UM–SJTU Joint Institute

Intro to Computer Organization
(VE370)

Lab Report

Lab 4
Pipelined Processor

Name: Yiwen Yang	ID: 519370910053
Name: Rundong Tang	ID: 519370910050
Name: Shuhui Yang	ID: 519370910064

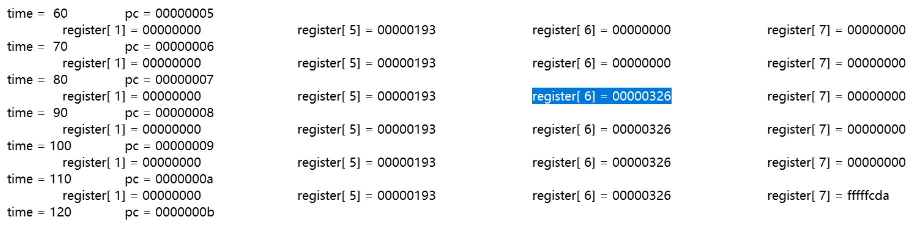
1 Introduction

A pipelined processor is divided into five stages, i.e. IF, ID, EX, MEM, WB, which are connected with one another to form a pipe like structure. It supports a subset of RISC-V instructions including arithmetic-logical instructions: add, addi, sub, and, andi, or, sll, slli, srl, srli, sra; memory-reference instructions: lw, sw, lb, lbu, sb; jumping instructions: beq, bne, bge, blt, jal, jalr. It's composed of PC, Instruction Memory, Register File, 32-bit ALU, 32-bit Adder, Immediate Generator, Control Unit, ALU Control, 2-to-1 MUX, Data Memory. In this lab, we use Verilog HDL to model and implement the pipelined processor.

2 Illustrations for Types of Instruction

2.1 add

Fetch ADD and increment PC; Decode instruction and fetch operands; Add rs1 and rs2; Pass the sum through MEM; Write the sum back to rd. For add instruction, we choose the instruction "add t1 t0 t0" as an example, where t1 stands for x6 and t0 stands for x5. Here x5 = 0x193 (from the previous instructions). We can see that at time = 80, this instruction is in the WB stage and x6 changes from 0 to 0x326, which means the system works correctly for this instruction.

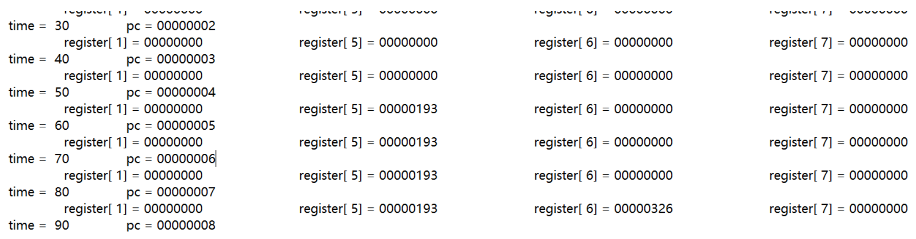


```
time =  60          pc = 00000005
        register[ 1] = 00000000        register[ 5] = 00000193        register[ 6] = 00000000        register[ 7] = 00000000
time =  70          pc = 00000006
        register[ 1] = 00000000        register[ 5] = 00000193        register[ 6] = 00000000        register[ 7] = 00000000
time =  80          pc = 00000007
        register[ 1] = 00000000        register[ 5] = 00000193        register[ 6] = 00000326        register[ 7] = 00000000
time =  90          pc = 00000008
        register[ 1] = 00000000        register[ 5] = 00000193        register[ 6] = 00000326        register[ 7] = 00000000
time = 100          pc = 00000009
        register[ 1] = 00000000        register[ 5] = 00000193        register[ 6] = 00000326        register[ 7] = 00000000
time = 110          pc = 0000000a
        register[ 1] = 00000000        register[ 5] = 00000193        register[ 6] = 00000326        register[ 7] = fffffcda
time = 120          pc = 0000000b
```

Figure 1: Illustration of add

2.2 addi

Fetch ADDI and increment PC; Decode instruction and fetch operands; Add rs1 and immediate; Pass the sum through MEM; Write the sum back to rd. For addi instruction, we choose the instruction "addi t0 x0 0x193" as an example. Here t0 stands for x5. We can find that when time = 50, the instruction is in the WB stage and x5 changes from 0 to 0x193, which means the system works correctly.

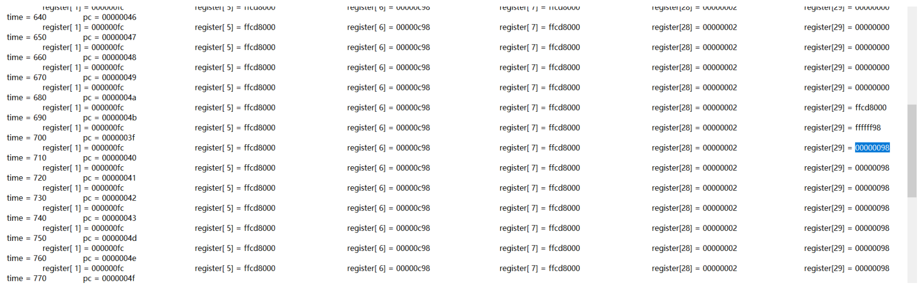


```
time =  30          pc = 00000002
        register[ 1] = 00000000        register[ 5] = 00000000        register[ 6] = 00000000        register[ 7] = 00000000
time =  40          pc = 00000003
        register[ 1] = 00000000        register[ 5] = 00000000        register[ 6] = 00000000        register[ 7] = 00000000
time =  50          pc = 00000004
        register[ 1] = 00000000        register[ 5] = 00000193        register[ 6] = 00000000        register[ 7] = 00000000
time =  60          pc = 00000005
        register[ 1] = 00000000        register[ 5] = 00000193        register[ 6] = 00000000        register[ 7] = 00000000
time =  70          pc = 00000006
        register[ 1] = 00000000        register[ 5] = 00000193        register[ 6] = 00000000        register[ 7] = 00000000
time =  80          pc = 00000007
        register[ 1] = 00000000        register[ 5] = 00000193        register[ 6] = 00000326        register[ 7] = 00000000
time =  90          pc = 00000008
```

Figure 2: Illustration of addi

2.3 lw

Fetch LW and increment PC; Decode instruction and fetch operands; Add offset to rs1 value; Read word from Data Memory; Write the word back to rd. For the lw instruction, we choose the instruction ”lw t4 0(sp)” as an example. Here, t4 stands for x29 and Memory[0+sp] is 0x98(from the previous instructions). We can see that at time t = 700 when the instruction is in the WB stage, the content of x29 changes to 0x98, which means the system works correctly for this instruction.

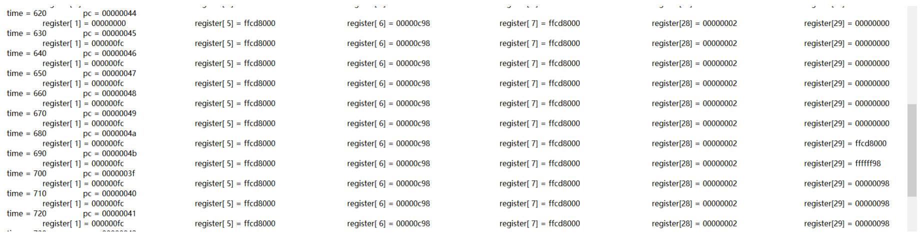


```
          register[ 1] = 000000fc       register[ 5] = ffcd8000       register[ 6] = 00000c98       register[ 7] = ffcd8000       register[28] = 00000002       register[29] = 00000000
time = 640          pc = 00000046
          register[ 1] = 000000fc       register[ 5] = ffcd8000       register[ 6] = 00000c98       register[ 7] = ffcd8000       register[28] = 00000002       register[29] = 00000000
time = 650          pc = 00000047
          register[ 1] = 000000fc       register[ 5] = ffcd8000       register[ 6] = 00000c98       register[ 7] = ffcd8000       register[28] = 00000002       register[29] = 00000000
time = 660          pc = 00000048
          register[ 1] = 000000fc       register[ 5] = ffcd8000       register[ 6] = 00000c98       register[ 7] = ffcd8000       register[28] = 00000002       register[29] = 00000000
time = 670          pc = 00000049
          register[ 1] = 000000fc       register[ 5] = ffcd8000       register[ 6] = 00000c98       register[ 7] = ffcd8000       register[28] = 00000002       register[29] = 00000000
time = 680          pc = 0000004a
          register[ 1] = 000000fc       register[ 5] = ffcd8000       register[ 6] = 00000c98       register[ 7] = ffcd8000       register[28] = 00000002       register[29] = ffcd8000
time = 690          pc = 0000004b
          register[ 1] = 000000fc       register[ 5] = ffcd8000       register[ 6] = 00000c98       register[ 7] = ffcd8000       register[28] = 00000002       register[29] = ffffff98
time = 700          pc = 0000003f
          register[ 1] = 000000fc       register[ 5] = ffcd8000       register[ 6] = 00000c98       register[ 7] = ffcd8000       register[28] = 00000002       register[29] = 00000098
time = 710          pc = 00000040
          register[ 1] = 000000fc       register[ 5] = ffcd8000       register[ 6] = 00000c98       register[ 7] = ffcd8000       register[28] = 00000002       register[29] = 00000098
time = 720          pc = 00000041
          register[ 1] = 000000fc       register[ 5] = ffcd8000       register[ 6] = 00000c98       register[ 7] = ffcd8000       register[28] = 00000002       register[29] = 00000098
time = 730          pc = 00000042
          register[ 1] = 000000fc       register[ 5] = ffcd8000       register[ 6] = 00000c98       register[ 7] = ffcd8000       register[28] = 00000002       register[29] = 00000098
time = 740          pc = 00000043
          register[ 1] = 000000fc       register[ 5] = ffcd8000       register[ 6] = 00000c98       register[ 7] = ffcd8000       register[28] = 00000002       register[29] = 00000098
time = 750          pc = 0000004d
          register[ 1] = 000000fc       register[ 5] = ffcd8000       register[ 6] = 00000c98       register[ 7] = ffcd8000       register[28] = 00000002       register[29] = 00000098
time = 760          pc = 0000004e
          register[ 1] = 000000fc       register[ 5] = ffcd8000       register[ 6] = 00000c98       register[ 7] = ffcd8000       register[28] = 00000002       register[29] = 00000098
time = 770          pc = 0000004f
```

Figure 3: Illustration of lw

2.4 sw

Fetch SW and increment PC; Decode instruction and fetch operands; Add offset to rs1 value; Read rs2. For the sw instruction, we choose the instruction ”sw t0 0(sp)” as an example. Here t0 stands for x5, which is 0x98 from the previous instructions. We can see that at time = 660, 4(sp) changes to 0x98, which means the system works correctly for this instruction.

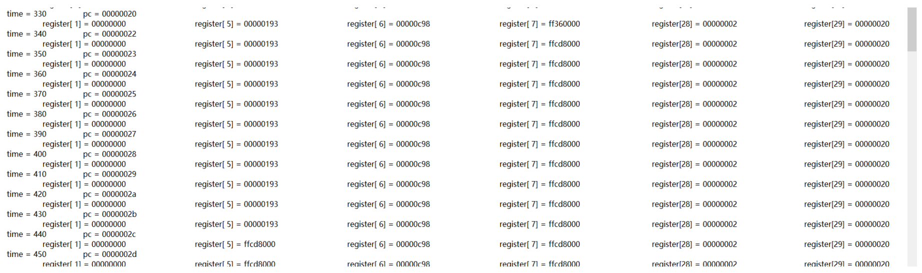


```
time = 620          pc = 00000044
          register[ 1] = 00000000       register[ 5] = ffcd8000       register[ 6] = 00000c98       register[ 7] = ffcd8000       register[28] = 00000002       register[29] = 00000000
time = 630          pc = 00000045
          register[ 1] = 000000fc       register[ 5] = ffcd8000       register[ 6] = 00000c98       register[ 7] = ffcd8000       register[28] = 00000002       register[29] = 00000000
time = 640          pc = 00000046
          register[ 1] = 000000fc       register[ 5] = ffcd8000       register[ 6] = 00000c98       register[ 7] = ffcd8000       register[28] = 00000002       register[29] = 00000000
time = 650          pc = 00000047
          register[ 1] = 000000fc       register[ 5] = ffcd8000       register[ 6] = 00000c98       register[ 7] = ffcd8000       register[28] = 00000002       register[29] = 00000000
time = 660          pc = 00000048
          register[ 1] = 000000fc       register[ 5] = ffcd8000       register[ 6] = 00000c98       register[ 7] = ffcd8000       register[28] = 00000002       register[29] = 00000000
time = 670          pc = 00000049
          register[ 1] = 000000fc       register[ 5] = ffcd8000       register[ 6] = 00000c98       register[ 7] = ffcd8000       register[28] = 00000002       register[29] = 00000000
time = 680          pc = 0000004a
          register[ 1] = 000000fc       register[ 5] = ffcd8000       register[ 6] = 00000c98       register[ 7] = ffcd8000       register[28] = 00000002       register[29] = ffcd8000
time = 690          pc = 0000004b
          register[ 1] = 000000fc       register[ 5] = ffcd8000       register[ 6] = 00000c98       register[ 7] = ffcd8000       register[28] = 00000002       register[29] = ffffff98
time = 700          pc = 0000003f
          register[ 1] = 000000fc       register[ 5] = ffcd8000       register[ 6] = 00000c98       register[ 7] = ffcd8000       register[28] = 00000002       register[29] = 00000098
time = 710          pc = 00000040
          register[ 1] = 000000fc       register[ 5] = ffcd8000       register[ 6] = 00000c98       register[ 7] = ffcd8000       register[28] = 00000002       register[29] = 00000098
time = 720          pc = 00000041
          register[ 1] = 000000fc       register[ 5] = ffcd8000       register[ 6] = 00000c98       register[ 7] = ffcd8000       register[28] = 00000002       register[29] = 00000098
```

Figure 4: Illustration of sw

2.5 beq

Fetch BEQ and increment PC; Decode instruction and fetch operands, pass on PC+4; Do rs1-rs2 and check if result = 0. Calculate branch target address; Update PC. For the beq instruction we choose the instruction ”beq t2 x0 error1” as an example. Here t2 stands for x7. We can see that at time = 380 pc is not the previous pc plus 4, but the destination of error1, which means the system works correctly for this instruction.

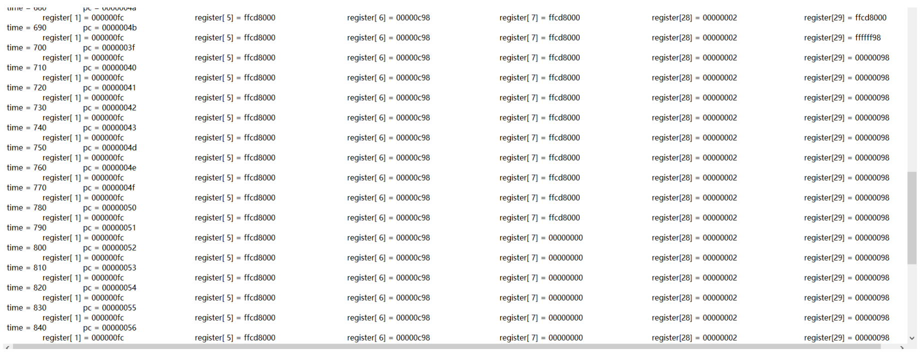


```
time = 330          pc = 00000020
         register[ 1] = 00000000     register[ 5] = 00000193     register[ 6] = 00000c98     register[ 7] = ff360000     register[28] = 00000002     register[29] = 00000020
time = 340          pc = 00000022
         register[ 1] = 00000000     register[ 5] = 00000193     register[ 6] = 00000c98     register[ 7] = ffcd8000     register[28] = 00000002     register[29] = 00000020
time = 350          pc = 00000023
         register[ 1] = 00000000     register[ 5] = 00000193     register[ 6] = 00000c98     register[ 7] = ffcd8000     register[28] = 00000002     register[29] = 00000020
time = 360          pc = 00000024
         register[ 1] = 00000000     register[ 5] = 00000193     register[ 6] = 00000c98     register[ 7] = ffcd8000     register[28] = 00000002     register[29] = 00000020
time = 370          pc = 00000025
         register[ 1] = 00000000     register[ 5] = 00000193     register[ 6] = 00000c98     register[ 7] = ffcd8000     register[28] = 00000002     register[29] = 00000020
time = 380          pc = 00000026
         register[ 1] = 00000000     register[ 5] = 00000193     register[ 6] = 00000c98     register[ 7] = ffcd8000     register[28] = 00000002     register[29] = 00000020
time = 390          pc = 00000027
         register[ 1] = 00000000     register[ 5] = 00000193     register[ 6] = 00000c98     register[ 7] = ffcd8000     register[28] = 00000002     register[29] = 00000020
time = 400          pc = 00000028
         register[ 1] = 00000000     register[ 5] = 00000193     register[ 6] = 00000c98     register[ 7] = ffcd8000     register[28] = 00000002     register[29] = 00000020
time = 410          pc = 00000029
         register[ 1] = 00000000     register[ 5] = 00000193     register[ 6] = 00000c98     register[ 7] = ffcd8000     register[28] = 00000002     register[29] = 00000020
time = 420          pc = 0000002a
         register[ 1] = 00000000     register[ 5] = 00000193     register[ 6] = 00000c98     register[ 7] = ffcd8000     register[28] = 00000002     register[29] = 00000020
time = 430          pc = 0000002b
         register[ 1] = 00000000     register[ 5] = 00000193     register[ 6] = 00000c98     register[ 7] = ffcd8000     register[28] = 00000002     register[29] = 00000020
time = 440          pc = 0000002c
         register[ 1] = 00000000     register[ 5] = ffcd8000     register[ 6] = 00000c98     register[ 7] = ffcd8000     register[28] = 00000002     register[29] = 00000020
time = 450          pc = 0000002d
         register[ 1] = 00000000     register[ 5] = ffcd8000     register[ 6] = 00000c98     register[ 7] = ffcd8000     register[28] = 00000002     register[29] = 00000020
```

Figure 5: Illustration of beq

2.6 jal

For the jal instruction, we choose the instruction ”jal x1 memory_test” as an example and we can see that at time = 730, the pc is not the previous pc plus 4 but the destination of memory_test, which means the system works correctly for this instruction.



```
time = 680          pc = 0000004a
         register[ 1] = 000000fc     register[ 5] = ffcd8000     register[ 6] = 00000c98     register[ 7] = ffcd8000     register[28] = 00000002     register[29] = ffcd8000
time = 690          pc = 0000004b
         register[ 1] = 000000fc     register[ 5] = ffcd8000     register[ 6] = 00000c98     register[ 7] = ffcd8000     register[28] = 00000002     register[29] = ffffff98
time = 700          pc = 0000003f
         register[ 1] = 000000fc     register[ 5] = ffcd8000     register[ 6] = 00000c98     register[ 7] = ffcd8000     register[28] = 00000002     register[29] = 00000098
time = 710          pc = 00000040
         register[ 1] = 000000fc     register[ 5] = ffcd8000     register[ 6] = 00000c98     register[ 7] = ffcd8000     register[28] = 00000002     register[29] = 00000098
time = 720          pc = 00000041
         register[ 1] = 000000fc     register[ 5] = ffcd8000     register[ 6] = 00000c98     register[ 7] = ffcd8000     register[28] = 00000002     register[29] = 00000098
time = 730          pc = 00000042
         register[ 1] = 000000fc     register[ 5] = ffcd8000     register[ 6] = 00000c98     register[ 7] = ffcd8000     register[28] = 00000002     register[29] = 00000098
time = 740          pc = 00000043
         register[ 1] = 000000fc     register[ 5] = ffcd8000     register[ 6] = 00000c98     register[ 7] = ffcd8000     register[28] = 00000002     register[29] = 00000098
time = 750          pc = 0000004d
         register[ 1] = 000000fc     register[ 5] = ffcd8000     register[ 6] = 00000c98     register[ 7] = ffcd8000     register[28] = 00000002     register[29] = 00000098
time = 760          pc = 0000004e
         register[ 1] = 000000fc     register[ 5] = ffcd8000     register[ 6] = 00000c98     register[ 7] = ffcd8000     register[28] = 00000002     register[29] = 00000098
time = 770          pc = 0000004f
         register[ 1] = 000000fc     register[ 5] = ffcd8000     register[ 6] = 00000c98     register[ 7] = ffcd8000     register[28] = 00000002     register[29] = 00000098
time = 780          pc = 00000050
         register[ 1] = 000000fc     register[ 5] = ffcd8000     register[ 6] = 00000c98     register[ 7] = ffcd8000     register[28] = 00000002     register[29] = 00000098
time = 790          pc = 00000051
         register[ 1] = 000000fc     register[ 5] = ffcd8000     register[ 6] = 00000c98     register[ 7] = 00000000     register[28] = 00000002     register[29] = 00000098
time = 800          pc = 00000052
         register[ 1] = 000000fc     register[ 5] = ffcd8000     register[ 6] = 00000c98     register[ 7] = 00000000     register[28] = 00000002     register[29] = 00000098
time = 810          pc = 00000053
         register[ 1] = 000000fc     register[ 5] = ffcd8000     register[ 6] = 00000c98     register[ 7] = 00000000     register[28] = 00000002     register[29] = 00000098
time = 820          pc = 00000054
         register[ 1] = 000000fc     register[ 5] = ffcd8000     register[ 6] = 00000c98     register[ 7] = 00000000     register[28] = 00000002     register[29] = 00000098
time = 830          pc = 00000055
         register[ 1] = 000000fc     register[ 5] = ffcd8000     register[ 6] = 00000c98     register[ 7] = 00000000     register[28] = 00000002     register[29] = 00000098
time = 840          pc = 00000056
         register[ 1] = 000000fc     register[ 5] = ffcd8000     register[ 6] = 00000c98     register[ 7] = 00000000     register[28] = 00000002     register[29] = 00000098
```

Figure 6: Illustration of jal

3 RTL Schematic

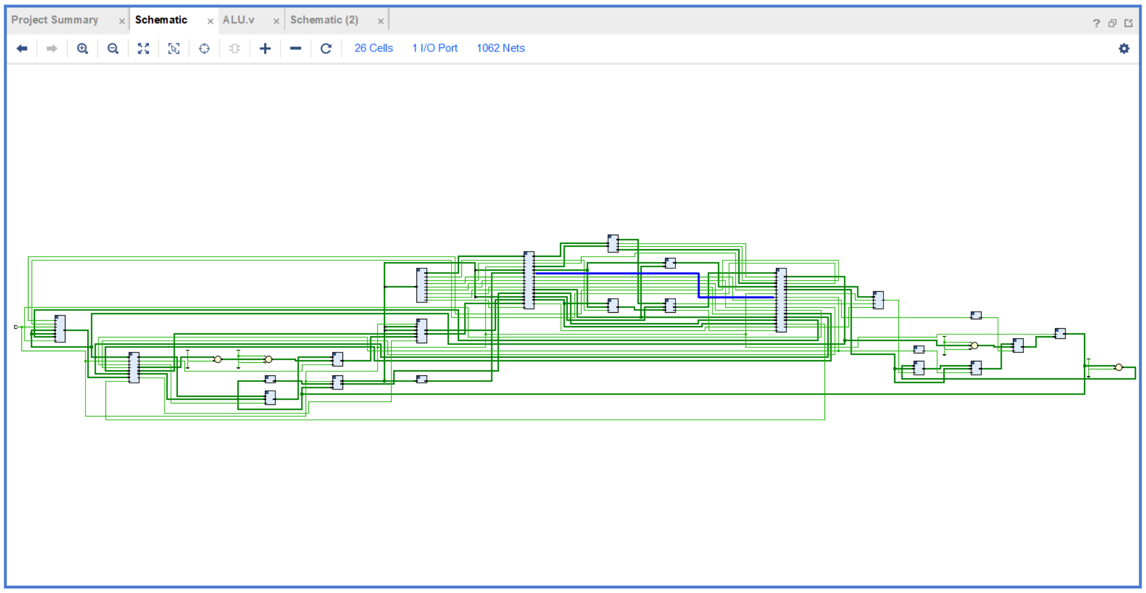


Figure 7: RTL Schematic