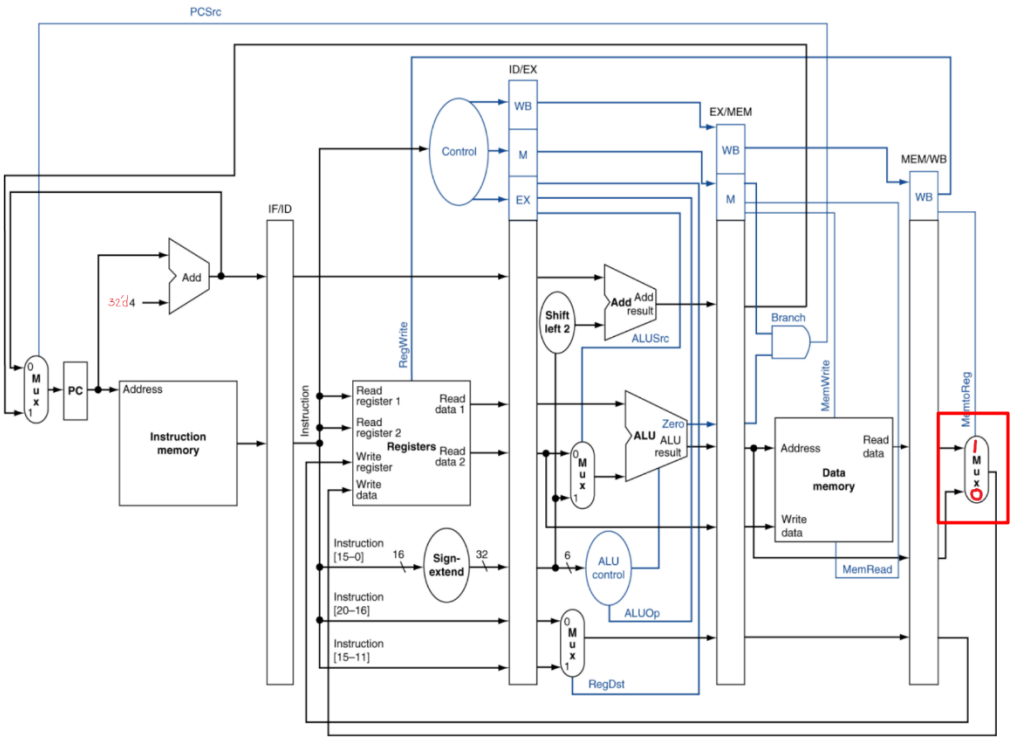
Computer Organization Lab4

ID: 110550085 Name: 房天越

Architecture diagrams:



This diagram is almost the same as the diagram given in the spec, the only difference is that the data0 and data1 are reversed in the red block.

Hardware module analysis:

In this Lab, I try to implement the CPU in a pipelined way, so we separate the CPU into 5 stages which are IF, ID, EX, MEM and WB. In each stage, store the signals that need to be transferred to the next stage into a register(Pipe_Reg), and then transfer the signals from the register together.

Compared with Lab3, I no longer need to implement some commands, such as jump, jal, and jr, etc. So, I remove some of the signals in the decoder, such as jump and branchtype to make it simpler. Also, I no longer need use multiplier for 3 and 4 inputs, so I reduce some signals that are generated by the decoder to one bit.

Simulation results:

Testcase1:

```
============== Final Result ==============
- Register File -
 r0 =      0  r1 =      3  r2 =      4  r3 =      1
 r4 =      6  r5 =      2  r6 =      7  r7 =      1
 r8 =      1  r9 =      0 r10 =      3 r11 =      0
r12 =      0 r13 =      0 r14 =      0 r15 =      0
r16 =      0 r17 =      0 r18 =      0 r19 =      0
r20 =      0 r21 =      0 r22 =      0 r23 =      0
r24 =      0 r25 =      0 r26 =      0 r27 =      0
r28 =      0 r29 =      0 r30 =      0 r31 =      0

- Memory Data -
 m0 =      0  m1 =      3  m2 =      0  m3 =      0
 m4 =      0  m5 =      0  m6 =      0  m7 =      0
 m8 =      0  m9 =      0 m10 =      0 m11 =      0
m12 =      0 m13 =      0 m14 =      0 m15 =      0
m16 =      0 m17 =      0 m18 =      0 m19 =      0
m20 =      0 m21 =      0 m22 =      0 m23 =      0
m24 =      0 m25 =      0 m26 =      0 m27 =      0
m28 =      0 m29 =      0 m30 =      0 m31 =      0
```

Testcase2:

```
============== Final Result ==============
- Register File -
 r0 =      0  r1 =      0  r2 =      4  r3 =      5
 r4 =     49  r5 =      0  r6 =      3  r7 =      5
 r8 =      1  r9 =      0 r10 =      7 r11 =      7
r12 =      0 r13 =      0 r14 =      0 r15 =      0
r16 =      0 r17 =      0 r18 =      0 r19 =      0
r20 =      0 r21 =      0 r22 =      0 r23 =      0
r24 =      0 r25 =      0 r26 =      0 r27 =      0
r28 =      0 r29 =      0 r30 =      0 r31 =      0

- Memory Data -
 m0 =      0  m1 =      7  m2 =      0  m3 =      0
 m4 =      0  m5 =      0  m6 =      0  m7 =      0
 m8 =      0  m9 =      0 m10 =      0 m11 =      0
m12 =      0 m13 =      0 m14 =      0 m15 =      0
m16 =      0 m17 =      0 m18 =      0 m19 =      0
m20 =      0 m21 =      0 m22 =      0 m23 =      0
m24 =      0 m25 =      0 m26 =      0 m27 =      0
m28 =      0 m29 =      0 m30 =      0 m31 =      0
```

Testcase3(Hazard):

```
============== Final Result ==============
- Register File -
 r0 =     0  r1 =    16  r2 =    20  r3 =     8
 r4 =    16  r5 =     8  r6 =    24  r7 =    26
 r8 =     8  r9 =   100 r10 =     0 r11 =     0
r12 =     0 r13 =     0 r14 =     0 r15 =     0
r16 =     0 r17 =     0 r18 =     0 r19 =     0
r20 =     0 r21 =     0 r22 =     0 r23 =     0
r24 =     0 r25 =     0 r26 =     0 r27 =     0
r28 =     0 r29 =     0 r30 =     0 r31 =     0

- Memory Data -
 m0 =     0  m1 =    16  m2 =     0  m3 =     0
 m4 =     0  m5 =     0  m6 =     0  m7 =     0
 m8 =     0  m9 =     0 m10 =     0 m11 =     0
m12 =     0 m13 =     0 m14 =     0 m15 =     0
m16 =     0 m17 =     0 m18 =     0 m19 =     0
m20 =     0 m21 =     0 m22 =     0 m23 =     0
m24 =     0 m25 =     0 m26 =     0 m27 =     0
m28 =     0 m29 =     0 m30 =     0 m31 =     0
```

For this testcase with which hazard would generate, I choose to reorder the commands to be like this:

```
I1:     addi    $1, $0, 16
I2:     addi    $3, $0, 8
I3:     addi    $9, $0, 100
I4:     sw      $1, 4($0)
I5:     addi    $2, $1, 4
I6:     lw      $4, 4($0)
I7:     addi    $7, #1, 10
I8:     add     $6, $3, $1
I9:     sub     $5, $4, $3
I10:    and     $8, $7, $3
```

By reordering, I've made the commands that would cause hazard separate, thus I could get the correct result.

So, for the above three testcases, I got the same result as what are given in the testcase_ans file, I believe that my designs are correct.

Problems you met and solutions:

Pipeline designing is quite a new concept for me, and thus is difficult to comprehend. I spent some time understanding what I should do for each stage. However, what the most difficult part for me is still debugging. In this Lab, there are a lot of signals that are almost the same, the only difference is that they are used in different stages. I accidentally wrote a part "instr_id" into "instr", and spent like an hour to find that slight mistake. And for the multiplier to the rightmost of the diagram, I spent a

morning just to find that I should reverse the data0 and data1.
I hope that as I write more and more Verilog codes, I would gradually get used to it.

Summary:

In this Lab, I have a clearer comprehension about how to implement a pipelined CPU, how all the signals should be used, adjusted, or connected, and how to deal with the hazard by reordering. I hope after this Lab, I would do better on my learning.