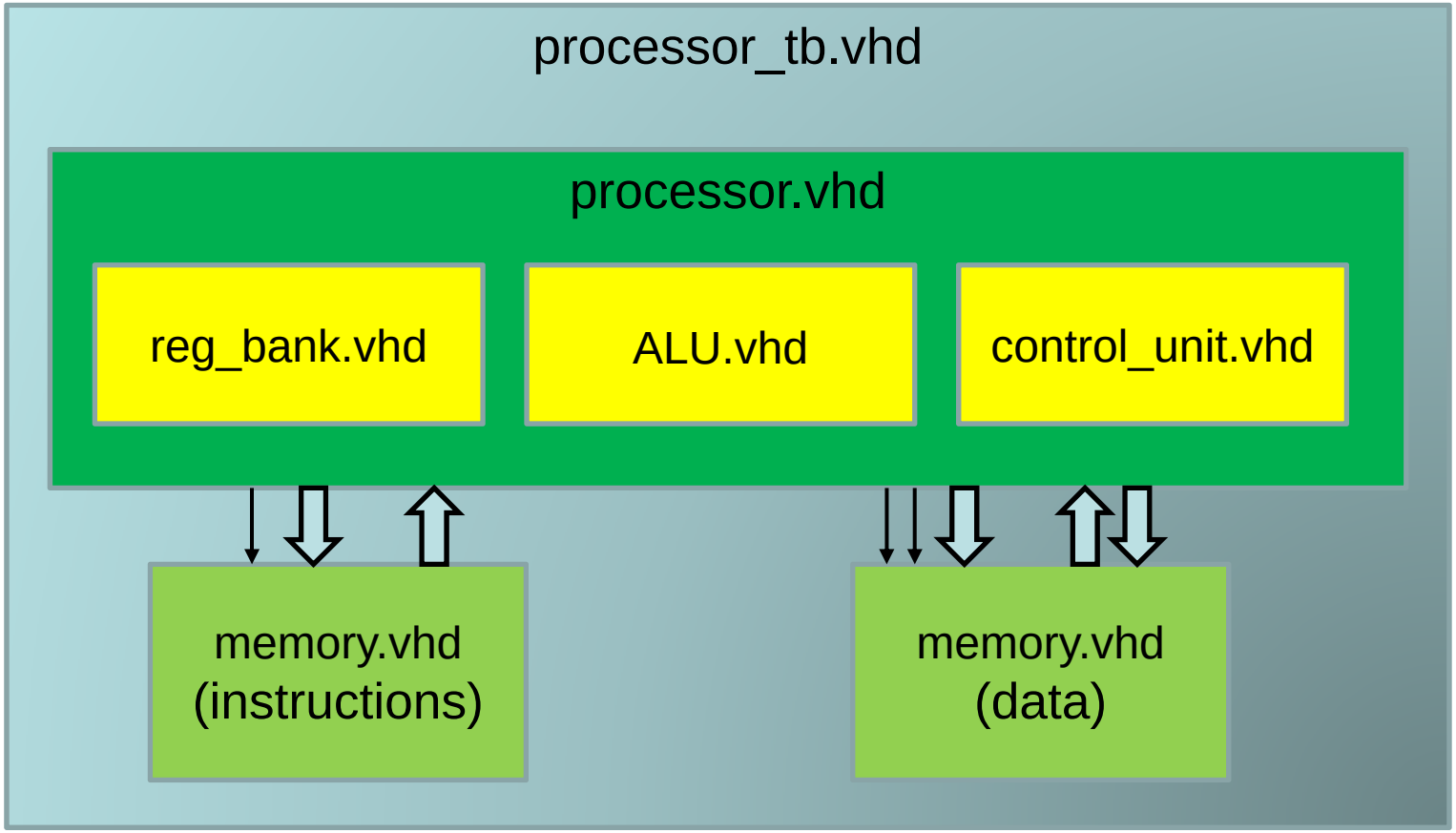
3º de doble grado en Ing. Informática y Matemáticas

Soporte de riesgos de datos y control

- El objetivo de esta práctica es implementar algunas mejoras sobre el microprocesador MIPS para resolver los riesgos (*“hazards”*) generados al segmentar el procesador MIPS.

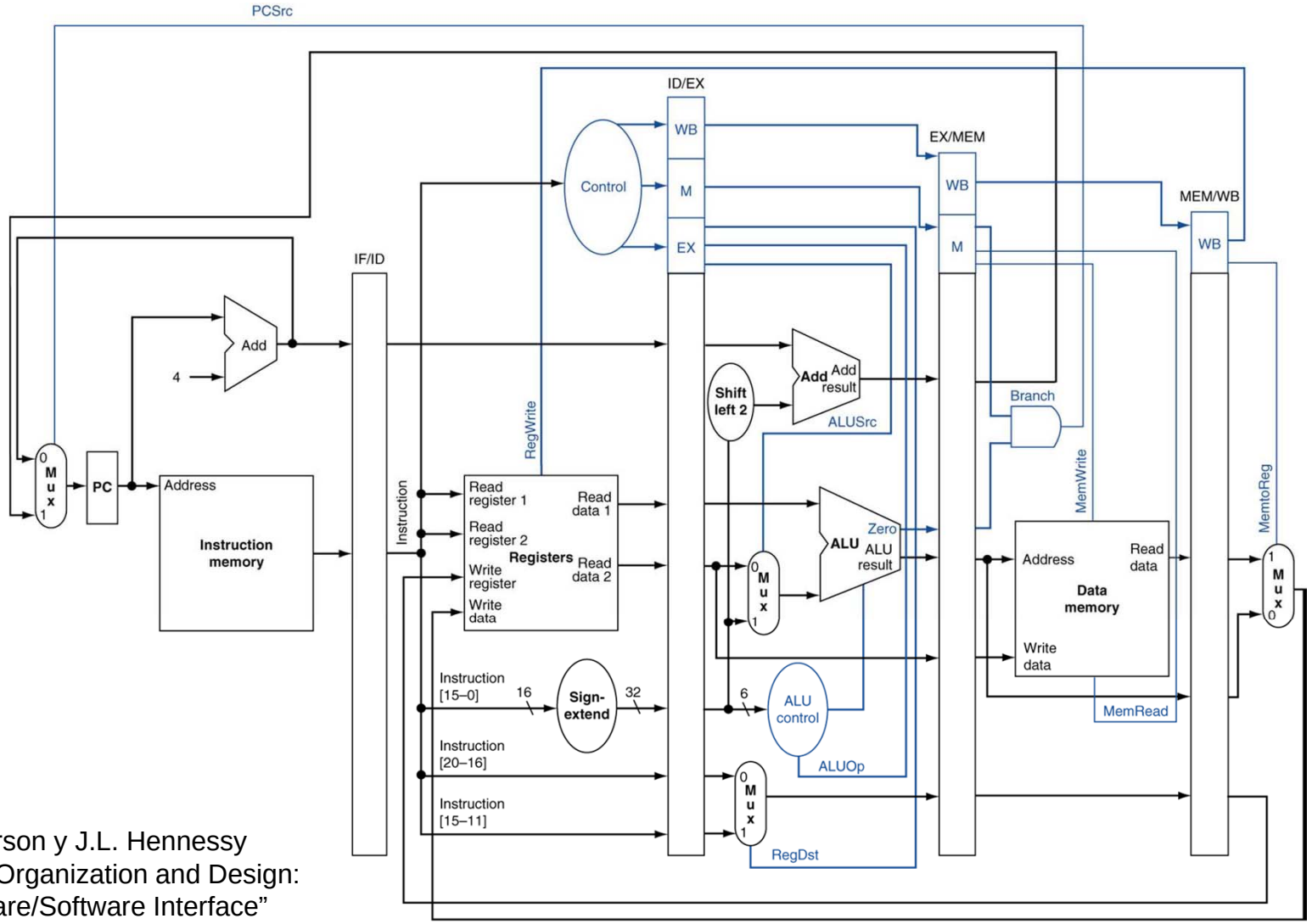
- Ej1: Riesgos de datos (los RAW)
- Ej2: Riesgos de control (saltos)

Se continúa el diseño de la P1



Módulos que se proveen y módulos a desarrollar

MIPS Segmentado (recordatorio)



*D.A. Patterson y J.L. Hennessy
“Computer Organization and Design:
The Hardware/Software Interface”

Ejercicio 1: Riesgos de datos

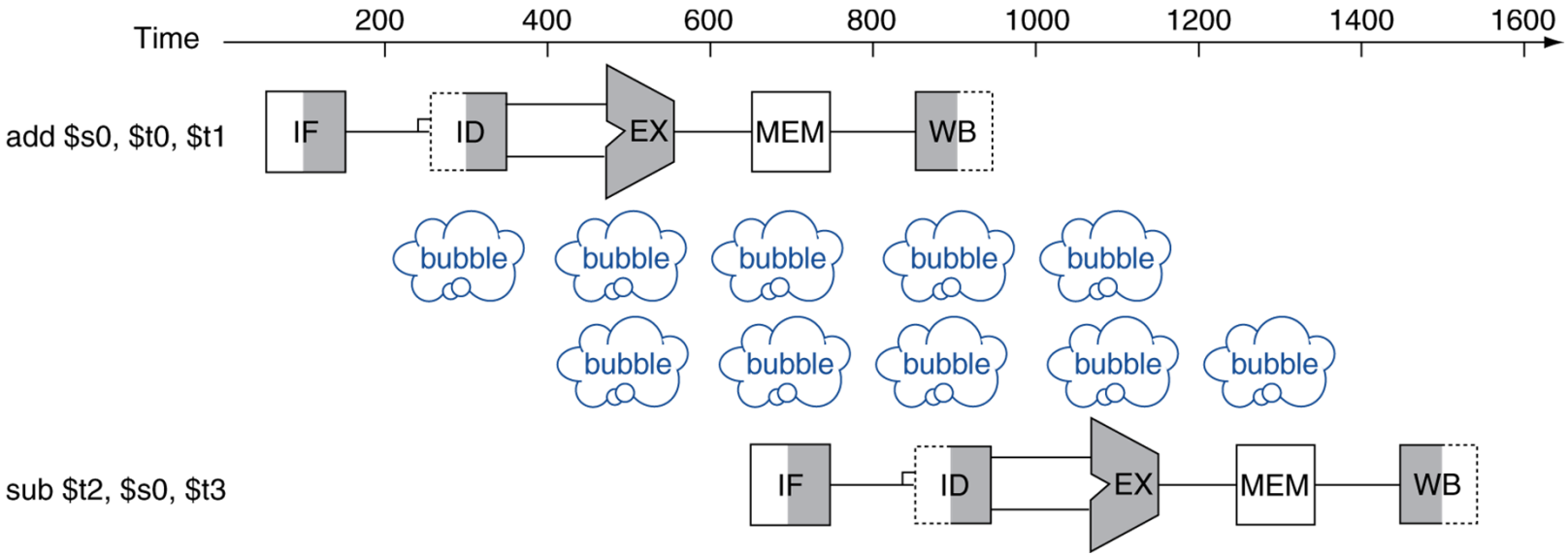
- Se resuelven los riesgos RAW (Read After Write).
  - Situaciones en las que la ejecución de una instrucción depende del resultado de otra anterior aún presente en el pipeline

- Se implementarán 3 mecanismos:
  1. Forwarding de datos hacia la ALU.
  2. Forwarding interno en el banco de registros.
  3. Detección del caso en que una instrucción LW carga un registro que es utilizado por la instrucción que le sigue.

Solución para resolver riesgos RAW: detener el *pipeline*

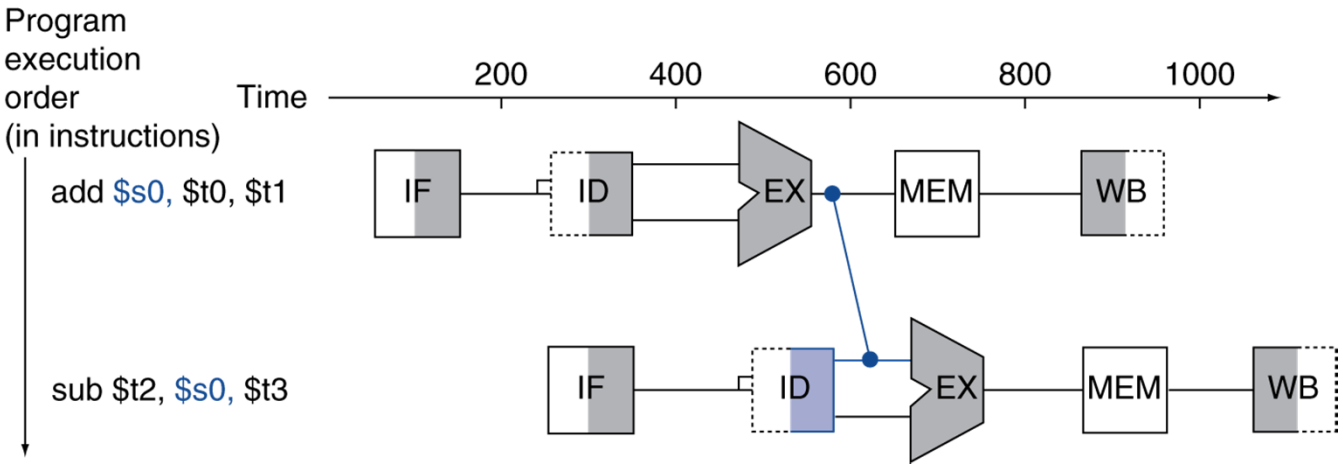
- Como la instrucción que viene detrás depende de la anterior. Una opción es detener el pipeline
- Esta opción genera detenciones innecesarias
- Un ejemplo:

```
add     $s0, $t0, $t1
sub     $t2, $s0, $t3
```



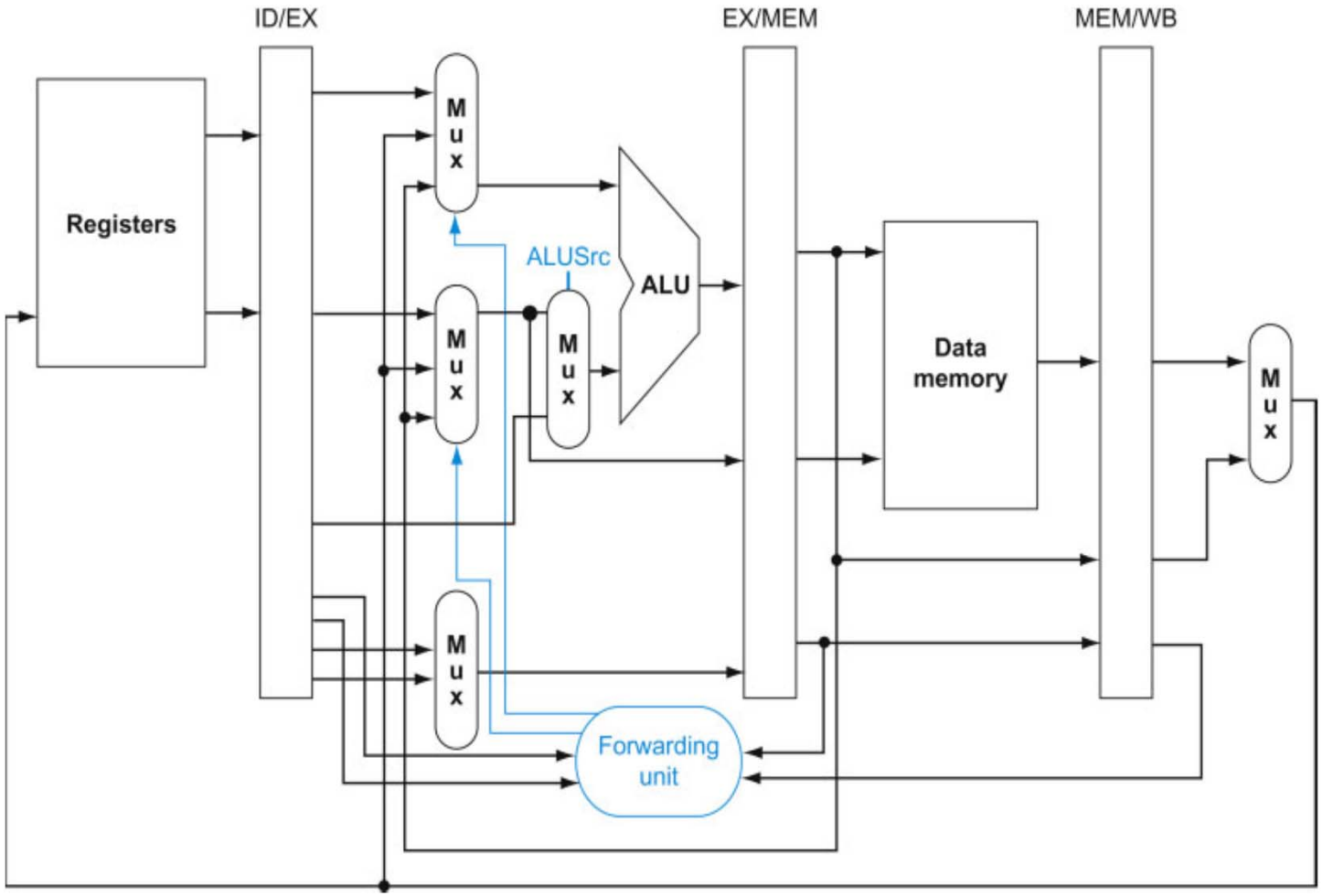
Solución para resolver riesgos RAW: Adelantar el Dato (Forwarding)

- Como el resultado ya está computado, no esperar a que se guarde en los registros.
- Evidentemente requiere conexiones adicionales en la ruta de datos



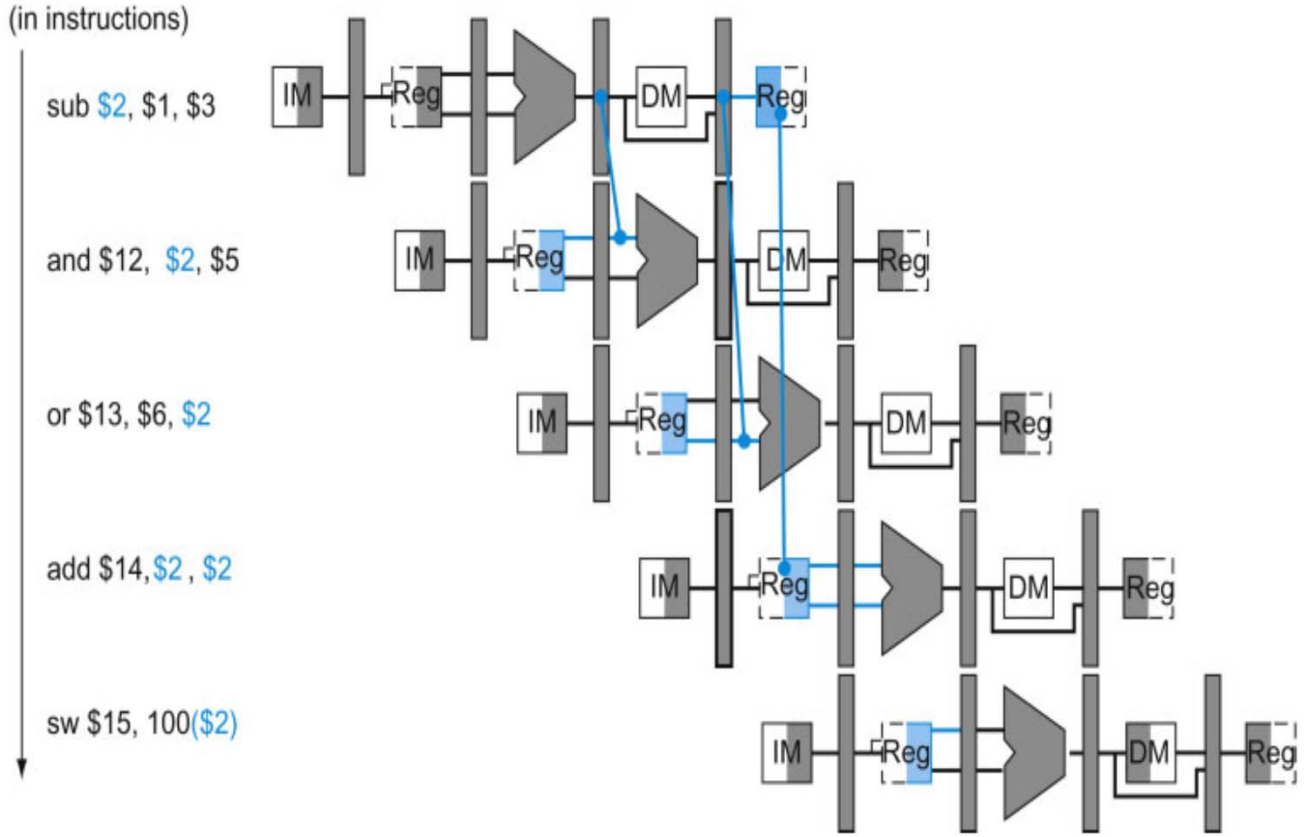
Ej. 1.1 : Forwarding de datos hacia la ALU.

Puedes repasar la lógica necesaria de la *forwading unit* en la teoría o la bibliografía



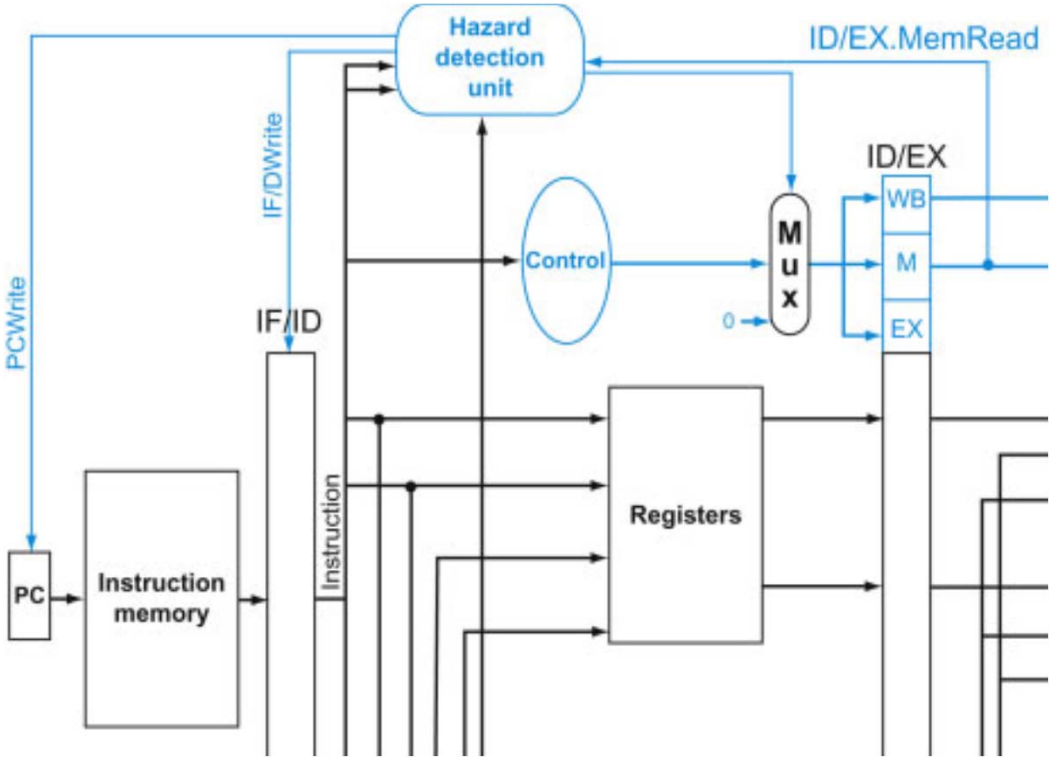
Ej 1.2: Forwarding interno en el banco de registros.

- El adelantamiento desde 3 instrucciones más adelante no está resulto con el banco de registros provisto.
- Realizar los cambios necesarios para que funcione



Ej 1.3: Riesgos Load-Use

- Detección del caso en que una instrucción LW carga un registro que es utilizado por la instrucción que le sigue.
- Necesariamente hay que detener el pipeline



Prueba del Ejercicio 1

- Usar el programa con riesgos provisto y comprobar el funcionamiento.

- Usar el script para compilar el código
  - El código fuente debe ser renombrado a “programa.asm” para ser compilado con el script “arqo_comp.bat”.

Ejercicio 2: Riesgos de Control

- En este ejercicio el procesador debe ser modificado para ejecutar correctamente la instrucción *branch* ante cualquier condición previa del programa.

- En particular, deben realizarse modificaciones en procesador para que la instrucción *branch* se ejecute correctamente después de una operación de tipo ALU así como después de una carga de memoria de datos.

Prueba del Ejercicio 2

- Realizar el código fuente de un programa (fichero en ensamblador) que sirva para comprobr el correcto funcionamiento del *branch* en estos supuestos de riesgo:
  - Una instrucción tipo-R previa modifica un registro y este se usa en un *branch.* Hacer que el salto sea **efectivo**, y que sea **no efectivo**.
  - Una lectura de memoria se guarda en un registro y a continuación se ejecuta un salto condiciona (branch). Hacer que el salto sea **efectivo**, y que sea **no efectivo**.

Recomendaciones

1. Estar seguro de haber corregido todos los fallos en P1. Es muy complejo (casi imposible) depurar cualquier problema partiendo de una P1 con fallos.

2. Documentar el código y usar nombres de señales descriptivos. Mejor seguir nomenclatura del libro

3. Hacer checkeos sintácticos y síntesis antes de simular.

4. Simular y entender que se está simulando

5. Preguntar al profesor lo que no quede claro (para eso está allí)