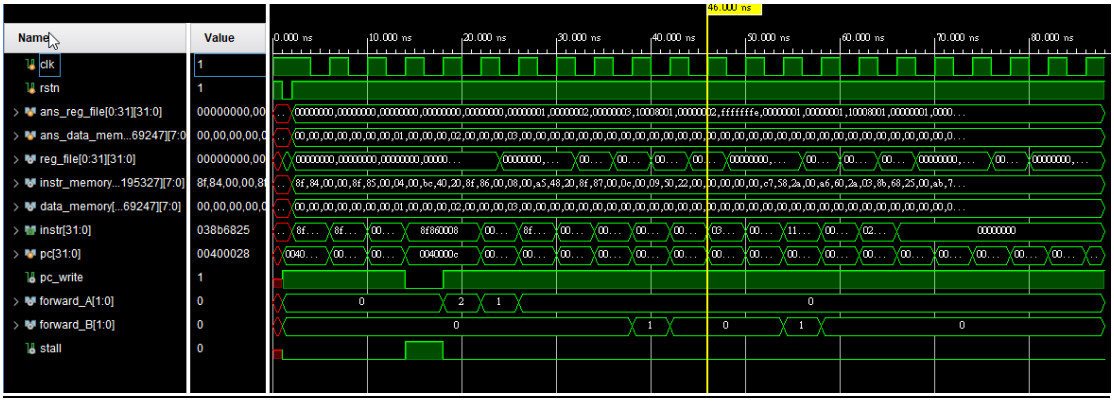
2024 spring Computer Organization

Lab4 Pipeline Processor Part 2

111550168 陳奕

- Experimental result

1. Waveform



2. Other cases

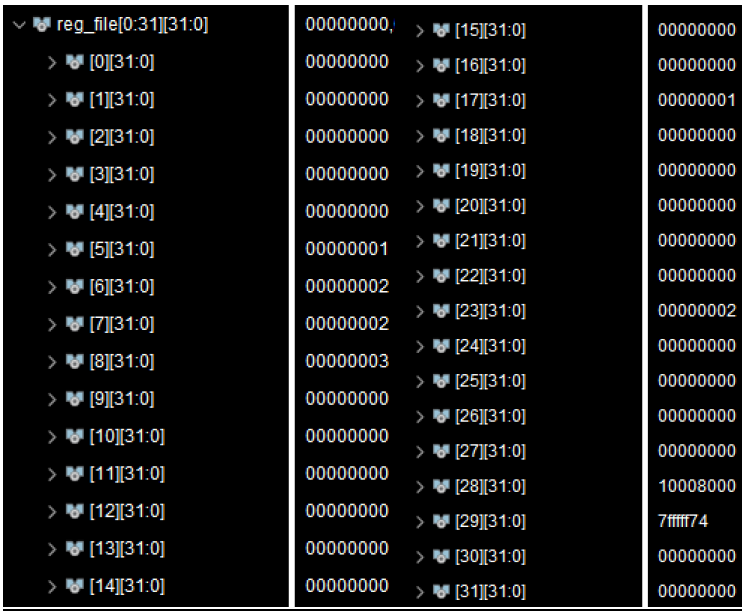
```
8f 85 00 04  // [00400000]  lw $5, 4($28)
8f 86 00 08  // [00400004]  lw $6, 8($28)
8f 87 00 0c  // [00400008]  lw $7, 12($28)
af 87 00 10  // [0040000c]  sw $7, 16($28) #lw->sw: load-use hazard, stall 1 cycle and forward from MEM/WB stage
00 a6 40 20  // [00400010]  add $8, $5, $6
af 88 00 14  // [00400014]  sw $8, 20($28) #alu operation->sw forward form MEM/WB stage
8f 87 00 08  // [00400018]  lw $7, 8($28)
10 E6 00 02  // [0040001c]  beq $7, $6, 2  #lw->beq :load-use hazard, stall 2 cycles and forward from MEM/WB stage
00 05 88 20  // [00400020]  add $17, $0, $5
00 06 90 20  // [00400024] add $18, $0, $6
03 a6 b8 20  // [00400028]  add $23, $29, $6
```

I tested the following three hazard scenarios:

load-use hazard (lw -> sw): When a load instruction (lw) is followed by a store instruction (sw) using the same register, this requires a cycle stall and data forwarding from the MEM/WB stage.

ALU operation forwarding (add -> sw): When an ALU operation instruction (add) is immediately followed by a store instruction (sw) using the calculation result, this requires data forwarding from the MEM/WB stage.

load-use hazard (lw -> beq): When a load instruction (lw) is followed by a branch instruction (beq) using the same register, this requires a two-cycle stall and data forwarding from the MEM/WB stage.



The result is identical to the outcome with jsSpim

- 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?

   MEM hazard:

   ALU operand A :

   Forward_A = 2'b01

   if (MEM_WB_reg_write && (MEM_WB_rd != 0) && (MEM_WB_rd == ID_EX_rs) && !(EX_MEM_reg_write && (EX_MEM_rd != 0) && (EX_MEM_rd == ID_EX_rs)))

   ALU operand B :

   Forward_B = 2'b01

   if (MEM_WB_reg_write && (MEM_WB_rd != 0) && (MEM_WB_rd == ID_EX_rt) &&!(EX_MEM_reg_write && (EX_MEM_rd != 0) && (EX_MEM_rd == ID_EX_rt)))

   EX hazard：

   ALU operand A :

   Forward_A = 2'b10

   if (EX_MEM_reg_write && (EX_MEM_rd != 0) && (EX_MEM_rd == ID_EX_rs))

   ALU operand B :

Forward_B = 2’b10

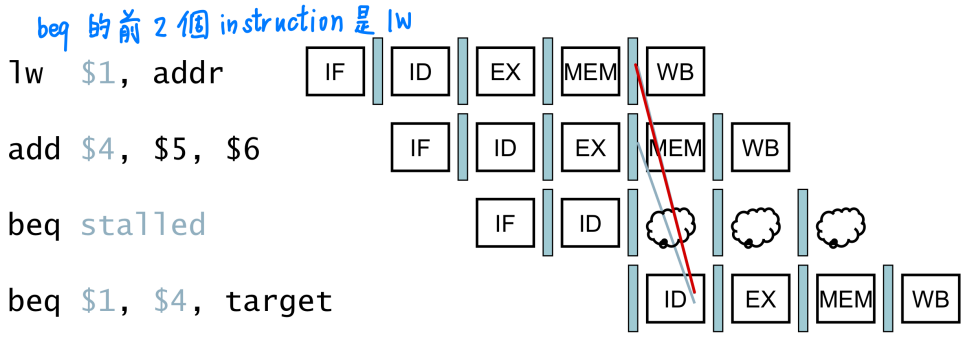
if (EX_MEM_reg_write && (EX_MEM_rd != 0) && (EX_MEM_rd == ID_EX_rt))

This is the screenshot in textbook p.369, the highlighter marked the wrong place, symbol ‘!=’ should be ‘=’

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

2. In forwarding for beq, is forwarding from MEM/WB to ID needed? Why?

   Yes. In beq, the branch decision depends on comparing two registers. If the values of these registers are being written back by a preceding load instruction (lw), those values need to be forwarded to the ID stage to ensure correct comparison. This circumstance show how it is needed:



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

   First Stall:

   Detect the load-use hazard where the source registers of the beq instruction are the same as the destination register of the previous lw instruction.

   Second Stall:

   I use this equation to detect second stall in hazard detection module:

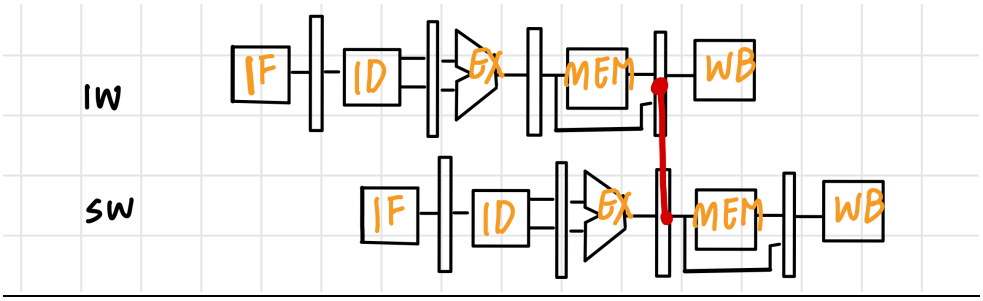
   assign branch_data_hazard_mem = EX_MEM_mem_read && cur_branch && ((EX_MEM_rd == IF_ID_rs) || (EX_MEM_rd == IF_ID_rt));

   Because beq requires accurate register values for the comparison, and these values are coming from the EX_MEM stage.

When stalling, set the control signals to insert a stall into the pipeline. This involves preventing the PC and IF/ID registers from updating and setting control signals in the ID/EX register to zero.

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, it is possible to remove the stall by forwarding the data directly from the MEM stage to the EX stage._



Forwarding Logic:

forward_sw_data = MEM/WB.WriteData

if (MEM/WB.RegWrite && (MEM/WB.RegisterRd != 0) &&

(MEM/WB.RegisterRd == EX/MEM.RegisterRt))