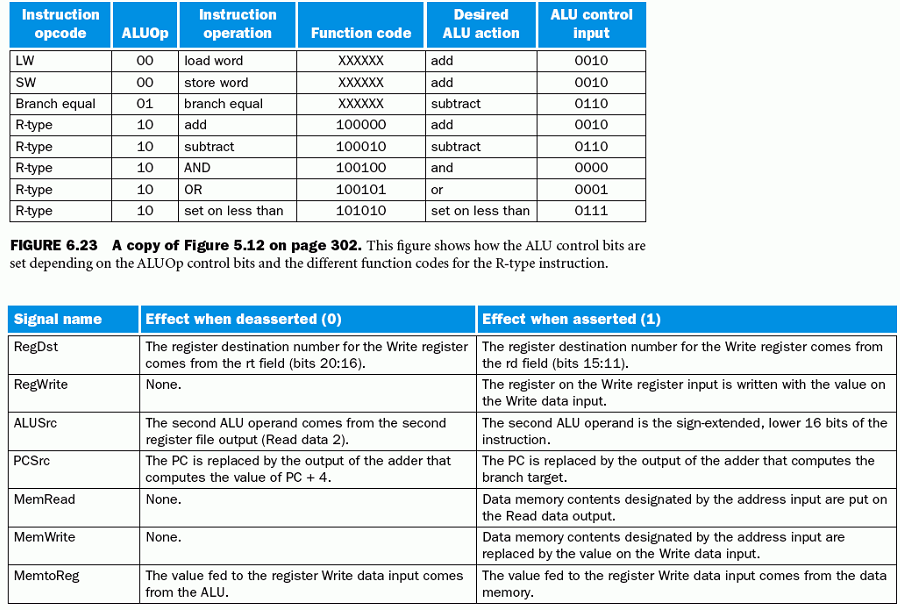
# Figura 6.11 – página 405

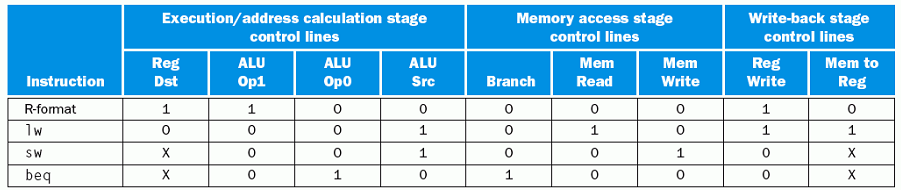
Esta es la que estamos hacienda. Cuando terminemos esto todas las instrucciones deberían funcionar bien, sin la prevención de riesgos de nada.

# Figura 6.17 – página 412

Luego debemos enviar la dirección de destino de una instrucción store a través de los registros del pipeline hasta la última etapa, y de esta tomar el dato, la dirección de destino y la señal para indicar que en la memoria se debe escribir.

# Figura 6.22 – página 417

Esta figura ya tiene incorporadas las señales de control, aunque todavía no se implementó la unidad de control para generarlas de acuerdo a la instrucción. Estaría bueno tenerla andando para controlarlas en un principio manualmente. Las señales de control que debemos implementar son las siguientes:



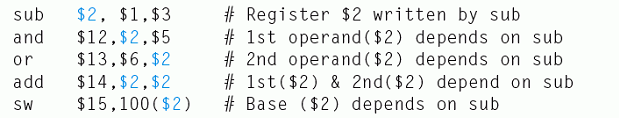


En cada etapa debemos tener en cuenta las siguientes señales de control:

* IF: Ninguna
* ID: Solo existe una señal de control para indicar que se va a escribir en los registros, pero esta señal de control es enviada desde la etapa WB.
* EX: RegDst (selecciona el destino de la operación), ALUOp (selecciona la operación que realizará la ALU), y ALUSrc (selecciona si el segundo operando de la ALU es el data\_b o el inmediato con extensión de signo.
* MEM: Branch (Señal que indica si se trata de una instrucción de salto, y de serlo se compara con el resultado de la ALU para elegir el origen del PC en la etapa IF), MemRead (utilizada para la instrucción load) y MemWrite (utilizada para la instrucción store).
* WB: MemToReg (decide entre enviar el resultado de la ALU, o el valor leído de memoria, al registro destino), RegWrite (escribe el valor elegido).

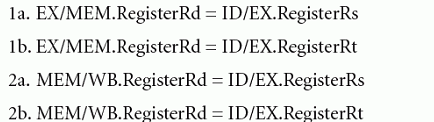
# Figura 6.32 – página 428

Esta ya tiene implementadas las señales de control, con la unidad de control. Supongamos que queremos ejecutar las siguientes instrucciones:



Necesitamos implementar la unidad de cortocircuito, que explicamos a continuación.

Básicamente existen 2 pares de condiciones que pueden significar un riesgo de datos:

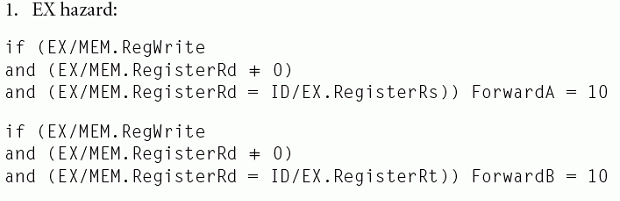


Recién necesitamos conocer el dato con el que debemos operar en la etapa de ejecución. Si uno de los registros fuentes de la etapa de ejecución coincide con el registro destino de la etapa MEM o WB, entonces existirá un riesgo. Debido a que si el dato se encuentra en la etapa MEM o WB, ya fue calculado, podemos adelantar dicho dato a la instrucción siguiente, en la etapa EX, para lo cual debemos agregar multiplexores en esta última etapa de manera tal de poder elegir qué dato tomar, si los leidos de los registros, o los obtenidos de las etapas MEM o WB. Las señales de control de cada multiplexor serán manejadas por la unidad de cortocircuito.

Supongamos ahora que queremos ejecutar las siguientes instrucciones:



Cuando la primera se encuentre en la etapa WB y la segunda en la etapa MEM, el tercer ADD necesitará el dato del registro $1, pero ¿de dónde obtiene ese dato? En realidad debería obtenerlo de la etapa MEM (que es la que contiene el dato más actual). Las condiciones y las señales emitidas por la unidad de cortocircuito serán las siguientes:



2. MEM hazard



# Hazard Detection Unit

## Data hazard pag. 430

Una instrucción trata de leer un registro luego de un load que escribe ese mismo registro.

Para detectarl en la HDU:



Para hacer el stall tengo que evitar que el PC aumente y que el registro IF/ID cambie. El resto del pipeline tiene que hacer nops.

Todas las control signals van a 0 (Figura 6.25), o sea inserto el nop poniendo todos los campos de control EX, MEM Y WB del registro ID/EX en 0.

FIGURA 6.36 pag 433