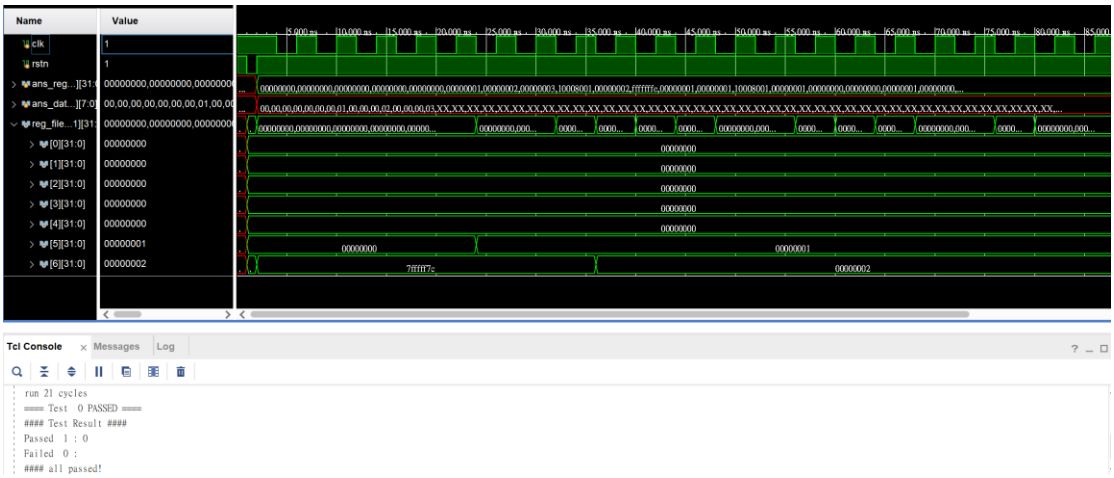
Lab4: Pipelined Processor-Part2

111550142 尤瑋辰

1. Experimental Result

   a. Show the waveform screen shot of the test we provided.



   b. What other cases you've tested? Why you choose them?

```
        .data   0x10008000      # start of Dynamic Data (pointed by $gp)
        .word   0x0             # 0($gp)
        .word   0x1             # 4($gp)
        .word   0x2             # 8($gp)
        .word   0x3             # 12($gp)
        .word   0x0             # 16($gp)
        .text   0x00400000      # start of Text (pointed by PC),
                                # Be careful there might be some other instructions in JsSPIM.
                                # Recommend at least 9 instructions to cover out those other instructions.
main:   addi    $v1, $v1, 0x1111 # v3 = v3 + 0x1111
        lw      $a0, 0($gp)
        lw      $a1, 4($gp)
        sw      $v1, 16($gp)    # save word v3 to gp+16, since addi result has been write to v3
        add     $t0, $a1, $gp   # t0 = gp, since `lw $a1` is at ID
        lw      $a2, 8($gp)
        add     $t1, $a1, $a1   # t1 = 2, since `lw $a1` is at MEM (when this enter ID, lw enter WB)
        lw      $a3, 12($gp)
        sub     $t2, $0, $t1    # t2 = 0, since `add $t1` is at MEM
        beq     $v1, $0, end    # not taken
        slt     $t3, $a2, $a3   # t3 = 1, Read after write need to wait for ? more instructions?
        or      $t5, $gp, $t3   # t5 = gp, since t3 has not been written yet
        lw      $t4, 4($gp)
        and     $t6, $a1, $t3   # t6 = 1, since t3 is written in this cycle
        beq     $t3, $t4, end   # taken
        addi    $s1, $0, 0x0001 # will this executed even branch is taken? How about others?
        sw      $gp, 0($gp)     # gp =  0x10008000;
        addi    $s3, $0, 0x0003
        add     $s4, $a2, $a2
        add     $s5, $a2, $a3
        add     $s6, $a3, $a3
end:    add     $s7, $s4, $a2   # s7 = 2
```

This case is from Lab3, it tests all types of instructions.

```
20 0e 00 01   // [0040002c] addi$14, $0 , 0x0001
11 cb 00 06   // [00400030] beq $14, $11, 24 [end-0x00400030
```

```
00 ab 70 24  // [0040002c] and $14, $5, $11                    ;
11 6e 00 06  // [00400030] beq $11, $14, 24 [end-0x00400030]

00 ab 70 24  // [0040002c] and $14, $5, $11
11 cb 00 06  // [00400030] beq $14, $11, 24 [end-0x00400030]

8f 8e 00 04  // [0040002c] lw $14, 4($28)                      ;
11 6e 00 06  // [00400030] beq $11, $14, 24 [end-0x00400030]

8f 8e 00 04  // [0040002c] lw $14, 4($28)                      ;
11 cb 00 06  // [00400030] beq $14, $11, 24 [end-0x00400030]

8f 8c 00 04  // [00400030] lw $12, 4($28)
00 ab 70 24  // [00400034] and $14, $5, $11
11 6c 00 06  // [00400038] beq $11, $12, 24[end-0x00400030]
```

The above cases test the four types of data hazards for branch instructions. The first case is an immediate instruction before 'beq', which needs 1 stall cycle. The second case is an ALU instruction before 'beq', which needs 1stall cycle. The third case is a load instruction before 'beq', which needs 2 stall cycles. Finally, the fourth case is a load instruction 1 cycle before 'beq', which needs 1 stall cycle.

2. Answer the following Questions:
   1. List out the equation to detect EX & MEM hazard in forwarding unit. Which part of the equation in textbook p. 369 is wrong?

      EX hazard:

$$if(EX/MEM.RegWrite\ and\ (EX/MEM.Rd\ \neq 0)$$
$$and\ (EX/MEM.Rd\ = ID/EX.Rs))\quad ForwardA = 10$$
$$if(EX/MEM.RegWrite\ and\ (EX/MEM.Rd\ \neq 0)$$
$$and\ (EX/MEM.Rd\ = ID/EX.Rt))\quad ForwardB = 10$$

      MEM hazard:

$$if(MEM/WB.RegWrite\ and\ (MEM/WB.Rd\ \neq 0)$$
$$and\ not\ (EX/MEM.RegWrite\ and\ (EX/MEM.Rd\ \neq 0)$$
$$and\ (EX/MEM.Rd\ = ID/EX.Rs))$$
$$and\ (MEM/WB.Rd\ = ID/EX.Rs)\quad ForwardA = 01$$

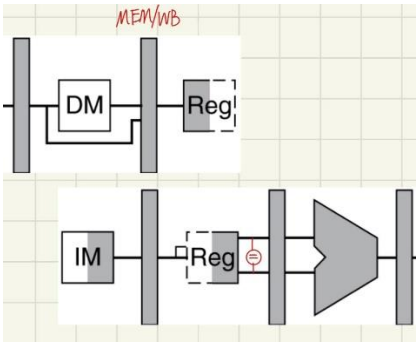
$$if(MEM/WB.RegWrite\ and\ (MEM/WB.Rd\ \neq 0)$$
$$and\ not\ (EX/MEM.RegWrite\ and\ (EX/MEM.Rd\ \neq 0)$$
$$and\ (EX/MEM.Rd\ = ID/EX.Rt))$$
$$and\ (MEM/WB.Rd\ = ID/EX.Rt)\quad ForwardB = 01$$

```
if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd ≠ 0)
and not(EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0))
    or and (EX/MEM.RegisterRd ≠ ID/EX.RegisterRs)
and (MEM/WB.RegisterRd = ID/EX.RegisterRs)) ForwardA = 01

if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd ≠ 0)
and not(EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0))
    or and (EX/MEM.RegisterRd ≠ ID/EX.RegisterRt)
and (MEM/WB.RegisterRd = ID/EX.RegisterRt)) ForwardB = 01
```

The complement of the EX hazard is incorrect. The "AND" circled in red pen should be "OR."

2. In forwarding for beq, is forwarding from MEM/WB to ID needed? Why?
   There is no need to forward these values from MEM/WB to ID. Because the results from MEM/WB are already written back to the register file by the time the 'beq' instruction needs them.



3. Briefly explain how you insert 2 stalls when beq reads registers right after lw writes it.

```
// stall 3: Load before Beq  => stall 2 bubbles
// IF_ID_Branch && ID_EX_MemRead and ID_EX_rt !=0 and (ID_EX_rt == IF_ID_rs or ID_EX_rt == IF_ID_rt);
wire beq_stall_3;
assign beq_stall_3 = control_branch && ID_EX_mem_read && (ID_EX_reg_file_write_reg1 != 0)
                  && ((ID_EX_reg_file_write_reg1 == reg_file_read_reg_1) || (ID_EX_reg_file_write_reg1 == reg_file_read_reg_2));

// stall 4: Load 1 cycle before Beq  => stall 1 bubble
// IF_ID_Branch && EX_MEM_MemRead and EX_MEM_rt !=0 and (EX_MEM_rt == IF_ID_rs or EX_MEM_rt == IF_ID_rt);
wire beq_stall_4;
assign beq_stall_4 = control_branch && EX_MEM_mem_read && (EX_MEM_reg_file_write_reg != 0)
                  && ((EX_MEM_reg_file_write_reg == reg_file_read_reg_1) || (EX_MEM_reg_file_write_reg == reg_file_read_reg_2));
```

I've split the problem into two parts. In the first part, we stall one cycle if 'beq' is right after 'lw'. In the second part, we stall one cycle if 'beq' is the second instruction right after 'lw'. After finishing the first part, in the next cycle, the condition will be the second part. Sometimes, only the second part occurs. By splitting the problem, we can handle these two conditions.

In detail, the way to detect the first condition is by the following expression:

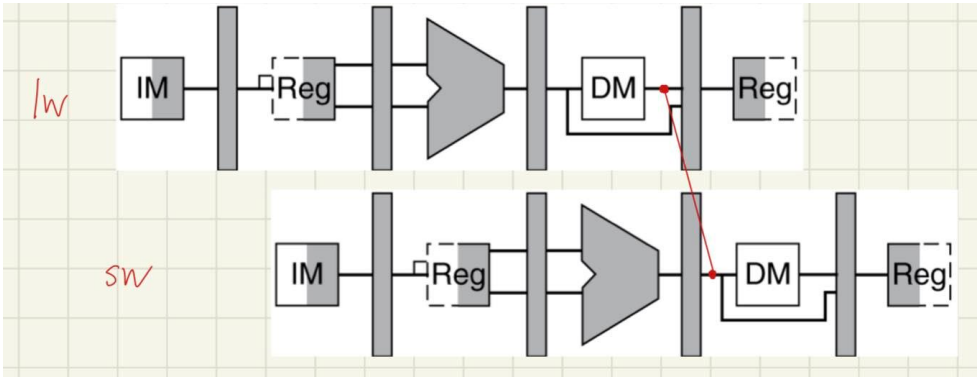
$$if(IF/ID\_Branch\ and\ ID/EX.MemRead\ and\ (ID/EX.Rt \neq 0)$$
$$and\ ((ID/EX.Rt = IF/ID.Rs)\ or\ (ID/EX.Rt = IF/ID.Rt)))$$

The second condition:

$$if(IF/ID\_Branch\ and\ EX/MEM.MemRead\ and\ (EX/MEM.Rt \neq 0)$$
$$and\ ((EX/MEM.Rt = IF/ID.Rs)\ or\ (EX/MEM.Rt = IF/ID.Rt)))$$

4. sw right after lw is quite common since copy and paste a data from one address to another is used frequently. In textbook, a stall is followed by the lw in this case. Is it possible to remove this stall? How?

Yes. But it needs additional forwarding mechanism as follows:



The detail will be like:

$$if(MEM/WB.MemRead\ and\ EX/MEM.MemWrite$$
$$and\ (MEM/WB.Rt = EX/MEM.Rt))\quad Forward = 1$$