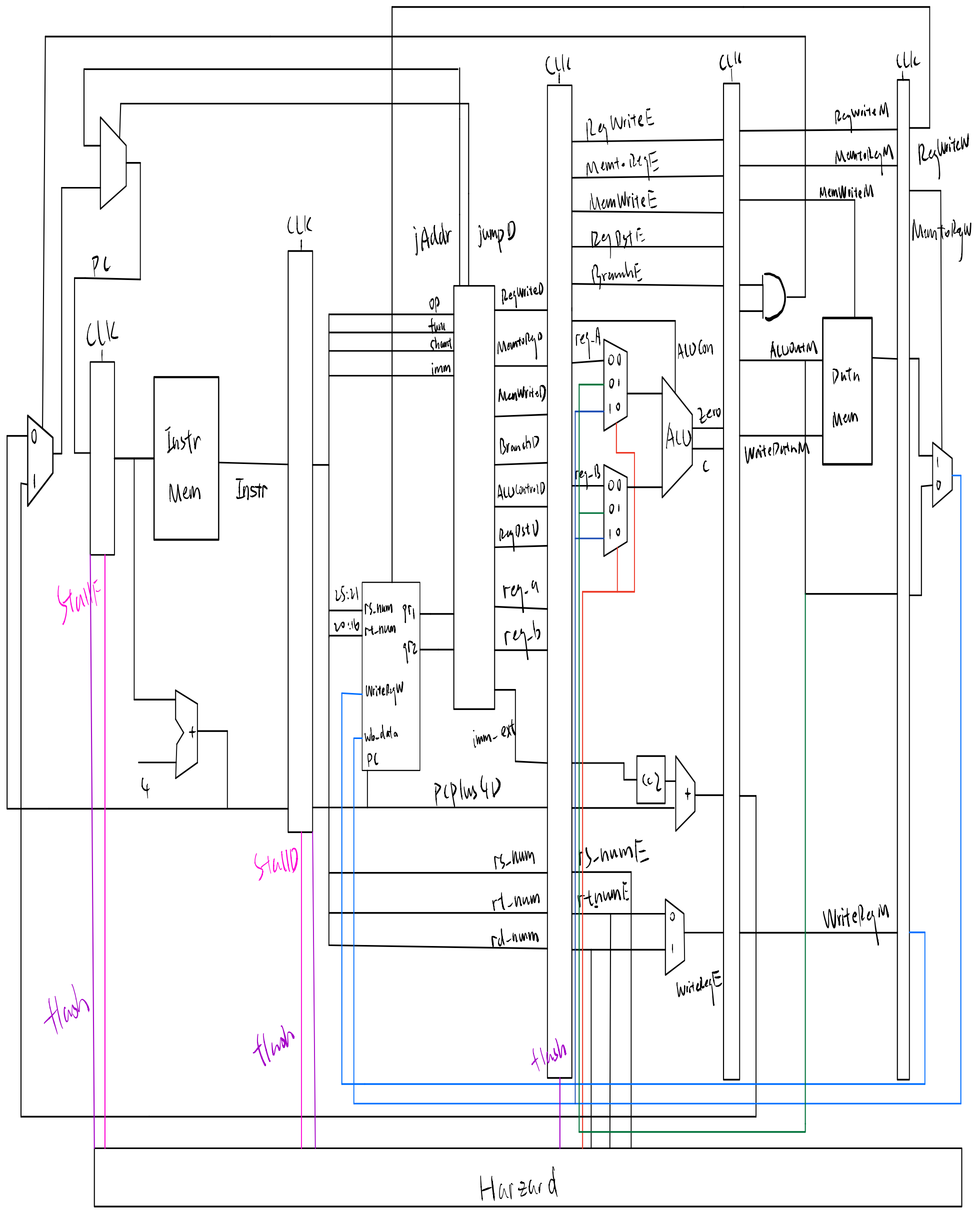
Project 4 Report

I. Project Introduction

In project 4, a pipelined MIPS central processing unit should be constructed with HDL Verilog. Based on what has been done in project 3, the ALU module, the main work of this project is to add sequential circuits to achieve pipelined feature, maintain a program counter (pc), build instruction and data memory, construct a complete datapath and control unit, handle possible hazards for pipelined feature and finally assemble all parts into a functioning CPU.

II. Block Diagram



III. Main Work

A. Files Module

1. newDefine.h

This file defines all the frequently used parameters. Therefore, other modules could directly refer to the corresponding parameter names. If in later test case section, some signal are unfamiliar, it is great to check this file out.

2. my_CPU.v

This is the main file of this project. It successively constructs a pipelined CPU by making use of edge triggered <code>always</code> Verilog syntax and assembles the following parts to a functioning CPU simulator.

3. instruction_mem.v & data_mem.v

Instruction memory and data memory are separated module. They could read the instructions and data from given path to a txt file, with specified pc or data address.

4. reg_file.v

Register file is also a separated module, which contains all 32 general registers and supports read and write instructions. During <code>jr</code> instruction, it also supports recording the return address.

5. control.v

This control takes in operation code, function code, shamt, immediate number, read values from register file. Then it generates all the control signals and values ready to be put into ALU.

6. ALU.v

Perform calculation on processed values given by control unit according to the ALU Control.

7. harzard.v

   This file implements the hazard handling. It supports forwarding, stalling and flashing.

8. Test_batch.v

   This is a test batch adapted from provided sample.

B. Test Cases

   1. instructionMem.txt

   2. dataMem.txt

IV. Test Case Display

For the ease of understanding and displaying, in every case the pipelined CPU module will take a few instructions into execution. The clock will continue until the last instruction of the group finishes execution. According to each output cases, explanations are given right after the picture. Detailed data flow is not given for every example, since the pipelined logic is basically the same. Also the pipelined feature is easy to discover in the output of each test cases. Besides, the overall continuous functioning test case is attached. It is shown in another file named continuous_test_output.

The following is few assumptions or presettings that are essential:

Register File:

gr[`gr0] = 32'h0000_0000,

gr[`gr8] = 32'h0000_0008,

gr[`gr9] = 32'h0000_0009,

gr[`gr10] = 32'h0000_000a,

gr[`gr11] = 32'h0000_000b,

gr[`gr12] = 32'h0000_000c,

gr[`gr13] = 32'h0000_000d,

gr[`gr14] = 32'h0000_000e,

gr[`gr15] = 32'h0000_000f,

gr[`gr16] = 32'h0000_0010,

gr[`gr17] = 32'h0000_0011,

gr[`gr18] = 32'h0000_0012,

gr[`gr19] = 32'h0000_0013,

gr[`gr20] = 32'h0000_0014;

gr[`gr21] = 32'h0000_0015,

gr[`gr22] = 32'h0000_0016,

gr[`gr23] = 32'h0000_0017,

gr[`gr24] = 32'h0000_0018,

gr[`gr25] = 32'h0000_0019;

Data Memory:

000000_00000_01000_01001_00000_100000,

000000_00000_00000_00000_00000_100000,

000000_00000_00000_00000_00000_100000,

000000_00000_01000_01001_00000_100001,

000000_01010_01011_01100_00000_100000,

000000_00000_00000_00000_00000_000001,

000000_00000_00000_00000_00000_000010,

000000_00000_00000_00000_00000_100000,

000000_00000_00000_00000_10000_000000;

Above are default value of corresponding general registers inside the register file module and data memory. Since the module is not actually running a well written MIPS code file, it will be quite complex if different general registers are used every time. Sometimes they may not even have a value stored inside. Therefore, in test cases, which is designed to illustrate the performance, operations are mostly done on these registers, using their predefined values. However, if instructions need to use other general registers, the module could also handle it.

* Note that all the outputs below are display in either binary or hexadecimal.

* Note that for add & addu, sub & subu, addi & addiu examples, we have

gr[`gr8] = 32’h0000_0001,

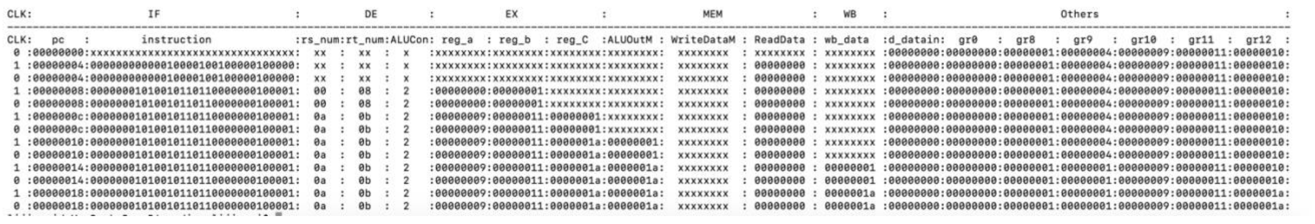
gr[`gr9] = 32’h0000_0004,

gr[`gr10] = 32’h0000_0009,

gr[`gr11] = 32’h0000_0011,

gr[`gr12] = 32’h0000_0010; since the code was not fully updated.

A. add & addu



```
CLK:                 IF                      :        DE        :          EX            :          MEM          :   WB    :                          Others                          :
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
CLK:   pc   :         instruction            :rs_num:rt_num:ALUCon: reg_a  : reg_b  : reg_C  :ALUOutM : WriteDataM : ReadData : wb_data  :d_datain:  gr0   :  gr8   :  gr9   :  gr10  :  gr11  :  gr12  :
 0 :00000000:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :00000000:00000000:00000001:00000004:00000009:00000011:00000010:
 1 :00000004:00000000000010000100100000100000:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : 00000000 : xxxxxxxx :00000000:00000000:00000001:00000004:00000009:00000011:00000010:
 0 :00000004:00000000000010000100100000100000:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : 00000000 : xxxxxxxx :00000000:00000000:00000001:00000004:00000009:00000011:00000010:
 1 :00000008:00000001010010110110000000100001:  00  :  08  :  2   :00000000:00000001:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : 00000000 : xxxxxxxx :00000000:00000000:00000001:00000004:00000009:00000011:00000010:
 0 :00000008:00000001010010110110000000100001:  00  :  08  :  2   :00000000:00000001:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : 00000000 : xxxxxxxx :00000000:00000000:00000001:00000004:00000009:00000011:00000010:
 1 :0000000c:00000001010010110110000000100001:  0a  :  0b  :  2   :00000009:00000011:00000001:xxxxxxxx:  xxxxxxxx  : 00000000 : xxxxxxxx :00000000:00000000:00000001:00000004:00000009:00000011:00000010:
 0 :0000000c:00000001010010110110000000100001:  0a  :  0b  :  2   :00000009:00000011:00000001:xxxxxxxx:  xxxxxxxx  : 00000000 : xxxxxxxx :00000000:00000000:00000001:00000004:00000009:00000011:00000010:
 1 :00000010:00000001010010110110000000100001:  0a  :  0b  :  2   :00000009:00000011:0000001a:00000001:  xxxxxxxx  : 00000000 : xxxxxxxx :00000000:00000000:00000001:00000004:00000009:00000011:00000010:
 0 :00000010:00000001010010110110000000100001:  0a  :  0b  :  2   :00000009:00000011:0000001a:00000001:  xxxxxxxx  : 00000000 : xxxxxxxx :00000000:00000000:00000001:00000004:00000009:00000011:00000010:
 1 :00000014:00000001010010110110000000100001:  0a  :  0b  :  2   :00000009:00000011:0000001a:0000001a:  xxxxxxxx  : 00000000 : 00000001 :00000000:00000000:00000001:00000001:00000009:00000011:00000010:
 0 :00000014:00000001010010110110000000100001:  0a  :  0b  :  2   :00000009:00000011:0000001a:0000001a:  xxxxxxxx  : 00000000 : 00000001 :00000000:00000000:00000001:00000001:00000009:00000011:00000010:
 1 :00000018:00000001010010110110000000100001:  0a  :  0b  :  2   :00000009:00000011:0000001a:0000001a:  xxxxxxxx  : 00000000 : 0000001a :00000000:00000000:00000001:00000001:00000009:00000011:0000001a:
 0 :00000018:00000001010010110110000000100001:  0a  :  0b  :  2   :00000009:00000011:0000001a:0000001a:  xxxxxxxx  : 00000000 : 0000001a :00000000:00000000:00000001:00000001:00000009:00000011:0000001a:
```

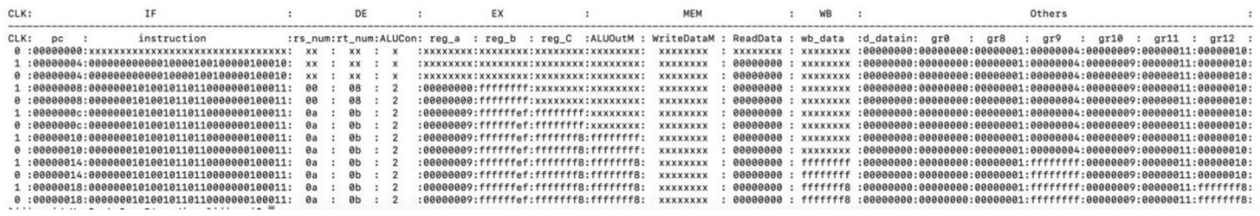
Inputted instructions:

add: 000000_00000_01000_01001_00000_100000

addu: 000000_01010_01011_01100 _00000_100000

Starting from pc = 32’h0000_0000, in the first clock edge, pc is updated by 4 (so that now it is pointing to the next instruction) and the instruction located at address 0x00000000 is fetched into the datapath. With the second clock edge coming, rs_num and rt_num, which stand for the number of general registers to be taken value from, come into being, together with ALUCon. reg_A and reg_B are now storing the values taken from the register file according to rs_num (`gr0) and rt_num(`gr8). Also notice that at this moment, next instruction, which is an addu instruction comes in sequence. In the third clock rising edge, the main work is ALU operation. As it is shown in the picture, now reg_C equals the sum of previous reg_a and reg_b. In the meantime, rs_num and rd_num are changed to the number of general registers represented by the addu instruction. The reason why ALUCon at the moment stays the same is because with the decoding of control unit, add and addu will make use of the same add function of the ALU unit. The difference in processing is made when fetching the values of reg_a and reg_b. Then in the fourth clock rise, reg_C's value is passed to ALUOutM, which is going to go through a multiplexer together with ReadData to decide which one of them will be the value written back to the register file. Now stored in the reg_C are the addu ALU operation result. In the fifth clock rising edge, where add instruction comes to WB stage, the wb_data is now the value of ALUOutM since it is an add instruction. The destination register, gr[`gr9] changed from 32'h0000_0004 to 32'h0000_0001 which is the addition result. Then add instruction finishes. In the sixth clock, result of addu instruction is also taken from ALUOutM and written into its destination register, gr[`gr12], whose value changed from 32'h0000_0010 to 32'h0000_000a. Both two instructions finishe then.

B. sub & subu



```
CLK:                IF                :        DE        :          EX            :              MEM              :    WB    :                            Others                                   :
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
CLK:   pc    :        instruction          :rs_num:rt_num:ALUCon: reg_a   : reg_b   : reg_C   :ALUOutM : WriteDataM : ReadData : wb_data  :d_datain:  gr0    :  gr8    :  gr9    :  gr10   :  gr11   :  gr12   :
 0 :00000000:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :00000000:00000000:00000001:00000004:00000009:00000011:00000010:
 1 :00000004:00000000000010000100100000100010:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : 00000000 : xxxxxxxx :00000000:00000000:00000001:00000004:00000009:00000011:00000010:
 0 :00000004:00000000000010000100100000100010:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : 00000000 : xxxxxxxx :00000000:00000000:00000001:00000004:00000009:00000011:00000010:
 1 :00000008:00000001010010110110000000100011:  00  :  08  :  2   :00000000:ffffffff:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : 00000000 : xxxxxxxx :00000000:00000000:00000001:00000004:00000009:00000011:00000010:
 0 :00000008:00000001010010110110000000100011:  00  :  08  :  2   :00000000:ffffffff:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : 00000000 : xxxxxxxx :00000000:00000000:00000001:00000004:00000009:00000011:00000010:
 1 :0000000c:00000001010010110110000000100011:  0a  :  0b  :  2   :00000009:ffffffef:ffffffff:xxxxxxxx:  xxxxxxxx  : 00000000 : xxxxxxxx :00000000:00000000:00000001:00000004:00000009:00000011:00000010:
 0 :0000000c:00000001010010110110000000100011:  0a  :  0b  :  2   :00000009:ffffffef:ffffffff:xxxxxxxx:  xxxxxxxx  : 00000000 : xxxxxxxx :00000000:00000000:00000001:00000004:00000009:00000011:00000010:
 1 :00000010:00000001010010110110000000100011:  0a  :  0b  :  2   :00000009:ffffffef:fffffff8:ffffffff:  xxxxxxxx  : 00000000 : xxxxxxxx :00000000:00000000:00000001:00000004:00000009:00000011:00000010:
 0 :00000010:00000001010010110110000000100011:  0a  :  0b  :  2   :00000009:ffffffef:fffffff8:ffffffff:  xxxxxxxx  : 00000000 : xxxxxxxx :00000000:00000000:00000001:00000004:00000009:00000011:00000010:
 1 :00000014:00000001010010110110000000100011:  0a  :  0b  :  2   :00000009:ffffffef:fffffff8:fffffff8:  xxxxxxxx  : 00000000 : ffffffff :00000000:00000000:00000001:ffffffff:00000009:00000011:00000010:
 0 :00000014:00000001010010110110000000100011:  0a  :  0b  :  2   :00000009:ffffffef:fffffff8:fffffff8:  xxxxxxxx  : 00000000 : ffffffff :00000000:00000000:00000001:ffffffff:00000009:00000011:00000010:
 1 :00000018:00000001010010110110000000100011:  0a  :  0b  :  2   :00000009:ffffffef:fffffff8:fffffff8:  xxxxxxxx  : 00000000 : fffffff8 :00000000:00000000:00000001:ffffffff:00000009:00000011:fffffff8:
 0 :00000018:00000001010010110110000000100011:  0a  :  0b  :  2   :00000009:ffffffef:fffffff8:fffffff8:  xxxxxxxx  : 00000000 : fffffff8 :00000000:00000000:00000001:ffffffff:00000009:00000011:fffffff8:
```

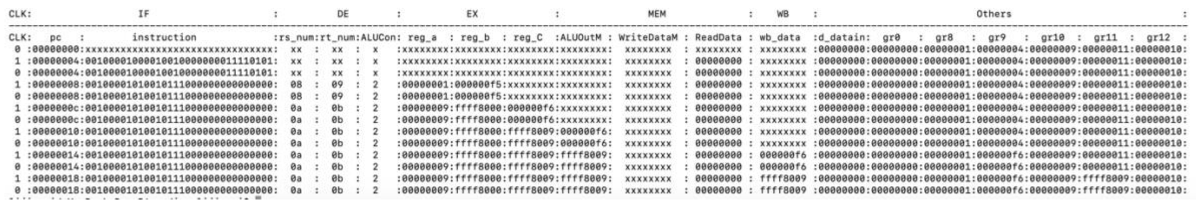
Inputted instructions:

sub: 000000_00000 _01000_01001_00000_100010

subu: 000000_01010_01011_01100_00000_100011

The pipelined logic flow of sub and subu are basically the same with add and addu. There are few more things to emphasis, however. The sub instruction will be taken as the example for illustration below. First, ALUCon is still the same as add and addu is because the minus sign is expressed in terms of converting value fetched from general register rt_num into its 2’s complement form (32’hffff_ffff, originally is 32’h0000_0001). Therefore, the add function of ALU could be reused. Second, by adding 32’h0000_0000 and 32’hffff_ffff, 32’hffff_ffff is obtained which stands for -1 (32’0000_0000 – 32’h0000_0001).

C. addi & addiu



```
CLK:                IF                     :           DE         :           EX            :                MEM              :    WB    :                         Others                            :
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
CLK:    pc    :          instruction         :rs_num:rt_num:ALUCon: reg_a   : reg_b   : reg_C   :ALUOutM : WriteDataM : ReadData : wb_data  :d_datain:  gr0    :  gr8    :  gr9    :  gr10   :  gr11   :  gr12   :
 0 :00000000:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :00000000:00000000:00000001:00000004:00000009:00000011:00000010:
 1 :00000004:00100001000010010000000011110101:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : 00000000 : xxxxxxxx :00000000:00000000:00000001:00000004:00000009:00000011:00000010:
 0 :00000004:00100001000010010000000011110101:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : 00000000 : xxxxxxxx :00000000:00000000:00000001:00000004:00000009:00000011:00000010:
 1 :00000008:00100001010010111000000000000000:  08  :  09  :  2   :00000001:000000f5:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : 00000000 : xxxxxxxx :00000000:00000000:00000001:00000004:00000009:00000011:00000010:
 0 :00000008:00100001010010111000000000000000:  08  :  09  :  2   :00000001:000000f5:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : 00000000 : xxxxxxxx :00000000:00000000:00000001:00000004:00000009:00000011:00000010:
 1 :0000000c:00100001010010111000000000000000:  0a  :  0b  :  2   :00000009:ffff8000:000000f6:xxxxxxxx:  xxxxxxxx  : 00000000 : xxxxxxxx :00000000:00000000:00000001:00000004:00000009:00000011:00000010:
 0 :0000000c:00100001010010111000000000000000:  0a  :  0b  :  2   :00000009:ffff8000:000000f6:xxxxxxxx:  xxxxxxxx  : 00000000 : xxxxxxxx :00000000:00000000:00000001:00000004:00000009:00000011:00000010:
 1 :00000010:00100001010010111000000000000000:  0a  :  0b  :  2   :00000009:ffff8000:ffff8009:000000f6:  xxxxxxxx  : 00000000 : xxxxxxxx :00000000:00000000:00000001:00000004:00000009:00000011:00000010:
 0 :00000010:00100001010010111000000000000000:  0a  :  0b  :  2   :00000009:ffff8000:ffff8009:000000f6:  xxxxxxxx  : 00000000 : xxxxxxxx :00000000:00000000:00000001:00000004:00000009:00000011:00000010:
 1 :00000014:00100001010010111000000000000000:  0a  :  0b  :  2   :00000009:ffff8000:ffff8009:ffff8009:  xxxxxxxx  : 00000000 : 000000f6 :00000000:00000000:00000001:000000f6:00000009:00000011:00000010:
 0 :00000014:00100001010010111000000000000000:  0a  :  0b  :  2   :00000009:ffff8000:ffff8009:ffff8009:  xxxxxxxx  : 00000000 : 000000f6 :00000000:00000000:00000001:000000f6:00000009:00000011:00000010:
 1 :00000018:00100001010010111000000000000000:  0a  :  0b  :  2   :00000009:ffff8000:ffff8009:ffff8009:  xxxxxxxx  : 00000000 : ffff8009 :00000000:00000000:00000001:000000f6:00000009:ffff8009:00000010:
 0 :00000018:00100001010010111000000000000000:  0a  :  0b  :  2   :00000009:ffff8000:ffff8009:ffff8009:  xxxxxxxx  : 00000000 : ffff8009 :00000000:00000000:00000001:000000f6:00000009:ffff8009:00000010:
```

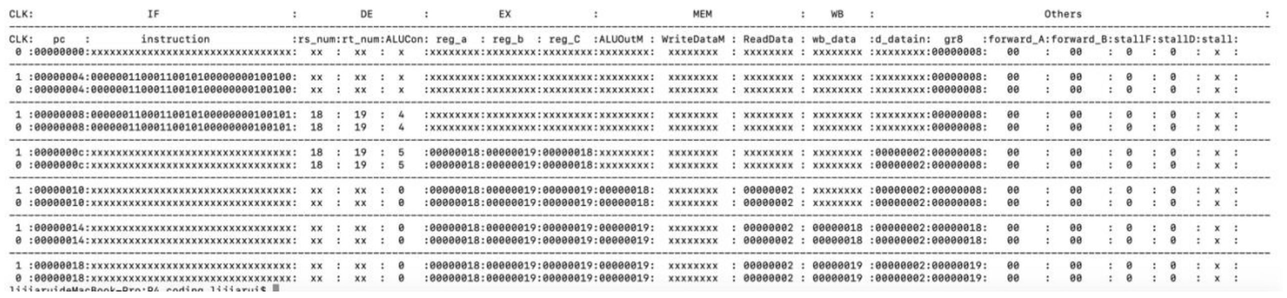
Inputted instructions:

addi: 001000_00000_01000_0000_0000_1111_0101

addiu: 001001_01010_01011_1000_0000_0000_0000

The same as add and addu instructions, it takes 6 clock cycle to finish the execution of both two instructions. It is easy to observe the pipelined stages folloing the logic described in add & addu section. There are two main difference between them. First, for addi and addiu, the addition is between the last 16 bits immediate number and value stored in rs register. Therefore, for addi instruction, we have reg_a = 32’h0000_0001, reg_b = 32’h0000_00f5. Second, the destination register is rt now. As it is shown in the picture, in the last two clock cycle, it is `gr9 and `gr11 is the destination registers but not `gr12 this time.

D. and & or



```
CLK:               IF                       :        DE         :          EX              :            MEM            :    WB    :                       Others                              :
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
CLK:   pc   :         instruction           :rs_num:rt_num:ALUCon: reg_a  : reg_b  : reg_C  :ALUOutM : WriteDataM : ReadData : wb_data  :d_datain:  gr8   :forward_A:forward_B:stallF:stallD:stall:
 0 :00000000:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:   00    :   00    :  0   :  0   :  x  :
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000004:00000011000110010100000000100100:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:   00    :   00    :  0   :  0   :  x  :
 0 :00000004:00000011000110010100000000100100:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:   00    :   00    :  0   :  0   :  x  :
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000008:00000011000110010100000000100101:  18  :  19  :  4   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:   00    :   00    :  0   :  0   :  x  :
 0 :00000008:00000011000110010100000000100101:  18  :  19  :  4   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:   00    :   00    :  0   :  0   :  x  :
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :0000000c:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  18  :  19  :  5   :00000018:00000019:00000018:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :00000002:00000008:   00    :   00    :  0   :  0   :  x  :
 0 :0000000c:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  18  :  19  :  5   :00000018:00000019:00000018:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :00000002:00000008:   00    :   00    :  0   :  0   :  x  :
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000010:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000019:00000019:00000018:  xxxxxxxx  : 00000002 : xxxxxxxx :00000002:00000008:   00    :   00    :  0   :  0   :  x  :
 0 :00000010:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000019:00000019:00000018:  xxxxxxxx  : 00000002 : xxxxxxxx :00000002:00000008:   00    :   00    :  0   :  0   :  x  :
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000014:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000019:00000019:00000019:  xxxxxxxx  : 00000002 : 00000018 :00000002:00000018:   00    :   00    :  0   :  0   :  x  :
 0 :00000014:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000019:00000019:00000019:  xxxxxxxx  : 00000002 : 00000018 :00000002:00000018:   00    :   00    :  0   :  0   :  x  :
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000018:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000019:00000019:00000019:  xxxxxxxx  : 00000002 : 00000019 :00000002:00000019:   00    :   00    :  0   :  0   :  x  :
 0 :00000018:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000019:00000019:00000019:  xxxxxxxx  : 00000002 : 00000019 :00000002:00000019:   00    :   00    :  0   :  0   :  x  :
lijiaruideMacBook-Pro:P4_coding lijiarui$
```

Inputted instructions:

and: 000000_11000_11001_01000_00000_100100

or: 000000_11000_11001 _01000_00000_100101

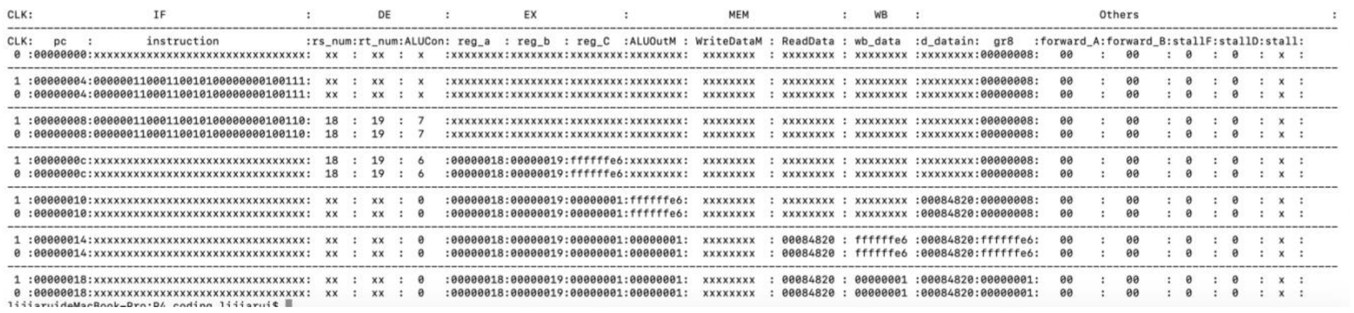
Note that there are some changes in the test case outcome starting from this example. This is because many updates to the coding are added to the source file. First, instructions are no longer remain the same when no instruction is there since, now the program is reading instructions from a separate text file according to given program counter. Therefore, when the address is out of range, the returned value is unknown. Second, now the value of reg_a and reg_b are displayed in the third cycle, which means they now stands for the actual value put into the ALU module in this cycle, but not the register value fetched from the register file in DE stage.

For and & or instructions, they are both reading the value of :

`gr24 (32'b0000_0000_0000_0000_0000_0000_0001_1000) ,

`gr25 (32'b0000_0000_0000_0000_0000_0000_0001_1001) to do operation. And finally they will write the value to `gr8. Therefore as it is shown in the picture, in the third and fourth clock cycle, the result of ALU equals to value of `gr24 (and) and `gr25 (or) respectively. And in the fifth and sixth clock rise, obviously to discover in the picture, they are written into `gr8 sequentially.

E. nor & xor



```
CLK:                 IF                   :         DE        :          EX           :                MEM            :    WB    :                           Others                              :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
CLK:   pc    :        instruction          :rs_num:rt_num:ALUCon: reg_a   : reg_b   : reg_C   :ALUOutM : WriteDataM : ReadData : wb_data  :d_datain:  gr8    :forward_A:forward_B:stallF:stallD:stall:
 0 :00000000:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:   00    :   00    :  0   :  0   :  x  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000004:00000011000110010100000000100111:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:   00    :   00    :  0   :  0   :  x  :
 0 :00000004:00000011000110010100000000100111:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:   00    :   00    :  0   :  0   :  x  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000008:00000011000110010100000000100110:  18  :  19  :  7   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:   00    :   00    :  0   :  0   :  x  :
 0 :00000008:00000011000110010100000000100110:  18  :  19  :  7   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:   00    :   00    :  0   :  0   :  x  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :0000000c:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  18  :  19  :  6   :00000018:00000019:ffffffe6:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:   00    :   00    :  0   :  0   :  x  :
 0 :0000000c:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  18  :  19  :  6   :00000018:00000019:ffffffe6:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:   00    :   00    :  0   :  0   :  x  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000010:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000019:00000001:ffffffe6:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :00084820:00000008:   00    :   00    :  0   :  0   :  x  :
 0 :00000010:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000019:00000001:ffffffe6:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :00084820:00000008:   00    :   00    :  0   :  0   :  x  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000014:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000019:00000001:00000001:  xxxxxxxx  : 00084820 : ffffffe6 :00084820:ffffffe6:   00    :   00    :  0   :  0   :  x  :
 0 :00000014:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000019:00000001:00000001:  xxxxxxxx  : 00084820 : ffffffe6 :00084820:ffffffe6:   00    :   00    :  0   :  0   :  x  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000018:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000019:00000001:00000001:  xxxxxxxx  : 00084820 : 00000001 :00084820:00000001:   00    :   00    :  0   :  0   :  x  :
 0 :00000018:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000019:00000001:00000001:  xxxxxxxx  : 00084820 : 00000001 :00084820:00000001:   00    :   00    :  0   :  0   :  x  :
lijiaruideMacBook-Pro:P4 coding lijiarui$
```

Inputted instruction:

nor: 000000_11000_11001_01000_00000_100111

xor: 000000_11000_11001_01000_00000_100110

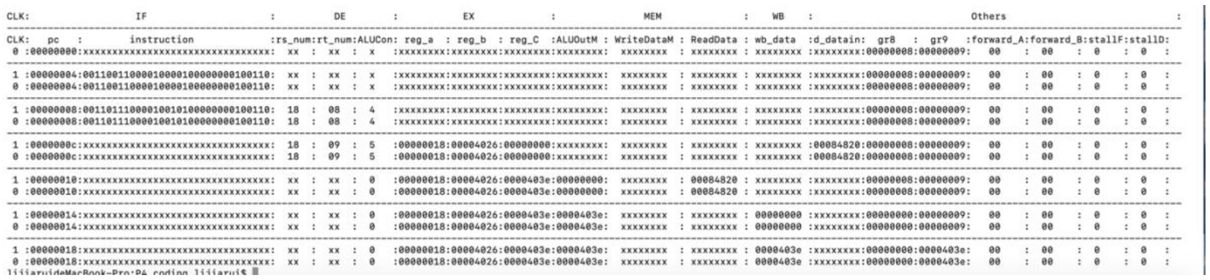
For <code>nor</code> & xor, same rs, rt and rd as and & or instruction are used to test. Therefore, nor & xor instructions are performed on values of `gr24 and `gr25 this time. As it is shown, the result are

32’b1111_1111_1111_1111_1111_1111_1110_0110,

32’b0000_0000_0000_0000_0000_0000_0000_0001;

respectively. The pipelined procedure is the same as previous.

F. andi & ori



```
CLK:                 IF                   :         DE        :          EX           :                MEM            :    WB    :                           Others                         :
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
CLK:   pc    :        instruction          :rs_num:rt_num:ALUCon: reg_a   : reg_b   : reg_C   :ALUOutM : WriteDataM : ReadData : wb_data  :d_datain:  gr8    :  gr9    :forward_A:forward_B:stallF:stallD:
 0 :00000000:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:   00    :   00    :  0   :  0   :
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000004:00110011000010000100000000100110:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:   00    :   00    :  0   :  0   :
 0 :00000004:00110011000010000100000000100110:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:   00    :   00    :  0   :  0   :
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000008:00110111000010010100000000100110:  18  :  08  :  4   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:   00    :   00    :  0   :  0   :
 0 :00000008:00110111000010010100000000100110:  18  :  08  :  4   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:   00    :   00    :  0   :  0   :
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :0000000c:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  18  :  09  :  5   :00000018:00004026:00000000:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :00084820:00000008:00000009:   00    :   00    :  0   :  0   :
 0 :0000000c:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  18  :  09  :  5   :00000018:00004026:00000000:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :00084820:00000008:00000009:   00    :   00    :  0   :  0   :
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000010:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00004026:0000403e:00000000:  xxxxxxxx  : 00084820 : xxxxxxxx :xxxxxxxx:00000008:00000009:   00    :   00    :  0   :  0   :
 0 :00000010:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00004026:0000403e:00000000:  xxxxxxxx  : 00084820 : xxxxxxxx :xxxxxxxx:00000008:00000009:   00    :   00    :  0   :  0   :
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000014:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00004026:0000403e:0000403e:  xxxxxxxx  : xxxxxxxx : 00000000 :xxxxxxxx:00000000:00000009:   00    :   00    :  0   :  0   :
 0 :00000014:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00004026:0000403e:0000403e:  xxxxxxxx  : xxxxxxxx : 00000000 :xxxxxxxx:00000000:00000009:   00    :   00    :  0   :  0   :
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000018:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00004026:0000403e:0000403e:  xxxxxxxx  : xxxxxxxx : 0000403e :xxxxxxxx:00000000:0000403e:   00    :   00    :  0   :  0   :
 0 :00000018:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00004026:0000403e:0000403e:  xxxxxxxx  : xxxxxxxx : 0000403e :xxxxxxxx:00000000:0000403e:   00    :   00    :  0   :  0   :
lijiaruideMacBook-Pro:P4 coding lijiarui$
```

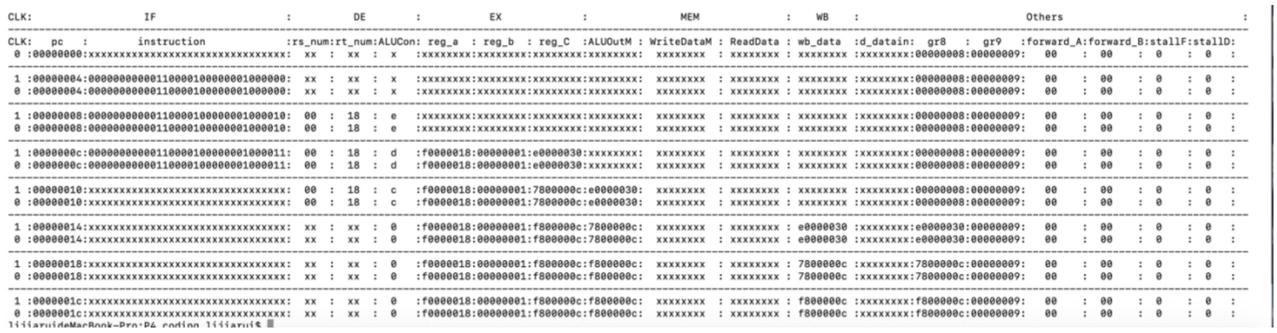
Inputted instructions:

andi: 001100_11000_01000_01000_00000_100110

ori: 001101_11000_01001_01000_00000_100110

In <code>andi</code> & <code>ori</code> instructions, ALU operations are performed on rs (`gr24 here) and extended immediate number. Therefore, now in the third clock rise, reg_a is the value fetched from `gr24 while reg_b is now the extended immediate number. And in fifth and sixth stage, we can see the result is written back to `gr8 and `gr9 separately.

G. sll & srl & sra



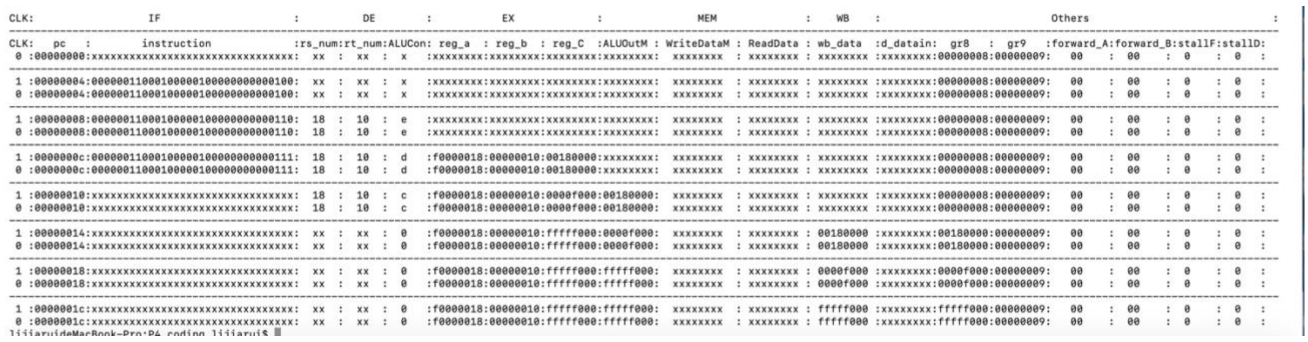
```
CLK:                 IF                     :           DE          :            EX              :                MEM              :     WB     :                           Others                            :
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
CLK:   pc   :          instruction          :rs_num:rt_num:ALUCon: reg_a  : reg_b  : reg_C  :ALUOutM : WriteDataM : ReadData : wb_data  :d_datain:  gr8   :  gr9   :forward_A:forward_B:stallF:stallD:
 0 :00000000:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:   00    :  00    :  0   :  0   :
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000004:00000000000110000100000001000000:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:   00    :  00    :  0   :  0   :
 0 :00000004:00000000000110000100000001000000:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:   00    :  00    :  0   :  0   :
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000008:00000000000110000100000001000010:  00  :  18  :  e   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:   00    :  00    :  0   :  0   :
 0 :00000008:00000000000110000100000001000010:  00  :  18  :  e   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:   00    :  00    :  0   :  0   :
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :0000000c:00000000000110000100000001000011:  00  :  18  :  d   :f0000018:00000001:e0000030:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:   00    :  00    :  0   :  0   :
 0 :0000000c:00000000000110000100000001000011:  00  :  18  :  d   :f0000018:00000001:e0000030:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:   00    :  00    :  0   :  0   :
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000010:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  00  :  18  :  c   :f0000018:00000001:7800000c:e0000030:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:   00    :  00    :  0   :  0   :
 0 :00000010:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  00  :  18  :  c   :f0000018:00000001:7800000c:e0000030:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:   00    :  00    :  0   :  0   :
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000014:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :f0000018:00000001:f800000c:7800000c:  xxxxxxxx  : xxxxxxxx : e0000030 :xxxxxxxx:e0000030:00000009:   00    :  00    :  0   :  0   :
 0 :00000014:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :f0000018:00000001:f800000c:7800000c:  xxxxxxxx  : xxxxxxxx : e0000030 :xxxxxxxx:e0000030:00000009:   00    :  00    :  0   :  0   :
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000018:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :f0000018:00000001:f800000c:f800000c:  xxxxxxxx  : xxxxxxxx : 7800000c :xxxxxxxx:7800000c:00000009:   00    :  00    :  0   :  0   :
 0 :00000018:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :f0000018:00000001:f800000c:f800000c:  xxxxxxxx  : xxxxxxxx : 7800000c :xxxxxxxx:7800000c:00000009:   00    :  00    :  0   :  0   :
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :0000001c:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :f0000018:00000001:f800000c:f800000c:  xxxxxxxx  : xxxxxxxx : f800000c :xxxxxxxx:f800000c:00000009:   00    :  00    :  0   :  0   :
 0 :0000001c:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :f0000018:00000001:f800000c:f800000c:  xxxxxxxx  : xxxxxxxx : f800000c :xxxxxxxx:f800000c:00000009:   00    :  00    :  0   :  0   :
lijiaruideMacBook-Pro:P4_coding lijiarui$
```

Inputted instructions:

sll:	000000_00000_11000_01000_00001_000000
srl:	000000_00000_11000_01000_00001_000010
sra:	000000_00000_11000_01000_00001_000011

Notice that to see the difference between srl and sra, the value stored in `gr24 is temporarily set to 32’hf000_0018. Therefore, each instruction is trying to shift the value in `gr24 by I bit according to its direction and write the result back to gr`8. The whole process is shown in the picture.

H. sllv & srlv & srav



```
CLK:                IF                     :          DE         :           EX              :                 MEM               :    WB    :                              Others                              :
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
CLK:   pc    :         instruction          :rs_num:rt_num:ALUCon: reg_a   : reg_b   : reg_C   :ALUOutM : WriteDataM : ReadData : wb_data  :d_datain:  gr8    :  gr9    :forward_A:forward_B:stallF:stallD:
 0 :00000000:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:   00    :  00     :  0   :  0   :
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000004:00000011000100000100000000000100:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:   00    :  00     :  0   :  0   :
 0 :00000004:00000011000100000100000000000100:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:   00    :  00     :  0   :  0   :
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000008:00000011000100000100000000000110:  18  :  10  :  e   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:   00    :  00     :  0   :  0   :
 0 :00000008:00000011000100000100000000000110:  18  :  10  :  e   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:   00    :  00     :  0   :  0   :
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :0000000c:00000011000100000100000000000111:  18  :  10  :  d   :f0000018:00000010:00180000:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:   00    :  00     :  0   :  0   :
 0 :0000000c:00000011000100000100000000000111:  18  :  10  :  d   :f0000018:00000010:00180000:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:   00    :  00     :  0   :  0   :
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000010:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  18  :  10  :  c   :f0000018:00000010:0000f000:00180000:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:   00    :  00     :  0   :  0   :
 0 :00000010:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  18  :  10  :  c   :f0000018:00000010:0000f000:00180000:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:   00    :  00     :  0   :  0   :
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000014:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :f0000018:00000010:fffff000:0000f000:  xxxxxxxx  : xxxxxxxx : 00180000 :xxxxxxxx:00180000:00000009:   00    :  00     :  0   :  0   :
 0 :00000014:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :f0000018:00000010:fffff000:0000f000:  xxxxxxxx  : xxxxxxxx : 00180000 :xxxxxxxx:00180000:00000009:   00    :  00     :  0   :  0   :
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000018:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :f0000018:00000010:fffff000:fffff000:  xxxxxxxx  : xxxxxxxx : 0000f000 :xxxxxxxx:0000f000:00000009:   00    :  00     :  0   :  0   :
 0 :00000018:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :f0000018:00000010:fffff000:fffff000:  xxxxxxxx  : xxxxxxxx : 0000f000 :xxxxxxxx:0000f000:00000009:   00    :  00     :  0   :  0   :
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :0000001c:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :f0000018:00000010:fffff000:fffff000:  xxxxxxxx  : xxxxxxxx : fffff000 :xxxxxxxx:fffff000:00000009:   00    :  00     :  0   :  0   :
 0 :0000001c:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :f0000018:00000010:fffff000:fffff000:  xxxxxxxx  : xxxxxxxx : fffff000 :xxxxxxxx:fffff000:00000009:   00    :  00     :  0   :  0   :
lijiaruideMacBook-Pro:P4 coding lijiarui$
```

Inputted instructions:

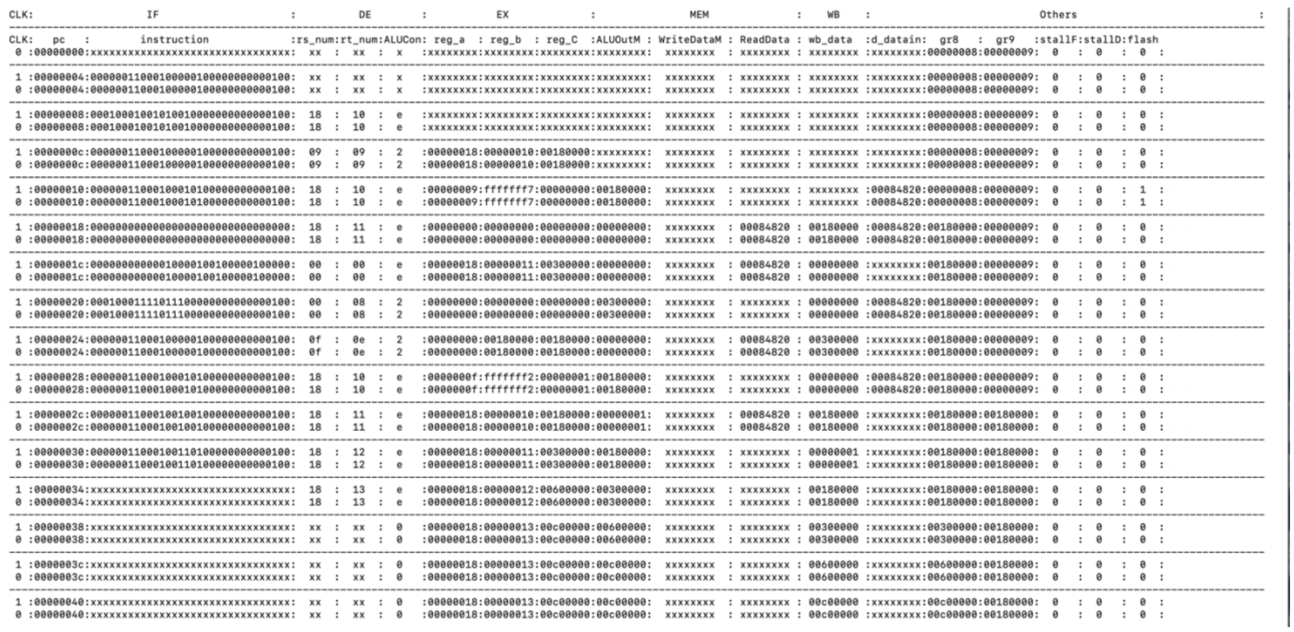
sllv: 000000_11000_10000_01000_00000_000100

srlv: 000000_11000_10000_01000_00000_000110

srav: 000000_11000_10000_01000_00000_000111

In this test case, `gr24 is still holding 32'hf000_0018. Since all instructions are using value in `gr16 as the shift amount, the values of reg_C in different clock rise are 4 place away from where it was in it corresponding reg_a value. Therefore, the test case stands.

I. beq



```
CLK:                IF                     :          DE         :           EX              :                 MEM               :    WB    :                        Others                        :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
CLK:   pc    :         instruction          :rs_num:rt_num:ALUCon: reg_a   : reg_b   : reg_C   :ALUOutM : WriteDataM : ReadData : wb_data  :d_datain:  gr8    :  gr9    :stallF:stallD:flash
 0 :00000000:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:  0   :  0   :  0  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000004:00000011000100000100000000000100:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:  0   :  0   :  0  :
 0 :00000004:00000011000100000100000000000100:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:  0   :  0   :  0  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000008:00010001001010010000000000000100:  18  :  10  :  e   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:  0   :  0   :  0  :
 0 :00000008:00010001001010010000000000000100:  18  :  10  :  e   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:  0   :  0   :  0  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :0000000c:00000011000100000100000000000100:  09  :  09  :  2   :00000018:00000010:00180000:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:  0   :  0   :  0  :
 0 :0000000c:00000011000100000100000000000100:  09  :  09  :  2   :00000018:00000010:00180000:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:  0   :  0   :  0  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000010:00000011000100010100000000000100:  18  :  10  :  e   :00000009:fffffff7:00000000:00180000:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :00084820:00000008:00000009:  0   :  0   :  1  :
 0 :00000010:00000011000100010100000000000100:  18  :  10  :  e   :00000009:fffffff7:00000000:00180000:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :00084820:00000008:00000009:  0   :  0   :  1  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000018:00000000000000000000000000000000:  18  :  11  :  e   :00000000:00000000:00000000:00000000:  xxxxxxxx  : 00084820 : 00180000 :00084820:00180000:00000009:  0   :  0   :  0  :
 0 :00000018:00000000000000000000000000000000:  18  :  11  :  e   :00000000:00000000:00000000:00000000:  xxxxxxxx  : 00084820 : 00180000 :00084820:00180000:00000009:  0   :  0   :  0  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :0000001c:00000000000010000100100000100000:  00  :  00  :  e   :00000018:00000011:00300000:00000000:  xxxxxxxx  : 00084820 : 00000000 :xxxxxxxx:00180000:00000009:  0   :  0   :  0  :
 0 :0000001c:00000000000010000100100000100000:  00  :  00  :  e   :00000018:00000011:00300000:00000000:  xxxxxxxx  : 00084820 : 00000000 :xxxxxxxx:00180000:00000009:  0   :  0   :  0  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000020:00010001111011100000000000000100:  00  :  08  :  2   :00000000:00000000:00000000:00300000:  xxxxxxxx  : xxxxxxxx : 00000000 :00084820:00180000:00000009:  0   :  0   :  0  :
 0 :00000020:00010001111011100000000000000100:  00  :  08  :  2   :00000000:00000000:00000000:00300000:  xxxxxxxx  : xxxxxxxx : 00000000 :00084820:00180000:00000009:  0   :  0   :  0  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000024:00000011000100000100000000000100:  0f  :  0e  :  2   :00000000:00180000:00180000:00000000:  xxxxxxxx  : 00084820 : 00300000 :xxxxxxxx:00180000:00000009:  0   :  0   :  0  :
 0 :00000024:00000011000100000100000000000100:  0f  :  0e  :  2   :00000000:00180000:00180000:00000000:  xxxxxxxx  : 00084820 : 00300000 :xxxxxxxx:00180000:00000009:  0   :  0   :  0  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000028:00000011000100010100000000000100:  18  :  10  :  e   :0000000f:fffffff2:00000001:00180000:  xxxxxxxx  : xxxxxxxx : 00000000 :00084820:00180000:00000009:  0   :  0   :  0  :
 0 :00000028:00000011000100010100000000000100:  18  :  10  :  e   :0000000f:fffffff2:00000001:00180000:  xxxxxxxx  : xxxxxxxx : 00000000 :00084820:00180000:00000009:  0   :  0   :  0  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :0000002c:00000011000100100100000000000100:  18  :  11  :  e   :00000018:00000010:00180000:00000001:  xxxxxxxx  : 00084820 : 00180000 :xxxxxxxx:00180000:00180000:  0   :  0   :  0  :
 0 :0000002c:00000011000100100100000000000100:  18  :  11  :  e   :00000018:00000010:00180000:00000001:  xxxxxxxx  : 00084820 : 00180000 :xxxxxxxx:00180000:00180000:  0   :  0   :  0  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000030:00000011000100110100000000000100:  18  :  12  :  e   :00000018:00000011:00300000:00180000:  xxxxxxxx  : xxxxxxxx : 00000001 :xxxxxxxx:00180000:00180000:  0   :  0   :  0  :
 0 :00000030:00000011000100110100000000000100:  18  :  12  :  e   :00000018:00000011:00300000:00180000:  xxxxxxxx  : xxxxxxxx : 00000001 :xxxxxxxx:00180000:00180000:  0   :  0   :  0  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000034:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  18  :  13  :  e   :00000018:00000012:00600000:00300000:  xxxxxxxx  : xxxxxxxx : 00180000 :xxxxxxxx:00180000:00180000:  0   :  0   :  0  :
 0 :00000034:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  18  :  13  :  e   :00000018:00000012:00600000:00300000:  xxxxxxxx  : xxxxxxxx : 00180000 :xxxxxxxx:00180000:00180000:  0   :  0   :  0  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000038:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000013:00c00000:00600000:  xxxxxxxx  : xxxxxxxx : 00300000 :xxxxxxxx:00300000:00180000:  0   :  0   :  0  :
 0 :00000038:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000013:00c00000:00600000:  xxxxxxxx  : xxxxxxxx : 00300000 :xxxxxxxx:00300000:00180000:  0   :  0   :  0  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :0000003c:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000013:00c00000:00c00000:  xxxxxxxx  : xxxxxxxx : 00600000 :xxxxxxxx:00600000:00180000:  0   :  0   :  0  :
 0 :0000003c:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000013:00c00000:00c00000:  xxxxxxxx  : xxxxxxxx : 00600000 :xxxxxxxx:00600000:00180000:  0   :  0   :  0  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000040:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000013:00c00000:00c00000:  xxxxxxxx  : xxxxxxxx : 00c00000 :xxxxxxxx:00c00000:00180000:  0   :  0   :  0  :
 0 :00000040:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000013:00c00000:00c00000:  xxxxxxxx  : xxxxxxxx : 00c00000 :xxxxxxxx:00c00000:00180000:  0   :  0   :  0  :
```

Inputted instructions:

000000_11000_10000_01000_00000_000100

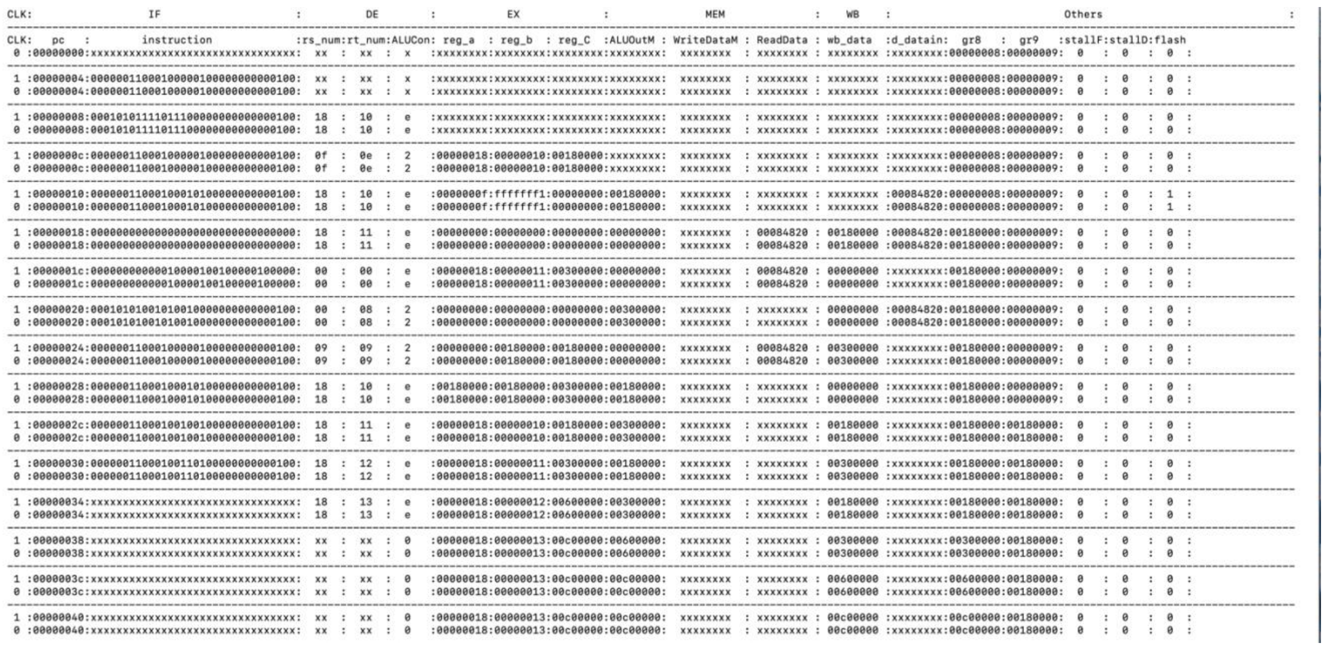
```
beq:    000100_01001_01001_0000_0000_0000_0100 (condition meet)
        000000_11000_10000_01000_00000_000100 (flash)
        000000_11000_10001_01000_00000_000100 (flash)
        000000_11000_10010_01000_00000_000100
        000000_11000_10011_01000_00000_000100
        000000_00000_01000_01001_00000_100000 (branch to here)
beq:    000100_01111_01110_0000_0000_0000_0100 (condition fail)
        000000_11000_10000_01000_00000_000100
        000000_11000_10001_01000_00000_000100
        000000_11000_10010_01000_00000_000100
        000000_11000_10011_01000_00000_000100
```

Branch instructions make use of the pc relative branching logic. This means that if the branch condition is met, in the next line of branch instruction to be executed should be the instruction at location pc plus offset value. Note that now pc is already incremented by four. Also in branch case, control hazard should be handled. After branch instruction passes its ALU stage, if branch occurred, the previous flip flops should be flashed and pc should be set to the branched address. When branch instruction arrives its WB stage, the new instruction should be fetched in.

As it is shown in the picture, the second instruction is <code>beq</code> and it meets the condition. Therefore, in the fourth overall clock rise (third clock rise for this <code>beq</code>), the <code>flash</code> signal is set to be one. In the next clock rise, all previous flip flops are flashed with pc set to the branched location in the meantime. And in the clock rise following, the instruction at the branched location is fetched in. A successful <code>beq</code> with control hazard handling is performed. Besides, notice that for the first instruction, the instruction before this <code>beq</code>, its operation is not affected.

This <code>sll</code> instruction successfully finished and writes the result to `gr8.

J. bne



```
CLK:               IF                  :          DE        :           EX             :             MEM            :    WB    :                         Others                          :
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
CLK:   pc   :        instruction          :rs_num:rt_num:ALUCon: reg_a  : reg_b  : reg_C  :ALUOutM : WriteDataM : ReadData : wb_data  :d_datain:  gr8   :  gr9   :stallF:stallD:flash
 0 :00000000:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:  0   :  0   :  0  :
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000004:00000011000100000100000000000100:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:  0   :  0   :  0  :
 0 :00000004:00000011000100000100000000000100:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:  0   :  0   :  0  :
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000008:00010101111011100000000000000100:  18  :  10  :  e   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:  0   :  0   :  0  :
 0 :00000008:00010101111011100000000000000100:  18  :  10  :  e   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:  0   :  0   :  0  :
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :0000000c:00000011000100000100000000000100:  0f  :  0e  :  2   :00000018:00000010:00180000:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:  0   :  0   :  0  :
 0 :0000000c:00000011000100000100000000000100:  0f  :  0e  :  2   :00000018:00000010:00180000:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:  0   :  0   :  0  :
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000010:00000011000100010100000000000100:  18  :  10  :  e   :0000000f:fffffff1:00000000:00180000:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :00084820:00000008:00000009:  0   :  0   :  1  :
 0 :00000010:00000011000100010100000000000100:  18  :  10  :  e   :0000000f:fffffff1:00000000:00180000:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :00084820:00000008:00000009:  0   :  0   :  1  :
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000018:00000000000000000000000000000000:  18  :  11  :  e   :00000000:00000000:00000000:00000000:  xxxxxxxx  : 00084820 : 00180000 :00084820:00180000:00000009:  0   :  0   :  0  :
 0 :00000018:00000000000000000000000000000000:  18  :  11  :  e   :00000000:00000000:00000000:00000000:  xxxxxxxx  : 00084820 : 00180000 :00084820:00180000:00000009:  0   :  0   :  0  :
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :0000001c:00000000000010000100100000100000:  00  :  00  :  e   :00000018:00000011:00300000:00000000:  xxxxxxxx  : 00084820 : 00000000 :xxxxxxxx:00180000:00000009:  0   :  0   :  0  :
 0 :0000001c:00000000000010000100100000100000:  00  :  00  :  e   :00000018:00000011:00300000:00000000:  xxxxxxxx  : 00084820 : 00000000 :xxxxxxxx:00180000:00000009:  0   :  0   :  0  :
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000020:00010101001010010000000000000100:  00  :  08  :  2   :00000000:00000000:00000000:00300000:  xxxxxxxx  : xxxxxxxx : 00000000 :00084820:00180000:00000009:  0   :  0   :  0  :
 0 :00000020:00010101001010010000000000000100:  00  :  08  :  2   :00000000:00000000:00000000:00300000:  xxxxxxxx  : xxxxxxxx : 00000000 :00084820:00180000:00000009:  0   :  0   :  0  :
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000024:00000011000100000100000000000100:  09  :  09  :  2   :00000000:00180000:00180000:00000000:  xxxxxxxx  : 00084820 : 00300000 :xxxxxxxx:00180000:00000009:  0   :  0   :  0  :
 0 :00000024:00000011000100000100000000000100:  09  :  09  :  2   :00000000:00180000:00180000:00000000:  xxxxxxxx  : 00084820 : 00300000 :xxxxxxxx:00180000:00000009:  0   :  0   :  0  :
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000028:00000011000100010100000000000100:  18  :  10  :  e   :00180000:00180000:00300000:00180000:  xxxxxxxx  : xxxxxxxx : 00000000 :xxxxxxxx:00180000:00000009:  0   :  0   :  0  :
 0 :00000028:00000011000100010100000000000100:  18  :  10  :  e   :00180000:00180000:00300000:00180000:  xxxxxxxx  : xxxxxxxx : 00000000 :xxxxxxxx:00180000:00000009:  0   :  0   :  0  :
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :0000002c:00000011000100100100000000000100:  18  :  11  :  e   :00000018:00000010:00180000:00300000:  xxxxxxxx  : xxxxxxxx : 00180000 :xxxxxxxx:00180000:00180000:  0   :  0   :  0  :
 0 :0000002c:00000011000100100100000000000100:  18  :  11  :  e   :00000018:00000010:00180000:00300000:  xxxxxxxx  : xxxxxxxx : 00180000 :xxxxxxxx:00180000:00180000:  0   :  0   :  0  :
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000030:00000011000100110100000000000100:  18  :  12  :  e   :00000018:00000011:00300000:00180000:  xxxxxxxx  : xxxxxxxx : 00300000 :xxxxxxxx:00180000:00180000:  0   :  0   :  0  :
 0 :00000030:00000011000100110100000000000100:  18  :  12  :  e   :00000018:00000011:00300000:00180000:  xxxxxxxx  : xxxxxxxx : 00300000 :xxxxxxxx:00180000:00180000:  0   :  0   :  0  :
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000034:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  18  :  13  :  e   :00000018:00000012:00600000:00300000:  xxxxxxxx  : xxxxxxxx : 00180000 :xxxxxxxx:00180000:00180000:  0   :  0   :  0  :
 0 :00000034:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  18  :  13  :  e   :00000018:00000012:00600000:00300000:  xxxxxxxx  : xxxxxxxx : 00180000 :xxxxxxxx:00180000:00180000:  0   :  0   :  0  :
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000038:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000013:00c00000:00600000:  xxxxxxxx  : xxxxxxxx : 00300000 :xxxxxxxx:00300000:00180000:  0   :  0   :  0  :
 0 :00000038:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000013:00c00000:00600000:  xxxxxxxx  : xxxxxxxx : 00300000 :xxxxxxxx:00300000:00180000:  0   :  0   :  0  :
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :0000003c:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000013:00c00000:00c00000:  xxxxxxxx  : xxxxxxxx : 00600000 :xxxxxxxx:00600000:00180000:  0   :  0   :  0  :
 0 :0000003c:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000013:00c00000:00c00000:  xxxxxxxx  : xxxxxxxx : 00600000 :xxxxxxxx:00600000:00180000:  0   :  0   :  0  :
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000040:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000013:00c00000:00c00000:  xxxxxxxx  : xxxxxxxx : 00c00000 :xxxxxxxx:00c00000:00180000:  0   :  0   :  0  :
 0 :00000040:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000013:00c00000:00c00000:  xxxxxxxx  : xxxxxxxx : 00c00000 :xxxxxxxx:00c00000:00180000:  0   :  0   :  0  :
```

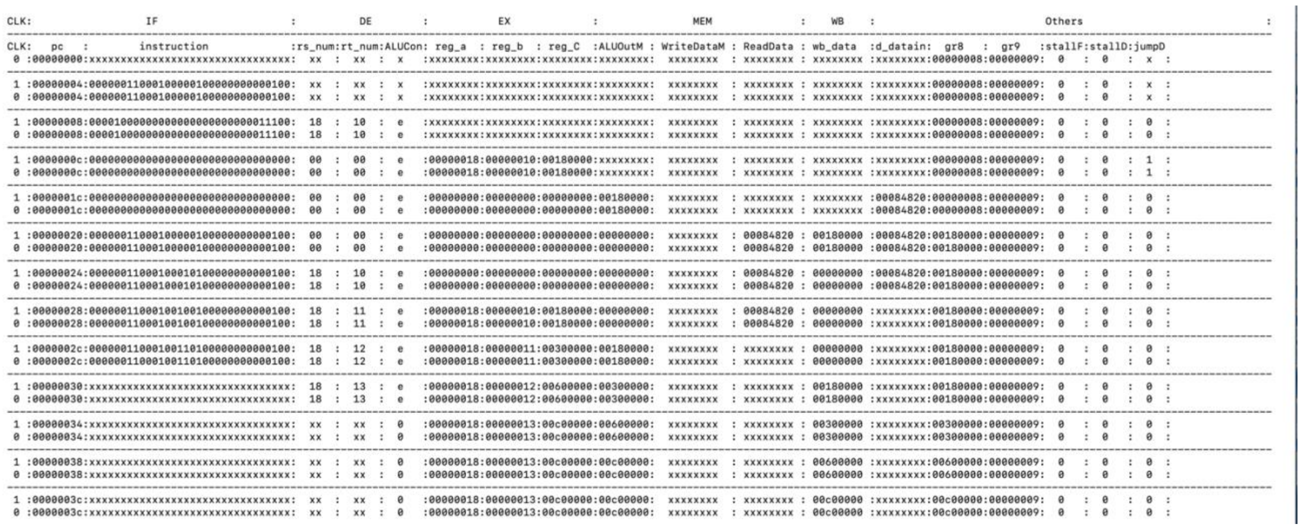
Inputted instructions:

	000000_11000_10000_01000_00000_000100
bne:	000101_01111_01110_0000_0000_0000_0100 (condition meet)
	000000_11000_10000_01000_00000_000100 (flash)
	000000_11000_10001_01000_00000_000100 (flash)
	000000_11000_10010_01000_00000_000100
	000000_11000_10011_01000_00000_000100
	000000_00000_01000_01001_00000_100000 (branch to here)
bne:	000101_01001_01001_0000_0000_0000_0100 (condition fail)
	000000_11000_10000_01000_00000_000100
	000000_11000_10001_01000_00000_000100
	000000_11000_10010_01000_00000_000100

000000_11000_10011_01000_00000_000100

Since <code>bne</code> is just the opposite instruction of <code>bnq</code>, the combination of test instructions is basically the same, the only difference is the two branch instructions switched its location and are revised to <code>bne</code> op code. Therefore, it is easy to understand following the previous illustration.

K. j



```
CLK:                IF                        :          DE         :          EX             :               MEM              :   WB    :                          Others                          :
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
CLK:   pc   :         instruction             :rs_num:rt_num:ALUCon: reg_a   : reg_b   : reg_C   :ALUOutM : WriteDataM : ReadData : wb_data  :d_datain:  gr8    :  gr9   :stallF:stallD:jumpD
 0 :00000000:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:  0   :  0   :  x  :
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000004:00000011000100000100000000000100:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:  0   :  0   :  x  :
 0 :00000004:00000011000100000100000000000100:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:  0   :  0   :  x  :
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000008:00001000000000000000000000011100:  18  :  10  :  e   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:  0   :  0   :  0  :
 0 :00000008:00001000000000000000000000011100:  18  :  10  :  e   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:  0   :  0   :  0  :
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :0000000c:00000000000000000000000000000000:  00  :  00  :  e   :00000018:00000010:00180000:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:  0   :  0   :  1  :
 0 :0000000c:00000000000000000000000000000000:  00  :  00  :  e   :00000018:00000010:00180000:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:00000009:  0   :  0   :  1  :
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :0000001c:00000000000000000000000000000000:  00  :  00  :  e   :00000000:00000000:00000000:00180000:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :00084820:00000008:00000009:  0   :  0   :  0  :
 0 :0000001c:00000000000000000000000000000000:  00  :  00  :  e   :00000000:00000000:00000000:00180000:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :00084820:00000008:00000009:  0   :  0   :  0  :
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000020:00000011000100000100000000000100:  00  :  00  :  e   :00000000:00000000:00000000:00000000:  xxxxxxxx  : 00084820 : 00180000 :00084820:00180000:00000009:  0   :  0   :  0  :
 0 :00000020:00000011000100000100000000000100:  00  :  00  :  e   :00000000:00000000:00000000:00000000:  xxxxxxxx  : 00084820 : 00180000 :00084820:00180000:00000009:  0   :  0   :  0  :
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000024:00000011000100010100000000000100:  18  :  10  :  e   :00000000:00000000:00000000:00000000:  xxxxxxxx  : 00084820 : 00000000 :00084820:00180000:00000009:  0   :  0   :  0  :
 0 :00000024:00000011000100010100000000000100:  18  :  10  :  e   :00000000:00000000:00000000:00000000:  xxxxxxxx  : 00084820 : 00000000 :00084820:00180000:00000009:  0   :  0   :  0  :
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000028:00000011000100100100000000000100:  18  :  11  :  e   :00000018:00000010:00180000:00000000:  xxxxxxxx  : 00084820 : 00000000 :xxxxxxxx:00180000:00000009:  0   :  0   :  0  :
 0 :00000028:00000011000100100100000000000100:  18  :  11  :  e   :00000018:00000010:00180000:00000000:  xxxxxxxx  : 00084820 : 00000000 :xxxxxxxx:00180000:00000009:  0   :  0   :  0  :
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :0000002c:00000011000100110100000000000100:  18  :  12  :  e   :00000018:00000011:00300000:00180000:  xxxxxxxx  : xxxxxxxx : 00000000 :xxxxxxxx:00180000:00000009:  0   :  0   :  0  :
 0 :0000002c:00000011000100110100000000000100:  18  :  12  :  e   :00000018:00000011:00300000:00180000:  xxxxxxxx  : xxxxxxxx : 00000000 :xxxxxxxx:00180000:00000009:  0   :  0   :  0  :
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000030:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  18  :  13  :  e   :00000018:00000012:00600000:00300000:  xxxxxxxx  : xxxxxxxx : 00180000 :xxxxxxxx:00180000:00000009:  0   :  0   :  0  :
 0 :00000030:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  18  :  13  :  e   :00000018:00000012:00600000:00300000:  xxxxxxxx  : xxxxxxxx : 00180000 :xxxxxxxx:00180000:00000009:  0   :  0   :  0  :
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000034:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000013:00c00000:00600000:  xxxxxxxx  : xxxxxxxx : 00300000 :xxxxxxxx:00300000:00000009:  0   :  0   :  0  :
 0 :00000034:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000013:00c00000:00600000:  xxxxxxxx  : xxxxxxxx : 00300000 :xxxxxxxx:00300000:00000009:  0   :  0   :  0  :
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000038:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000013:00c00000:00c00000:  xxxxxxxx  : xxxxxxxx : 00600000 :xxxxxxxx:00600000:00000009:  0   :  0   :  0  :
 0 :00000038:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000013:00c00000:00c00000:  xxxxxxxx  : xxxxxxxx : 00600000 :xxxxxxxx:00600000:00000009:  0   :  0   :  0  :
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :0000003c:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000013:00c00000:00c00000:  xxxxxxxx  : xxxxxxxx : 00c00000 :xxxxxxxx:00c00000:00000009:  0   :  0   :  0  :
 0 :0000003c:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000013:00c00000:00c00000:  xxxxxxxx  : xxxxxxxx : 00c00000 :xxxxxxxx:00c00000:00000009:  0   :  0   :  0  :
```

Inputted instructions:

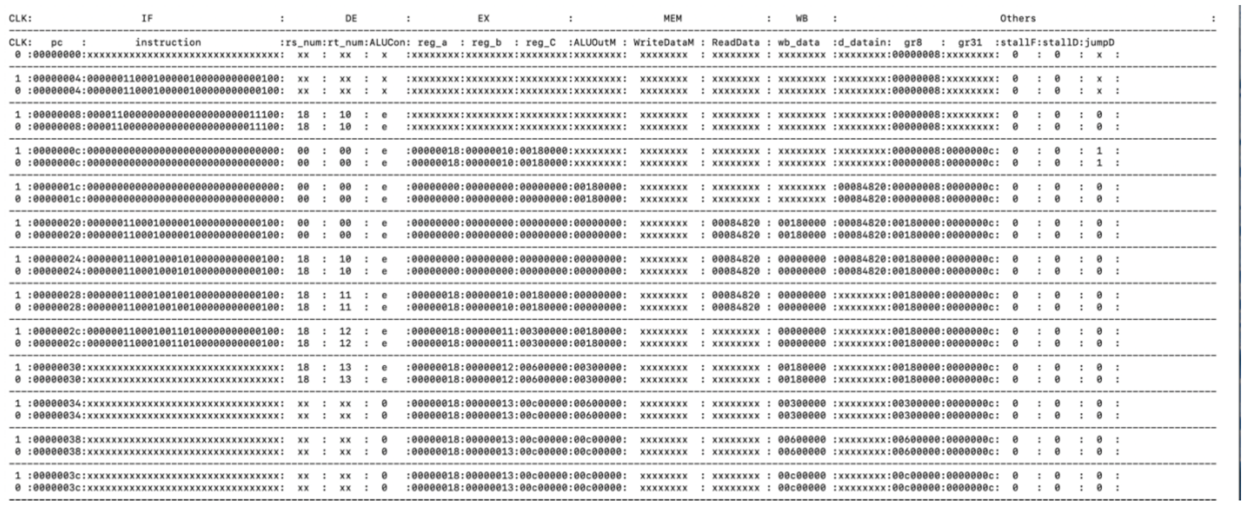
	000000_11000_10000_01000_00000_000100
j:	000010_00000_00000_0000_0000_0001_1100
	000000_11000_10000_01000_00000_000100
	000000_11000_10001_01000_00000_000100
	000000_11000_10010_01000_00000_000100
	000000_11000_10011_01000_00000_000100
	000000_00000_01000_01001_00000_100000
	000000_11000_10000_01000_00000_000100 (jump target)
	000000_11000_10001_01000_00000_000100
	000000_11000_10010_01000_00000_000100

000000_11000_10011_01000_00000_000100

<code>j</code> instruction will cause the execution of the program jump to the address of the least significant 26 bits of the instruction line. The main differences are <code>j</code> need not to do comparison. Therefore, the jump instruction could be determined right after the decoding stage.

As it is shown in the picture, in the end of third overall clock rise (for <code>j</code> is its second), jumpD is set to one. In the following clock rise, pc is set to the jump address and first two flip flops are flashed. After that, in the next clock rise, the instruction at the jump address is fetched in. Then the following is normal execution.

L. jal



```
CLK:                IF                      :          DE         :            EX             :              MEM              :    WB    :                          Others                          :
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
CLK:   pc    :          instruction          :rs_num:rt_num:ALUCon: reg_a  : reg_b  : reg_C  :ALUOutM : WriteDataM : ReadData : wb_data  :d_datain:  gr8   :  gr31  :stallF:stallD:jumpD
 0 :00000000:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:xxxxxxxx:  0   :  0   :  x  :
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000004:00000011000100000100000000000100:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:xxxxxxxx:  0   :  0   :  x  :
 0 :00000004:00000011000100000100000000000100:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:xxxxxxxx:  0   :  0   :  x  :
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000008:00001100000000000000000000011100:  18  :  10  :  e   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:xxxxxxxx:  0   :  0   :  0  :
 0 :00000008:00001100000000000000000000011100:  18  :  10  :  e   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:xxxxxxxx:  0   :  0   :  0  :
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :0000000c:00000000000000000000000000000000:  00  :  00  :  e   :00000018:00000010:00180000:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:0000000c:  0   :  0   :  1  :
 0 :0000000c:00000000000000000000000000000000:  00  :  00  :  e   :00000018:00000010:00180000:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:0000000c:  0   :  0   :  1  :
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :0000001c:00000000000000000000000000000000:  00  :  00  :  e   :00000000:00000000:00000000:00180000:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :00084820:00000008:0000000c:  0   :  0   :  0  :
 0 :0000001c:00000000000000000000000000000000:  00  :  00  :  e   :00000000:00000000:00000000:00180000:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :00084820:00000008:0000000c:  0   :  0   :  0  :
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000020:00000011000100000100000000000100:  00  :  00  :  e   :00000000:00000000:00000000:00000000:  xxxxxxxx  : 00084820 : 00180000 :00084820:00180000:0000000c:  0   :  0   :  0  :
 0 :00000020:00000011000100000100000000000100:  00  :  00  :  e   :00000000:00000000:00000000:00000000:  xxxxxxxx  : 00084820 : 00180000 :00084820:00180000:0000000c:  0   :  0   :  0  :
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000024:00000011000100010100000000000100:  18  :  10  :  e   :00000000:00000000:00000000:00000000:  xxxxxxxx  : 00084820 : 00000000 :00084820:00180000:0000000c:  0   :  0   :  0  :
 0 :00000024:00000011000100010100000000000100:  18  :  10  :  e   :00000000:00000000:00000000:00000000:  xxxxxxxx  : 00084820 : 00000000 :00084820:00180000:0000000c:  0   :  0   :  0  :
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000028:00000011000100100100000000000100:  18  :  11  :  e   :00000018:00000010:00180000:00000000:  xxxxxxxx  : 00084820 : 00000000 :xxxxxxxx:00180000:0000000c:  0   :  0   :  0  :
 0 :00000028:00000011000100100100000000000100:  18  :  11  :  e   :00000018:00000010:00180000:00000000:  xxxxxxxx  : 00084820 : 00000000 :xxxxxxxx:00180000:0000000c:  0   :  0   :  0  :
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :0000002c:00000011000100110100000000000100:  18  :  12  :  e   :00000018:00000011:00300000:00180000:  xxxxxxxx  : xxxxxxxx : 00000000 :xxxxxxxx:00180000:0000000c:  0   :  0   :  0  :
 0 :0000002c:00000011000100110100000000000100:  18  :  12  :  e   :00000018:00000011:00300000:00180000:  xxxxxxxx  : xxxxxxxx : 00000000 :xxxxxxxx:00180000:0000000c:  0   :  0   :  0  :
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000030:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  18  :  13  :  e   :00000018:00000012:00600000:00300000:  xxxxxxxx  : xxxxxxxx : 00180000 :xxxxxxxx:00180000:0000000c:  0   :  0   :  0  :
 0 :00000030:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  18  :  13  :  e   :00000018:00000012:00600000:00300000:  xxxxxxxx  : xxxxxxxx : 00180000 :xxxxxxxx:00180000:0000000c:  0   :  0   :  0  :
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000034:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000013:00c00000:00600000:  xxxxxxxx  : xxxxxxxx : 00300000 :xxxxxxxx:00300000:0000000c:  0   :  0   :  0  :
 0 :00000034:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000013:00c00000:00600000:  xxxxxxxx  : xxxxxxxx : 00300000 :xxxxxxxx:00300000:0000000c:  0   :  0   :  0  :
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000038:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000013:00c00000:00c00000:  xxxxxxxx  : xxxxxxxx : 00600000 :xxxxxxxx:00600000:0000000c:  0   :  0   :  0  :
 0 :00000038:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000013:00c00000:00c00000:  xxxxxxxx  : xxxxxxxx : 00600000 :xxxxxxxx:00600000:0000000c:  0   :  0   :  0  :
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :0000003c:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000013:00c00000:00c00000:  xxxxxxxx  : xxxxxxxx : 00c00000 :xxxxxxxx:00c00000:0000000c:  0   :  0   :  0  :
 0 :0000003c:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000013:00c00000:00c00000:  xxxxxxxx  : xxxxxxxx : 00c00000 :xxxxxxxx:00c00000:0000000c:  0   :  0   :  0  :
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

Inputted instruction:

```
        000000_11000_10000_01000_00000_000100
jal:    000011_00000_00000_0000_0000_0001_1100
        000000_11000_10000_01000_00000_000100
        000000_11000_10001_01000_00000_000100
        000000_11000_10010_01000_00000_000100
        000000_11000_10011_01000_00000_000100
```

000000_00000_01000_01001_00000_100000

000000_11000_10000_01000_00000_000100 (target)

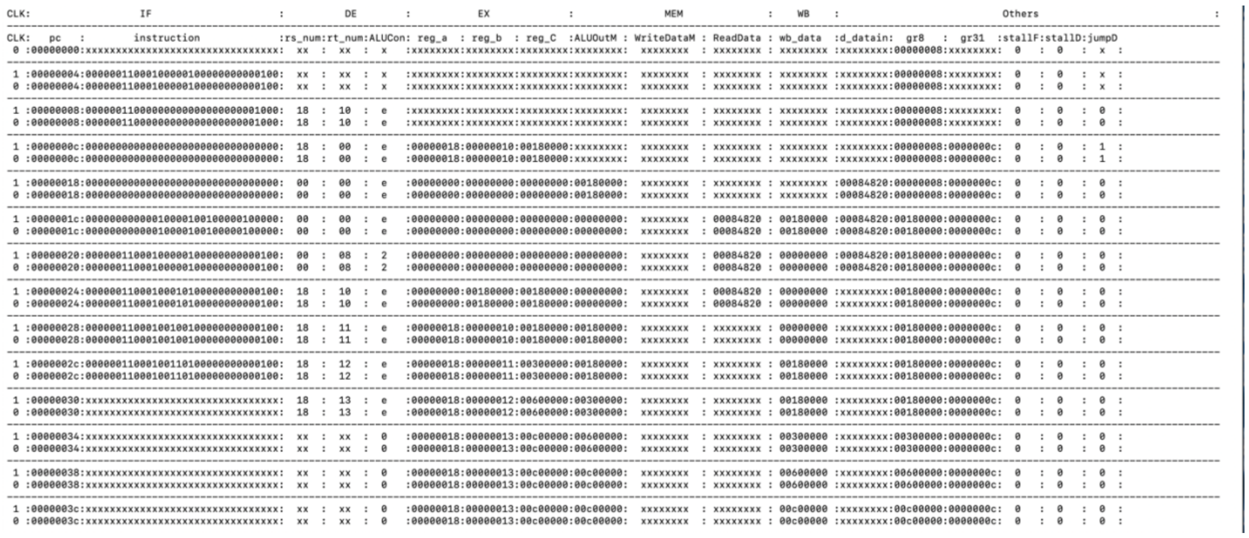
000000_11000_10001_01000_00000_000100

000000_11000_10010_01000_00000_000100

000000_11000_10011_01000_00000_000100

<code>jal</code> act mostly the same as <code>j</code>, except for the former will record the address of next instruction in `gr31 ($ra) for returning. As it is shown in the picture, before jumping, `gr31 is set to 32’h0000_000c.

M. jr



```
CLK:               IF                  :        DE        :          EX            :              MEM            :    WB    :                      Others                        :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
CLK:   pc   :        instruction          :rs_num:rt_num:ALUCon: reg_a  : reg_b  : reg_C  :ALUOutM : WriteDataM : ReadData : wb_data  :d_datain:  gr8   :  gr31  :stallF:stallD:jumpD
 0 :00000000:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:xxxxxxxx:  0   :  0   :  x  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000004:00000011000100000100000000000100:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:xxxxxxxx:  0   :  0   :  x  :
 0 :00000004:00000011000100000100000000000100:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:xxxxxxxx:  0   :  0   :  x  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000008:00000011000000000000000000001000:  18  :  10  :  e   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:xxxxxxxx:  0   :  0   :  0  :
 0 :00000008:00000011000000000000000000001000:  18  :  10  :  e   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:xxxxxxxx:  0   :  0   :  0  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :0000000c:00000000000000000000000000000000:  18  :  00  :  e   :00000018:00000010:00180000:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:0000000c:  0   :  0   :  1  :
 0 :0000000c:00000000000000000000000000000000:  18  :  00  :  e   :00000018:00000010:00180000:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:0000000c:  0   :  0   :  1  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000018:00000000000000000000000000000000:  00  :  00  :  e   :00000000:00000000:00000000:00180000:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :00084820:00000008:0000000c:  0   :  0   :  0  :
 0 :00000018:00000000000000000000000000000000:  00  :  00  :  e   :00000000:00000000:00000000:00180000:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :00084820:00000008:0000000c:  0   :  0   :  0  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :0000001c:00000000000010000100100000100000:  00  :  00  :  e   :00000000:00000000:00000000:00000000:  xxxxxxxx  : 00084820 : 00180000 :00084820:00180000:0000000c:  0   :  0   :  0  :
 0 :0000001c:00000000000010000100100000100000:  00  :  00  :  e   :00000000:00000000:00000000:00000000:  xxxxxxxx  : 00084820 : 00180000 :00084820:00180000:0000000c:  0   :  0   :  0  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000020:00000011000100000100000000000100:  00  :  08  :  2   :00000000:00000000:00000000:00000000:  xxxxxxxx  : 00084820 : 00000000 :00084820:00180000:0000000c:  0   :  0   :  0  :
 0 :00000020:00000011000100000100000000000100:  00  :  08  :  2   :00000000:00000000:00000000:00000000:  xxxxxxxx  : 00084820 : 00000000 :00084820:00180000:0000000c:  0   :  0   :  0  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000024:00000011000100010100000000000100:  18  :  10  :  e   :00000000:00180000:00180000:00000000:  xxxxxxxx  : 00084820 : 00000000 :xxxxxxxx:00180000:0000000c:  0   :  0   :  0  :
 0 :00000024:00000011000100010100000000000100:  18  :  10  :  e   :00000000:00180000:00180000:00000000:  xxxxxxxx  : 00084820 : 00000000 :xxxxxxxx:00180000:0000000c:  0   :  0   :  0  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000028:00000011000100100100000000000100:  18  :  11  :  e   :00000018:00000010:00180000:00180000:  xxxxxxxx  : xxxxxxxx : 00000000 :xxxxxxxx:00180000:0000000c:  0   :  0   :  0  :
 0 :00000028:00000011000100100100000000000100:  18  :  11  :  e   :00000018:00000010:00180000:00180000:  xxxxxxxx  : xxxxxxxx : 00000000 :xxxxxxxx:00180000:0000000c:  0   :  0   :  0  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :0000002c:00000011000100110100000000000100:  18  :  12  :  e   :00000018:00000011:00300000:00180000:  xxxxxxxx  : xxxxxxxx : 00180000 :xxxxxxxx:00180000:0000000c:  0   :  0   :  0  :
 0 :0000002c:00000011000100110100000000000100:  18  :  12  :  e   :00000018:00000011:00300000:00180000:  xxxxxxxx  : xxxxxxxx : 00180000 :xxxxxxxx:00180000:0000000c:  0   :  0   :  0  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000030:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  18  :  13  :  e   :00000018:00000012:00600000:00300000:  xxxxxxxx  : xxxxxxxx : 00180000 :xxxxxxxx:00180000:0000000c:  0   :  0   :  0  :
 0 :00000030:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  18  :  13  :  e   :00000018:00000012:00600000:00300000:  xxxxxxxx  : xxxxxxxx : 00180000 :xxxxxxxx:00180000:0000000c:  0   :  0   :  0  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000034:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000013:00c00000:00600000:  xxxxxxxx  : xxxxxxxx : 00300000 :xxxxxxxx:00300000:0000000c:  0   :  0   :  0  :
 0 :00000034:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000013:00c00000:00600000:  xxxxxxxx  : xxxxxxxx : 00300000 :xxxxxxxx:00300000:0000000c:  0   :  0   :  0  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000038:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000013:00c00000:00c00000:  xxxxxxxx  : xxxxxxxx : 00600000 :xxxxxxxx:00600000:0000000c:  0   :  0   :  0  :
 0 :00000038:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000013:00c00000:00c00000:  xxxxxxxx  : xxxxxxxx : 00600000 :xxxxxxxx:00600000:0000000c:  0   :  0   :  0  :
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :0000003c:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000013:00c00000:00c00000:  xxxxxxxx  : xxxxxxxx : 00c00000 :xxxxxxxx:00c00000:0000000c:  0   :  0   :  0  :
 0 :0000003c:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000018:00000013:00c00000:00c00000:  xxxxxxxx  : xxxxxxxx : 00c00000 :xxxxxxxx:00c00000:0000000c:  0   :  0   :  0  :
```

Inputted instructions:

000000_11000_10000_01000_00000_000100

jr: 000000_11000_00000_00000_00000_001000

000000_11000_10000_01000_00000_000100

000000_11000_10001_01000_00000_000100

000000_11000_10010_01000_00000_000100

000000_11000_10011_01000_00000_000100

000000_00000_01000_01001_00000_100000 (target)

000000_11000_10000_01000_00000_000100

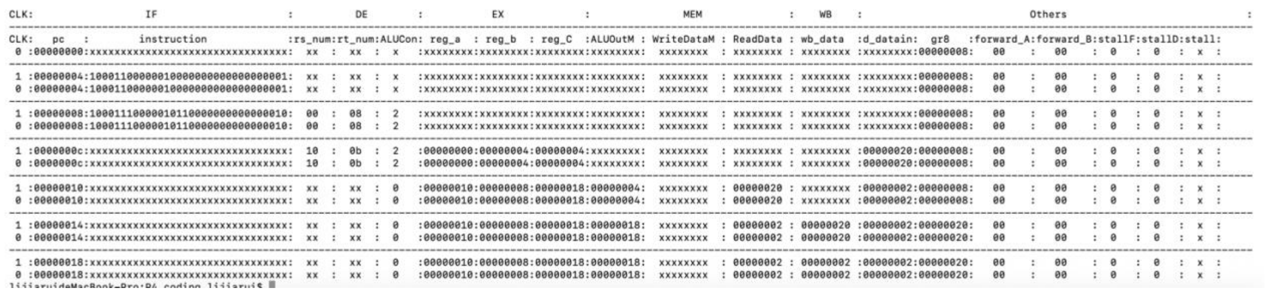
000000_11000_10001_01000_00000_000100

000000_11000_10010_01000_00000_000100

000000_11000_10011_01000_00000_000100

<code>jr</code> is similar to <code>j</code>, except for it is using the value stored in specified rs as the jump target. In the test case, `gr24 is used as the specified rs. Therefore, the pc should jump to 32’h0000_0018 (sixth line of the instructions, if considering the first line having zero index).

N. lw



```
CLK:                IF                   :          DE        :           EX            :              MEM             :   WB    :                         Others                          :
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
CLK:   pc    :         instruction            :rs_num:rt_num:ALUCon: reg_a   : reg_b   : reg_C   :ALUOutM : WriteDataM : ReadData : wb_data  :d_datain:  gr8    :forward_A:forward_B:stallF:stallD:stall:
 0 :00000000:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:   00    :   00    :  0   :  0   :  x  :
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000004:10001100000010000000000000000001:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:   00    :   00    :  0   :  0   :  x  :
 0 :00000004:10001100000010000000000000000001:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:   00    :   00    :  0   :  0   :  x  :
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000008:10001110000010110000000000000010:  00  :  08  :  2   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:   00    :   00    :  0   :  0   :  x  :
 0 :00000008:10001110000010110000000000000010:  00  :  08  :  2   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000008:   00    :   00    :  0   :  0   :  x  :
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :0000000c:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  10  :  0b  :  2   :00000000:00000004:00000004:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :00000020:00000008:   00    :   00    :  0   :  0   :  x  :
 0 :0000000c:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  10  :  0b  :  2   :00000000:00000004:00000004:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :00000020:00000008:   00    :   00    :  0   :  0   :  x  :
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000010:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000010:00000008:00000018:00000004:  xxxxxxxx  : 00000020 : xxxxxxxx :00000002:00000008:   00    :   00    :  0   :  0   :  x  :
 0 :00000010:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000010:00000008:00000018:00000004:  xxxxxxxx  : 00000020 : xxxxxxxx :00000002:00000008:   00    :   00    :  0   :  0   :  x  :
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000014:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000010:00000008:00000018:00000018:  xxxxxxxx  : 00000002 : 00000020 :00000002:00000020:   00    :   00    :  0   :  0   :  x  :
 0 :00000014:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000010:00000008:00000018:00000018:  xxxxxxxx  : 00000002 : 00000020 :00000002:00000020:   00    :   00    :  0   :  0   :  x  :
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000018:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000010:00000008:00000018:00000018:  xxxxxxxx  : 00000002 : 00000002 :00000002:00000020:   00    :   00    :  0   :  0   :  x  :
 0 :00000018:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000010:00000008:00000018:00000018:  xxxxxxxx  : 00000002 : 00000002 :00000002:00000020:   00    :   00    :  0   :  0   :  x  :
lijiaruideMacBook-Pro:P4_coding lijiarui$
```

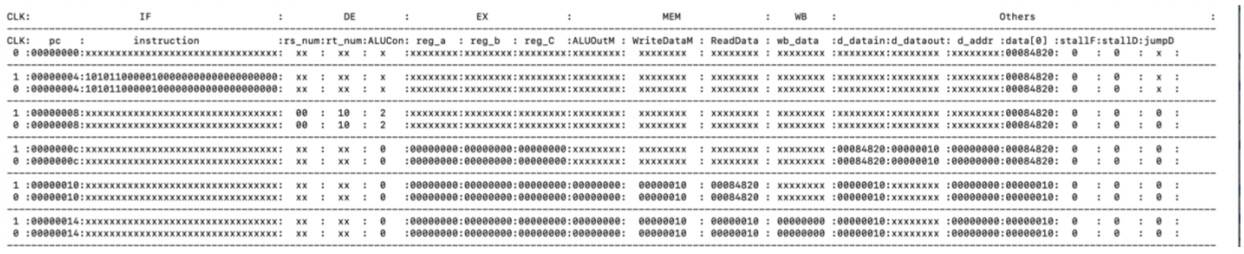
Inputted instruction:

lw: 100011_00000_01000_0000_0000_0000_0001
lw: 100011_10000_01011_0000_0000_0000_0010

Now comes the memory related instructions. About the lw instruction, in the first cycle, the first lw instruction is fetched. In the second clock, the first instruction comes into decoding stage. rs_num stands for the address value register and rt_num is the number of register to be written. With the third clock rise, reg_a and reg_b become register value passed into ALU and reg_C stores the address adding result. Note that at the end of the third clock, d_datain is the data fetched from the data memory. By doing so, in the end of coming fourth stage, we can see that ReadData holds the data value read from the memory. Therefore, in the final stage, wb_data can be selected from ALUOutM and ReadData (in lw case is ReadData).

As it is shown in the picture, in the fifth stage, wb_data is set to be 32’h0000_0020. And also shown in the picture, general register number 8, which is the rt register of this lw instruction, finally holds the value of 32’h0000_0020, the loaded data at the end of WB stage.

O. sw



```
CLK:              IF                  :         DE        :          EX           :              MEM           :   WB    :                     Others                     :
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
CLK:   pc   :        instruction       :rs_num:rt_num:ALUCon: reg_a  : reg_b  : reg_C  :ALUOutM : WriteDataM : ReadData : wb_data  :d_datain:d_dataout: d_addr :data[0] :stallF:stallD:jumpD
 0 :00000000:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:xxxxxxxx :xxxxxxxx:00084820:  0   :  0   :  x  :
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000004:10101100000100000000000000000000:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:xxxxxxxx :xxxxxxxx:00084820:  0   :  0   :  x  :
 0 :00000004:10101100000100000000000000000000:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:xxxxxxxx :xxxxxxxx:00084820:  0   :  0   :  x  :
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000008:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  00  :  10  :  2   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:xxxxxxxx :xxxxxxxx:00084820:  0   :  0   :  0  :
 0 :00000008:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  00  :  10  :  2   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:xxxxxxxx :xxxxxxxx:00084820:  0   :  0   :  0  :
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :0000000c:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000000:00000000:00000000:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :00084820:00000010 :00000000:00084820:  0   :  0   :  0  :
 0 :0000000c:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000000:00000000:00000000:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :00084820:00000010 :00000000:00084820:  0   :  0   :  0  :
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000010:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000000:00000000:00000000:00000000:  00000010  : 00084820 : xxxxxxxx :00000010:xxxxxxxx :00000000:00000010:  0   :  0   :  0  :
 0 :00000010:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000000:00000000:00000000:00000000:  00000010  : 00084820 : xxxxxxxx :00000010:xxxxxxxx :00000000:00000010:  0   :  0   :  0  :
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000014:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000000:00000000:00000000:00000000:  00000010  : 00000010 : 00000000 :00000010:xxxxxxxx :00000000:00000010:  0   :  0   :  0  :
 0 :00000014:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000000:00000000:00000000:00000000:  00000010  : 00000010 : 00000000 :00000010:xxxxxxxx :00000000:00000010:  0   :  0   :  0  :
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

Inputted instruction:

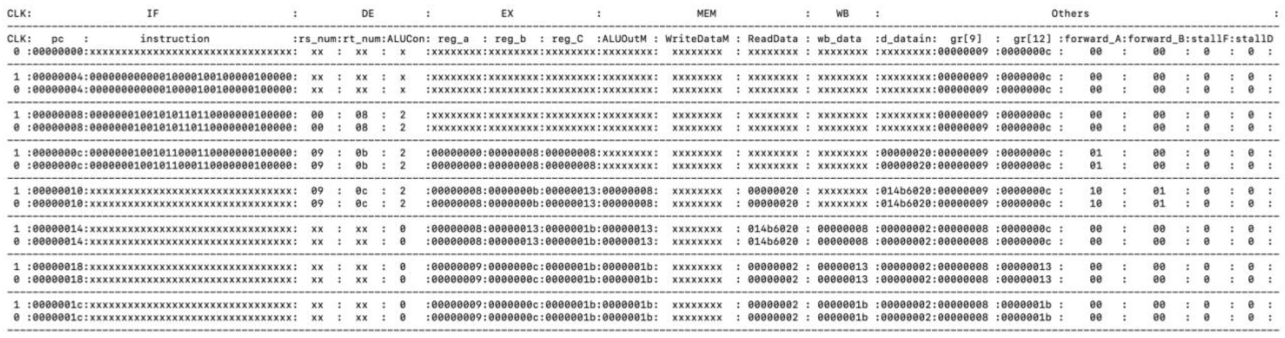
sw: 101011_00000_10000_0000_0000_0000_0000

In this instruction, the value of `gr16 (32’h0000_0010) should be written into address 32’h0000_0000 of data memory in the fourth clock rising edge. As it is displayed in the picture, data[0] changed from 32’h0008_4820 to 32’h0000_0010 in the fourth clock rise.

P. hazards

In this CPU project, I also solved the hazard problems. For control hazards, cases have already been discussed when testing branch and jump instructions. Therefore, the following will mainly focus on data hazards. As far as structural hazard is concern, it is implicitly solved since instruction memory and data memory are separated.

1. Forwarding



```
CLK:              IF                  :         DE        :          EX           :              MEM           :   WB    :                        Others                          :
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
CLK:   pc   :        instruction       :rs_num:rt_num:ALUCon: reg_a  : reg_b  : reg_C  :ALUOutM : WriteDataM : ReadData : wb_data  :d_datain:  gr[9]   :  gr[12] :forward_A:forward_B:stallF:stallD
 0 :00000000:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000009 :0000000c :   00    :   00    :  0   :  0   :
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000004:00000000000010000100100000100000:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000009 :0000000c :   00    :   00    :  0   :  0   :
 0 :00000004:00000000000010000100100000100000:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000009 :0000000c :   00    :   00    :  0   :  0   :
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000008:00000001001010110110000000100000:  00  :  08  :  2   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000009 :0000000c :   00    :   00    :  0   :  0   :
 0 :00000008:00000001001010110110000000100000:  00  :  08  :  2   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000009 :0000000c :   00    :   00    :  0   :  0   :
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :0000000c:00000001001011000110000000100000:  09  :  0b  :  2   :00000000:00000008:00000008:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :00000020:00000009 :0000000c :   01    :   00    :  0   :  0   :
 0 :0000000c:00000001001011000110000000100000:  09  :  0b  :  2   :00000000:00000008:00000008:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :00000020:00000009 :0000000c :   01    :   00    :  0   :  0   :
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000010:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  09  :  0c  :  2   :00000008:0000000b:00000013:00000008:  xxxxxxxx  : 00000020 : xxxxxxxx :014b6020:00000009 :0000000c :   10    :   01    :  0   :  0   :
 0 :00000010:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  09  :  0c  :  2   :00000008:0000000b:00000013:00000008:  xxxxxxxx  : 00000020 : xxxxxxxx :014b6020:00000009 :0000000c :   10    :   01    :  0   :  0   :
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000014:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000008:00000013:0000001b:00000013:  xxxxxxxx  : 014b6020 : 00000008 :00000002:00000008 :0000000c :   00    :   00    :  0   :  0   :
 0 :00000014:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000008:00000013:0000001b:00000013:  xxxxxxxx  : 014b6020 : 00000008 :00000002:00000008 :0000000c :   00    :   00    :  0   :  0   :
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000018:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000009:0000000c:0000001b:0000001b:  xxxxxxxx  : 00000002 : 00000013 :00000002:00000008 :00000013 :   00    :   00    :  0   :  0   :
 0 :00000018:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000009:0000000c:0000001b:0000001b:  xxxxxxxx  : 00000002 : 00000013 :00000002:00000008 :00000013 :   00    :   00    :  0   :  0   :
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :0000001c:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000009:0000000c:0000001b:0000001b:  xxxxxxxx  : 00000002 : 0000001b :00000002:00000008 :0000001b :   00    :   00    :  0   :  0   :
 0 :0000001c:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000009:0000000c:0000001b:0000001b:  xxxxxxxx  : 00000002 : 0000001b :00000002:00000008 :0000001b :   00    :   00    :  0   :  0   :
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

Inputted instructions:

add: 000000_00000_01000_01001_00000_100000

add: 000000_01001_01011_01100_00000_100000

add: 000000_01001_01100_01100_00000_100000

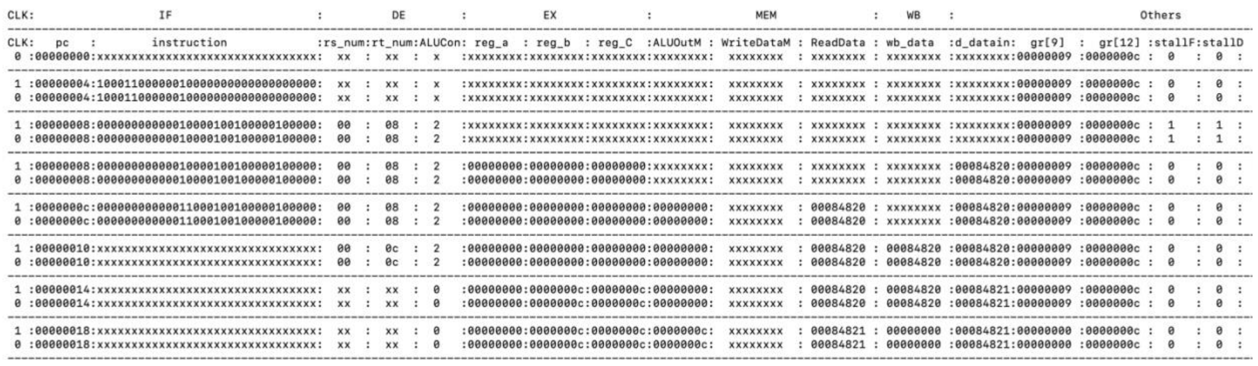
This is a short test on the two cases of forwarding, MEM forward and WB forward. The second <code>add</code> uses the destination register of the first add, meaning in the fourth overall clock cycle, a MEM forward should take place. The third instruction is referring to the destination register of both first and second instructions. Therefore, we should observe both MEM and WB forward in the fifth overall clock cycle. MEM should be forwarded to rs and WB should be forwarded to rt.

As it is shown in the picture, in the fourth clock cycle, we have reg_a equals 32’h0000_0008, which is the result of previous ALU operation. Note that this should originally be 32’h0000_0009, since the instruction is referring to `gr9 and it is originally 32’h0000_0009.

And in the fifth cycle, we have reg_a equals 32’h0000_0008 while reg_b equals 32’h0000_0013 which is the ALU result of last cycle. The former is forwarded from MEM stage and the latter is forwarded from WB stage.

For the first instruction it is written back to `gr9 at the end of fifth clock. For the second and third instructions, the result is written back to `gr12 at sixth and seventh clock sequentially, as shown in the picture.

2. Stalling



```
CLK:              IF                    :        DE        :          EX           :           MEM            :   WB    :                        Others
----------------------------------------------------------------------------------------------------------------------------------------------------------------
CLK:   pc    :        instruction          :rs_num:rt_num:ALUCon: reg_a  : reg_b  : reg_C  :ALUOutM : WriteDataM : ReadData : wb_data  :d_datain:  gr[9]  :  gr[12] :stallF:stallD
 0 :00000000:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000009 :0000000c :  0   :  0  :
----------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000004:10001100000010000000000000000000:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000009 :0000000c :  0   :  0  :
 0 :00000004:10001100000010000000000000000000:  xx  :  xx  :  x   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000009 :0000000c :  0   :  0  :
----------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000008:00000000000010000100100000100000:  00  :  08  :  2   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000009 :0000000c :  1   :  1  :
 0 :00000008:00000000000010000100100000100000:  00  :  08  :  2   :xxxxxxxx:xxxxxxxx:xxxxxxxx:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :xxxxxxxx:00000009 :0000000c :  1   :  1  :
----------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000008:00000000000010000100100000100000:  00  :  08  :  2   :00000000:00000000:00000000:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :00084820:00000009 :0000000c :  0   :  0  :
 0 :00000008:00000000000010000100100000100000:  00  :  08  :  2   :00000000:00000000:00000000:xxxxxxxx:  xxxxxxxx  : xxxxxxxx : xxxxxxxx :00084820:00000009 :0000000c :  0   :  0  :
----------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :0000000c:00000000000011000100100000100000:  00  :  08  :  2   :00000000:00000000:00000000:00000000:  xxxxxxxx  : 00084820 : xxxxxxxx :00084820:00000009 :0000000c :  0   :  0  :
 0 :0000000c:00000000000011000100100000100000:  00  :  08  :  2   :00000000:00000000:00000000:00000000:  xxxxxxxx  : 00084820 : xxxxxxxx :00084820:00000009 :0000000c :  0   :  0  :
----------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000010:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  00  :  0c  :  2   :00000000:00000000:00000000:00000000:  xxxxxxxx  : 00084820 : 00084820 :00084820:00000009 :0000000c :  0   :  0  :
 0 :00000010:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  00  :  0c  :  2   :00000000:00000000:00000000:00000000:  xxxxxxxx  : 00084820 : 00084820 :00084820:00000009 :0000000c :  0   :  0  :
----------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000014:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000000:0000000c:0000000c:00000000:  xxxxxxxx  : 00084820 : 00084820 :00084821:00000009 :0000000c :  0   :  0  :
 0 :00000014:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000000:0000000c:0000000c:00000000:  xxxxxxxx  : 00084820 : 00084820 :00084821:00000009 :0000000c :  0   :  0  :
----------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 :00000018:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000000:0000000c:0000000c:0000000c:  xxxxxxxx  : 00084821 : 00000000 :00084821:00000000 :0000000c :  0   :  0  :
 0 :00000018:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:  xx  :  xx  :  0   :00000000:0000000c:0000000c:0000000c:  xxxxxxxx  : 00084821 : 00000000 :00084821:00000000 :0000000c :  0   :  0  :
----------------------------------------------------------------------------------------------------------------------------------------------------------------
```

Inputted instruction:

100011_00000_01000_00000_00000_000000

000000_00000_01000_01001_00000_100000

000000_00000_01100_01001_00000_100000

Stall happens when there is an lw instruction followed by an intructions trying to use the register which is going to hold the loaded data. In this case, the second instruction is trying to use the `gr8, which is the destination register of first lw instruction. Therefore, in the second clock, a stall occurred, holding the pc and fetched instruction the same for one cycle.

3. Flashing (please refer to previous discuss on branch and jump instructions)

V. Reflections