Computation II - 5EIB0
mMIPS labs 3/4: extending the mMIPS with forwarding

This is a combined assignment for the last **two** labs. In these labs you are going to extend the mMIPS with forwarding. Forwarding allows the mMIPS to use the result of a calculation *before* the result is written to the register file. This greatly reduces the number of data hazards in your mMIPS.

1 Implementing forwarding

The mMIPS which you used in the earlier labs cannot use the result of a calculation until this result is written to the register file. However, the subsequent instruction might need the result of a previous instruction. This gives rise to a data hazard. Consider as an example the following assembler program:

```
addu    t8,s7,s6
subu    t7,s6,s7
addu    t8,t8,t7
```

The first two assembler instructions perform respectively an addition and subtraction. The results of these operations are stored in registers t8 and t7 respectively. The third assembler instruction adds these intermediate results together. In the original mMIPS a hazard occurs between the subtract and the second add instruction since the result of the subtraction is not yet written to register t7. However, the result is already computed and stored in the result register of the ALU. This means we can speed up the mMIPS by creating a bypass that allows the ALU to access its results before they are stored in the register file. This can be done in two ways (See figure 1 and figure 2).

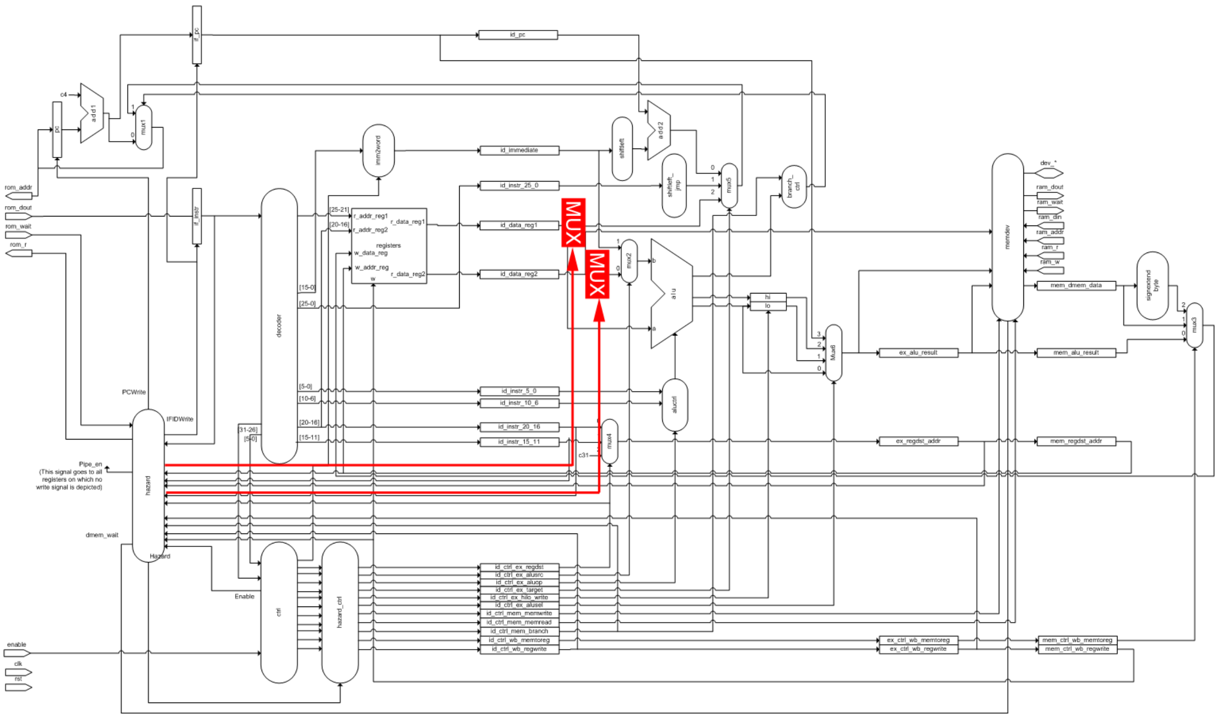

Figure 1: Forwarding option 1

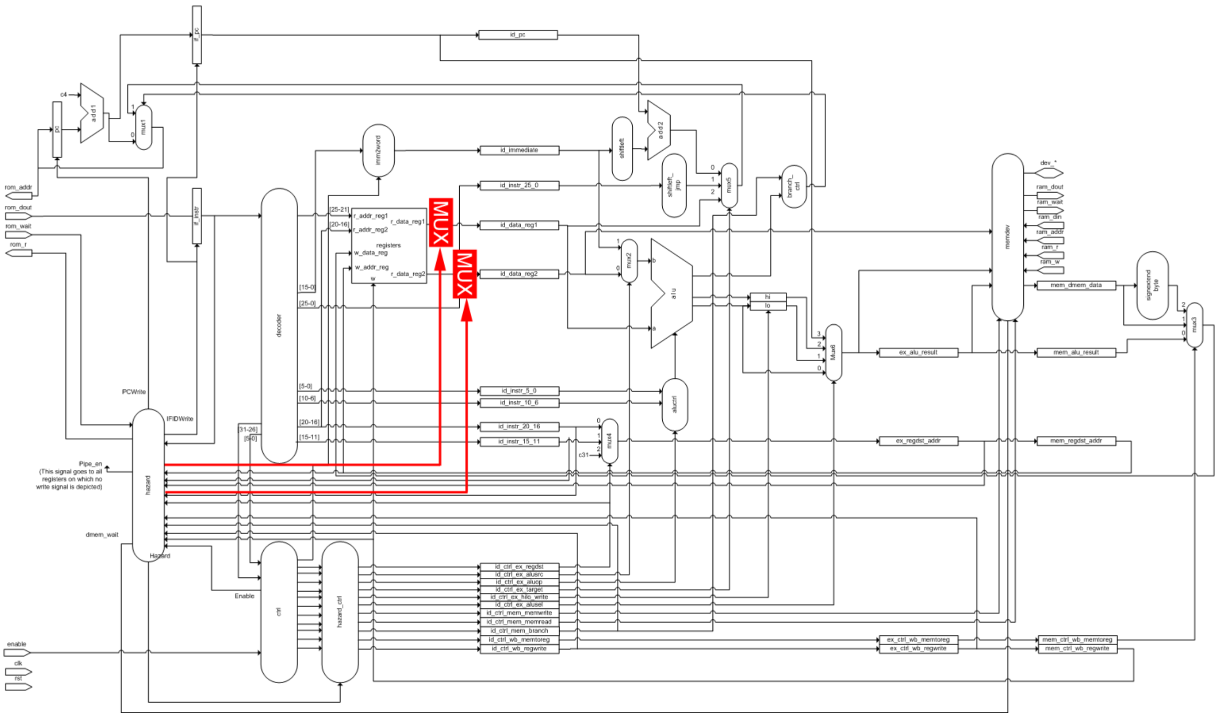

Figure 2: Forwarding option 2

1. How many inputs do the multiplexers need? What signals connect to the multiplexers?

Keep the following in mind when implementing forwarding:

- Register 0 always has a constant value of 0, no matter what will be written in the register. So, never forward register 0!
- Always forward the newest data: EX before MEM, MEM before WB, WB before ID.
- The two register inputs can be forwarded from different or the same pipeline stages. Hence, the two multiplexers should be controlled independently.
- Reading operations (lw and lb) have 1 delay slot (i.e., the instruction after every lw/lb is independent of the result of the lw/lb instruction. If required, NOP is inserted by the compiler to fill this delay slot.). However the data from the memory is not available until the WB stage. As a result, a data hazard can occur between a load instruction in the MEM stage and an instruction in the ID stage. This hazard can not be resolved with forwarding.

Implement forwarding to the mMIPS of lab 2 with the following steps:

1. Create the two forwarding multiplexers in the file `mmips.v`, connect the inputs to the signals that have to be forwarded and connect the output to correct modules (which modules to use depends whether you choose option 1 or 2). Note that there are `mux2`, `mux3` and `mux4` modules available with 2, 3 and 4 input respectively. Use the parameter WIDTH to specify the width of the input and output signals.
2. Create two extra output ports on the hazard module (e.g. `forwardA` and `forwardB`) and two wires. Connect the wires to the `hazard` module and to the select input on the multiplexers. What is the width of the select signal?
3. Modify the hazard module to, instead of inserting NOPs in case of a data hazard, forward the right data (signals). Remember that you still need the data hazard for lw/lb instructions in the MEM stage.

To test and verify your results, it might be useful to write a very short program that includes data hazards. Check whether this program works correctly. If needed, you should correct any errors that you find. If you are convinced that your forwarding to work for this small program, run the larger image processing code. Make sure to compare the output to your reference output to check if your output is bit exact. You can detect bugs in your design by simulating the mMIPS and looking at the signals inside its design. What signals/modules are interesting to look at?

Below you can find a short example that shows how to change the hazard detection in the WB stage such that data can be forwarded from this stage:
Original hazard detection code:

```
else if (MEMWBRegWrite == 1’b1 && (
            MEMWBWriteRegister == ifidreadregister1 ||
            MEMWBWriteRegister == ifidreadregister2))
    // WB hazard
    hazard = 1’b1;
```

When forwarding from the WB stage, you should change the code to:

```
if (MEMWBRegWrite == 1’b1 &&
            MEMWBWriteRegister == ifidreadregister1 &&
            MEMWBWriteRegister != 0)
    // Forward A from WB
    forwardA = 2’b11; //Assuming 2’b11 selects WB forwarding signal

if (MEMWBRegWrite == 1’b1 &&
            MEMWBWriteRegister == ifidreadregister2 &&
            MEMWBWriteRegister != 0)
    // Forward B from WB
    forwardB = 2’b11; //Assuming 2’b11 selects WB forwarding signal
```