Forwarding Control Details in the MIPS Pipeline

As shown in the figure below, a logic unit must be added to enable forwarding of operands:

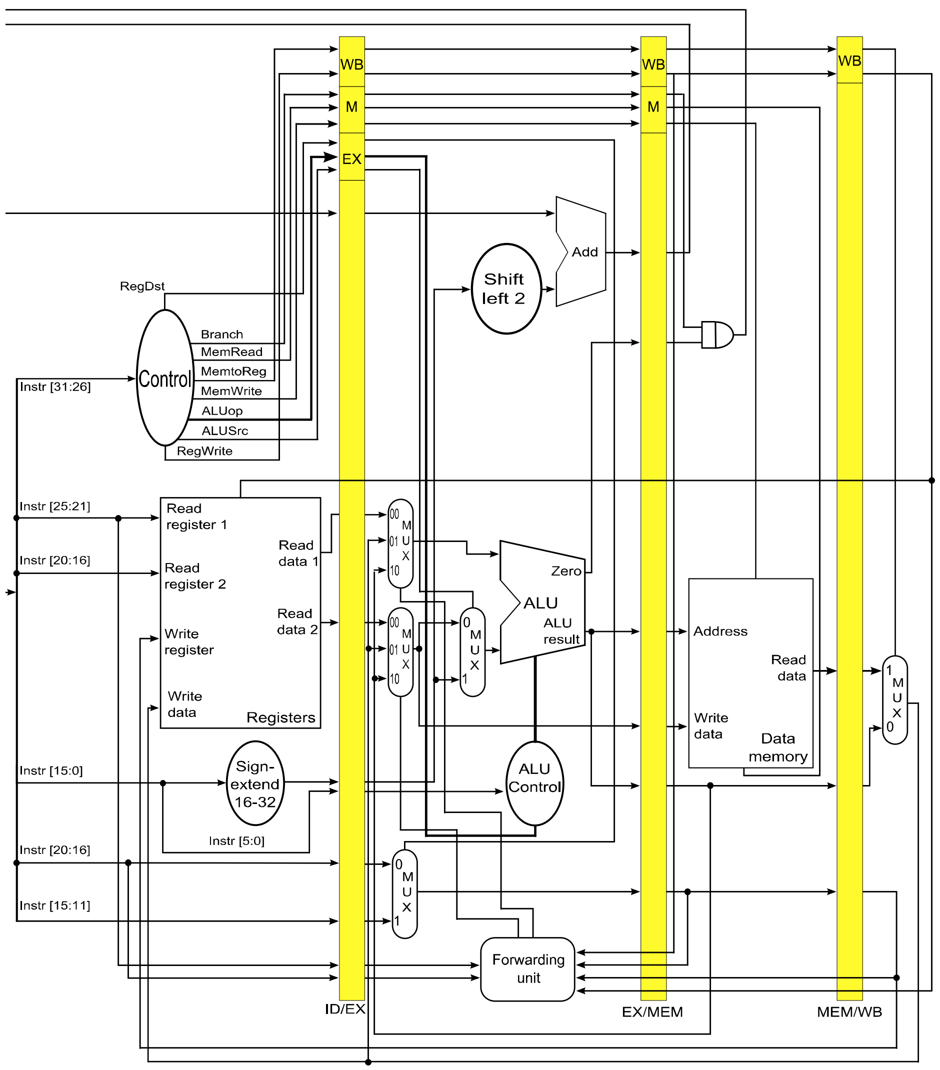


Fig 1: The Forwarding Unit within the MIPS pipeline

The forwarding unit takes a total of six input values, and produces two output values:

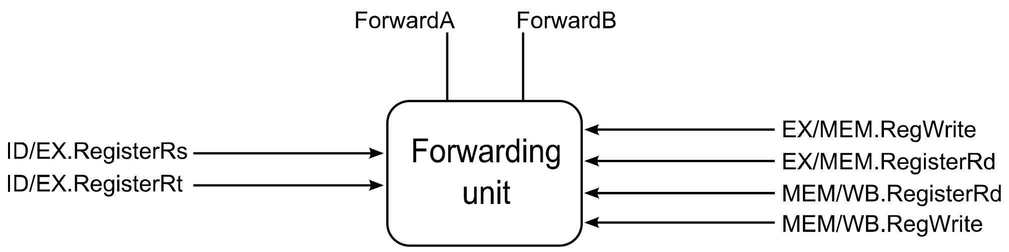


Fig 2: The Forwarding Unit Interface

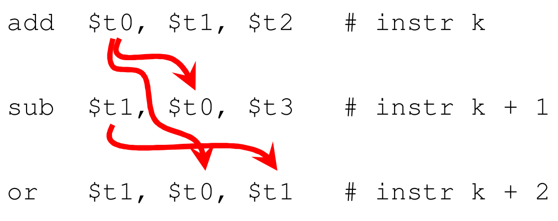
The purpose of the forwarding unit is to guarantee that the instruction entering the EX stage of the pipeline receives the correct values for its register operands.

The input/output values are characterized as:

Input Value	Width	Description
EX/MEM.RegisterRd	5 bits	destination register for instruction k + 1
MEM/WB.RegisterRd	5 bits	destination register for instruction k
EX/MEM.RegWrite	1 bit	RegWrite setting for instruction k + 1
MEM/WB.RegWrite	1 bit	RegWrite setting for instruction k
ID/EX.RegisterRs	5 bits	left operand for instruction k + 2
ID/EX.RegisterRt	5 bits	right operand for instruction k + 2

Output Value	Width	Description
ForwardA	2 bits	controls selection of left operand to ALU
ForwardB	2 bits	controls selection of right operand to ALU

For example, given the following sequence of instructions, the six input values would be as shown below:

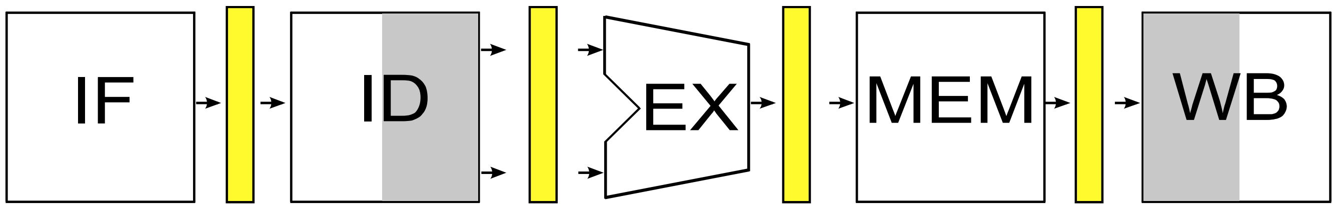


```
add   $t0, $t1, $t2    # instr k

sub   $t1, $t0, $t3    # instr k + 1

or    $t1, $t0, $t1    # instr k + 2
```

EX/MEM.Rd	01001
MEM/WB.Rd	01000
EX/MEM.RegWrite	1
MEM/WB.RegWrite	1
ID/EX.RegisterRs	01000
ID/EX.RegisterRt	01001



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

                     or  $t1, $t0, $t1    sub $t1, $t0, $t3    add $t0, $t1, $t2

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

In this example, the result computed by instruction k must be forwarded to instruction k + 1 when it enters the EX stage, and to instruction k + 2 when it enters the EX stage; and, the result computed by instruction k + 1 must be forwarded to instruction k + 2, when it enters the EX stage.

The forwarding unit sets the control signals for the two multiplexors shown to the left of the ALU (Fig 1), called ForwardA for the top multiplexor (corresponding to the left operand to the ALU) and ForwardB for the lower multiplexor (corresponding to the right operand for the ALU). You should make sure you also read the relevant sections of P&H.

For this assignment, you are to design the internal circuitry of the Forwarding Unit and then implement your design using Logisim. You should consider the possibilities for a modular implementation of the internal circuitry; not only is that an appropriate design tactic, but it also will lead to a cleaner implementation that is easier for you to test. In particular, you might want to view the logic for each of the inter-stage units as a separate module (although one will depend on the other).

The design of the logic circuitry requires producing appropriate truth tables, showing how the six input values determine the two output values, and then generating a Boolean function that expresses each truth table. Although it is not mandatory, you might find it best to represent those functions in sum-of-products form.

For the first part of your solution, you will create a plain text file that contains your truth tables (with all columns clearly labeled) and the corresponding Boolean functions. You should format the contents of this file neatly, and label everything clearly.

The second part of your solution will be a single Logisim file (`.circ`) containing an implementation of your design. You are restricted to using the Base and Gates libraries in Logisim (which are entirely sufficient for this assignment). Your solution must take six inputs and produce two outputs, as described above. All inputs and outputs must be clearly labeled.

In order to make it easier to evaluate your implementation, you are required to lay out the interface as shown in Fig 3 below:

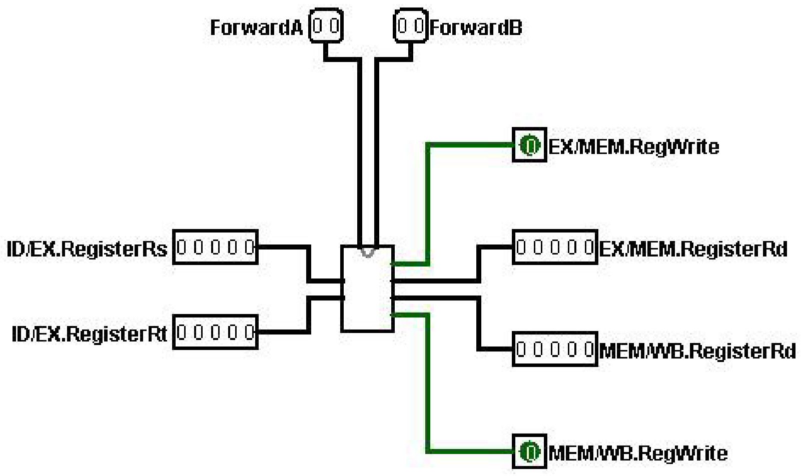


Fig 3: Logisim Interface for the Forwarding Unit

What to turn in and how:

Submit a single tar file containing your text document and your Logisim file. Submit nothing else.

The TAs will evaluate your submission and apply a grading rubric to score it..

Instructions, and the appropriate link, for submitting to the Curator are given in the *Student Guide* at the Curator website:

http://www.cs.vt.edu/curator/.

You will be allowed to submit your solution multiple times; the last submission will be evaluated.

Pairs Work:

For this assignment, you are allowed, but not required, to work with one other student to derive a solution. If you do work with a partner, only one of you should submit your final solution to the Curator, and you should make sure that your text file lists both of your names and PIDs. If you begin working on the assignment as part of a pair, and then decide to break up that pair, you must complete the assignment on your own.

Pledge:

Include the appropriate pledge statement from the course website in the text file you submit.