Project Report

CSC 3050 Simplified Pipelined MIPS Microprocessor

Wei Wu

118010335

April 30, 2020

The School of Science and Engineering

香 港 中 文 大 學（深 圳）
The Chinese University of Hong Kong, Shenzhen

1. Understandings of the Project

This project is about writing a simplified pipelined MIPS microprocessor using hardware description language Verilog. Generally speaking, the CPU works with the general registers, instruction memory, and data memory.

The pipeline contains 5 stages. The first stage is Instruction Fetch (IF), where one piece of instruction is fetched from the instruction memory according to PC. The second stage is Instruction Decode (ID). The control unit generates control signals for both ALU and data path from the opcode (and function for R-type instructions). Meanwhile, the general registers are read. The third stage is Execution (EX). According to the control signals, the ALU performs certain calculation. The fourth stage is Memory (MEM), where the data memory is either read or written. The fifth stage is Write Back (WB), where the result is written back to the general registers.

Furthermore, the pipeline hazards are handled using forwarding, stalling, and flushing.

2. The Implementations of the Project

2.1. The Big Picture Idea

First of all, we should build the instruction memory, data memory, and general registers. Then, we can build the 5 pipeline stages one by one. Before the Instruction Fetch stage, there should be a multiplexer for updating PC. In the Instruction Fetch stage, the instruction memory at PC is accessed. In the Instruction Decode stage, the opcode and function (for R-type instructions) are sent to the control unit. At the same time, according to rs and rt, two general registers are read. In the Execution Stage, the ALU takes as input two operands. One is rs, and the other is rt or the sign-extended immediate. Besides, the ALU is controlled by the ALU control signal. In the Memory stage, there are two cases. The data memory is either read or written. In both cases, the ALU result is the memory address to access. In the Write Back stage, either the ALU result or the data from memory is written back to the general registers. After finishing all the stages, we need to add pipeline registers between each two stages.

There are three types of pipeline hazards, namely structural hazards, data hazards, and control hazards. The structural hazards are solved by using separate instruction memory and data memory. The data hazards can be tackled by forwarding and stalling. While the control hazards can be handled by flushing.

2.2. Implementation Details

2.2.0. The ALU

The ALU is based on project 3. Since instructions like MULT and DIV are no longer required, the register hi and lo are removed.

2.2.1. The control part

The control part (module control_unit in the code) takes both opcode and funct as input. It generates control signals for both the ALU and the data path. The signals are RegWrite, MemtoReg, MemWrite, Branch, ALUControl, ALUSrc, RegDst, and Jump. Their functions are listed below.

Signal	**Length (bit)**	**Function**
RegWrite	1	Determines whether to write the result back to the general registers
MemtoReg	1	Determines whether to write the data from memory or the ALU result back to the general registers
MemWrite	1	Determines whether to write the data memory
Branch	2	Indicates whether the current instruction is a branch instruction
ALUControl	5	Determines the behavior of the ALU
ALUSrc	1	Determines the source of the second ALU operand
RegDst	1	Determines whether to write back to rt or rd
Jump	2	Indicates whether the current instruction is a jump instruction

2.2.2. The data flow

In the CPU, components such as the ALU and the control unit are combinational logics. In other words, the output will be updated once the input changes. While the whole pipeline is a sequential logic. The pipeline registers update only at positive clock edges.

2.2.3. The memory

The memory is divided into instruction memory and data memory. Since in the MIPS architecture, the memory is byte addressable, each byte of memory is implemented with reg[7:0]. Besides, we made both instruction memory and data memory 1KB large.

2.2.4. The general registers

In the MIPS architecture, there are 32 general registers. Each general register is 32-bit and therefore implemented with reg[31:0]. In particular, $zero is always equal to 0.

2.2.5. Branch instructions

In this project, there are two types of branch instructions, namely BEQ and BNE. Thus, a 2-bit Branch control signal is used. To be specific, 00 stands for no branch, 01 stands for BEQ, and 10 stands for BNE.

In the Memory stage, if (BranchM=01 and ZeroM=1) or (BranchM=10 and ZeroM=0), the branch will be taken. On the one hand, the signal PCSrcM will be 1. Thus, at the next positive clock edge, the PC will be updated to PCBranchM. On the other hand, the IF/ID and ID/EX pipeline registers will be flushed. Besides, at the next positive clock edge, the InstrD in IF/ID will be 32’h0000_0000 (NOP).

2.2.6. Jump instructions

In this project, there are three types of jump instructions, namely J, JAL, and JR. Thus, a 2-bit Jump control signal is used. To be specific, 00 stands for no jump, 01 stands for J, 10 stands for JAL, and 11 stands for JR.

In the Execution stage, if JumpE != 00, the jump will be executed. For J, PC changes to {PCPlus4F[31:28], TargetE} at the next positive clock edge. For JAL, in addition to J, the current PC will be stored in the register $ra. For JR, PC changes to reg_A.

After that, the IF/ID pipeline register should be flushed. Similar to branch, at the next positive clock edge, the InstrD in IF/ID will be 32’h0000_0000 (NOP).

2.2.7. Pipeline Hazards

There exist three types of hazards: structural hazards, data hazards, and control hazards. Structural hazards are the conflict in the use of a resource. These hazards can be easily solved using separate instruction memory and data memory.

Data hazards happen when an instruction depends on the completion of data access by a previous instruction. In the MIPS architecture, only RAW (read after write) is possible. In most cases, data hazards can be tackled by forwarding. We can bypass the data in the MEM or WB stage to the EX stage. The logic of forwarding is as follows:

```
if (EX/MEM.RegWrite and (EX/MEM.RegRd != 0)
  and (EX/MEM.RegRd = ID/EX.RegRs))
        ForwardA = 10
if (MEM/WB.RegWrite and (MEM/WB.RegRd != 0)
  and (EX/MEM.RegRd != ID/EX.Reg.Rs)
  and (MEM/WB.RegRd = ID/EX.RegRs))
        ForwardA = 01


if (EX/MEM.RegWrite and (EX/MEM.RegRd != 0)
  and (EX/MEM.RegRd = ID/EX.RegRt))
        ForwardB = 10
if (MEM/WB.RegWrite and (MEM/WB.RegRd != 0)
  and (EX/MEM.RegRd != ID/EX.Reg.Rt)
  and (MEM/WB.RegRd = ID/EX.RegRt))
        ForwardB = 01
```

However, only forwarding cannot handle all the data hazards. When a lw is followed by an instruction to read the loaded register, the pipeline must be stalled for one clock cycle. The logic of stalling is as follows:

```
if (ID/EX.MemRead and ((ID/EX.RegisterRt = IF/ID.RegisterRs)
  or (ID/EX.RegisterRt = IF/ID.registerRt))
        LoadStall = 1
```

When a stall occurs, both PC and IF/ID should not be changed. Furthermore, a NOP should be inserted to the EX stage by changing the control signals RegWrite, MemtoReg, MemWrite, Branch, and Jump in the ID/EX pipeline register to 0.

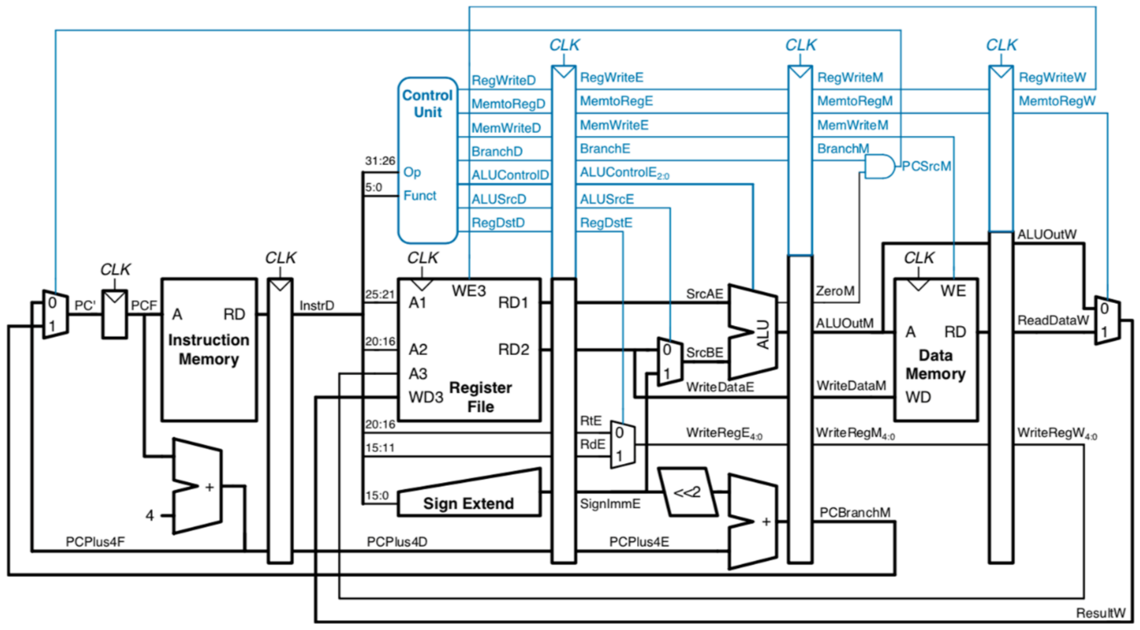
Control hazards occur when the decision on control action depends on a previous instruction. More specifically, a branch taken or jump. They could be dealt with by flushing the pipeline.

Whenever a branch is taken, on the one hand, the signal PCSrcM will be 1. Thus, at the next positive clock edge, the PC will be updated to PCBranchM. On the other hand, the IF/ID and ID/EX pipeline registers will be flushed. Besides, at the next positive clock edge, the InstrD in IF/ID will be 32’h0000_0000 (NOP).

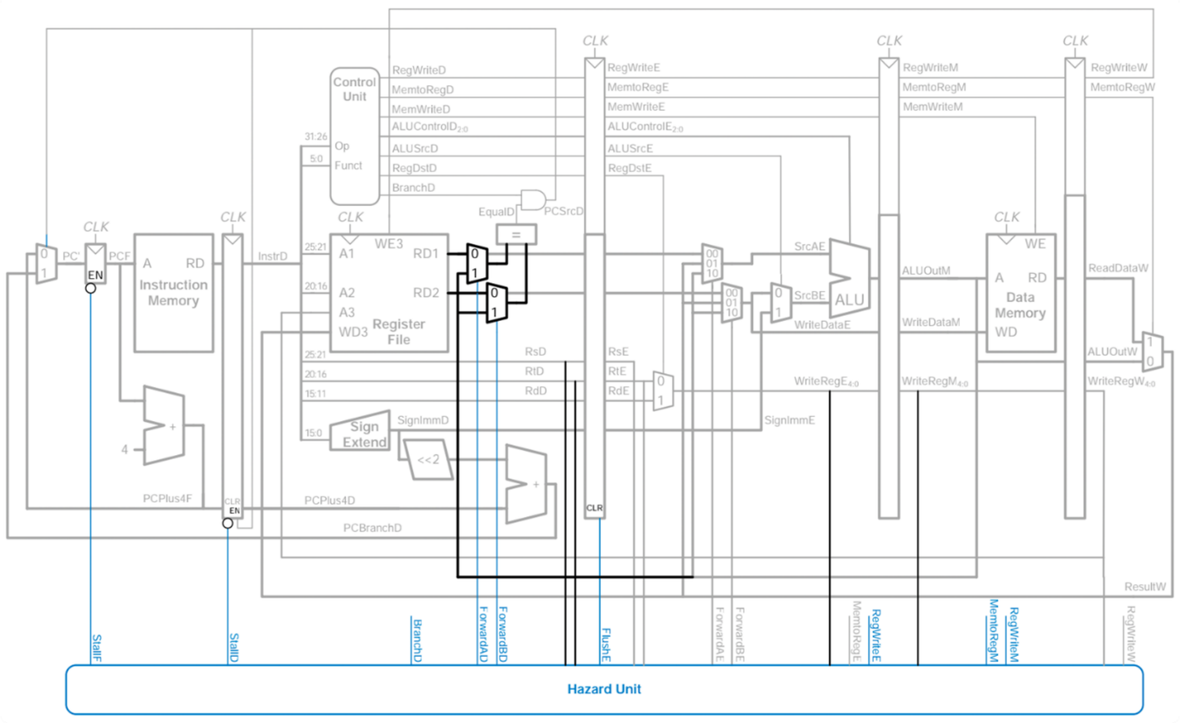
Whenever a jump occurs, PC will change to {PCPlus4F[31:28], TargetE} (for J and JAL) or reg_A (for JR) at the next positive clock edge. After that, the IF/ID pipeline register will be flushed. Similar to branch, at the next positive clock edge, the InstrD in IF/ID will be 32’h0000_0000 (NOP).

2.3. Block Diagram

2.3.1. Pipelined Processor with Control



2.3.2. Pipelined Processor with Full Hazard Handling



Note:

My program mostly follows the above block diagrams. However, some of the control signals are slightly modified in order to make the code clearer.

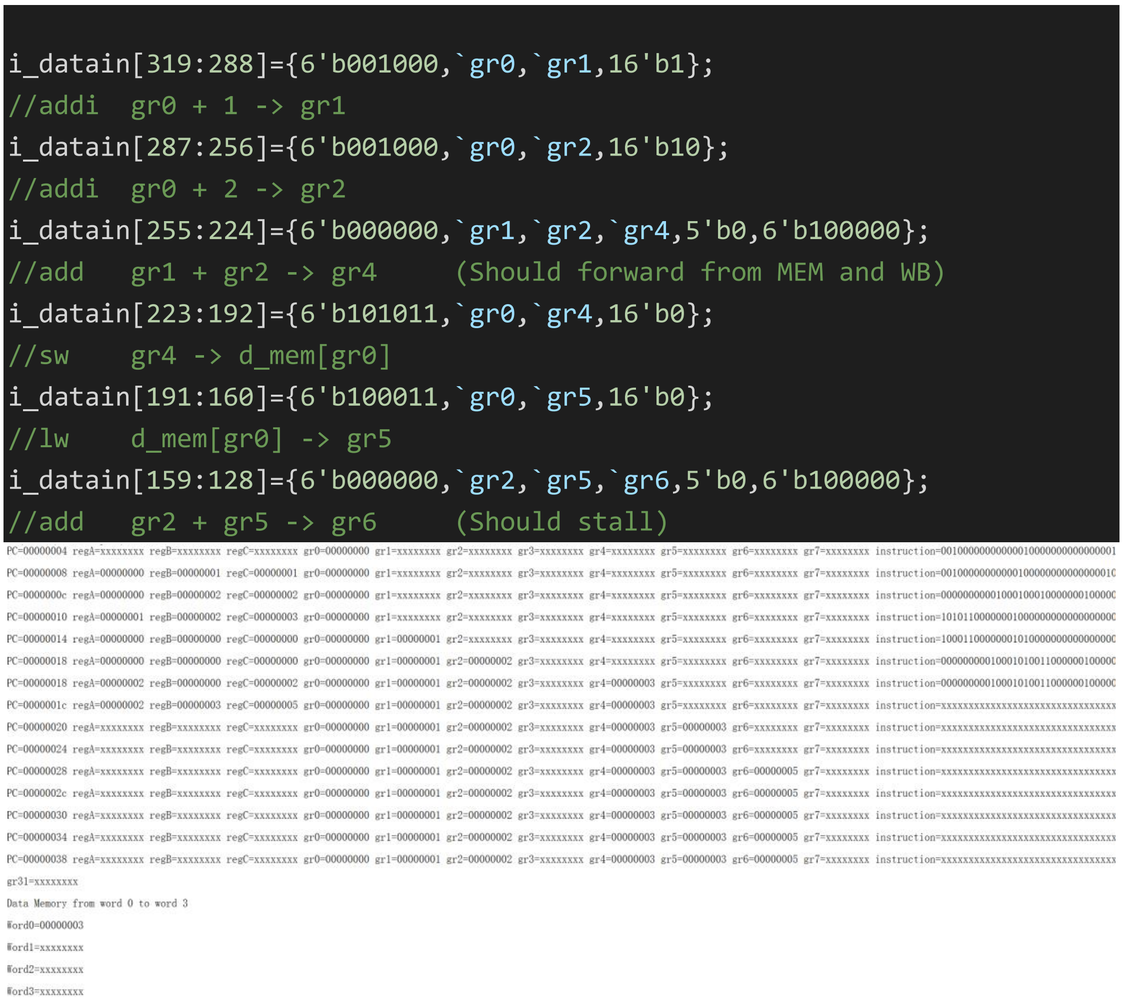
2.4. Explanation of Instructions

2.4.0. Testing Environment

Windows 10 x64

iVerilog v11-20190809 x64

2.4.1. Test for lw, sw, Forwarding & Stalling

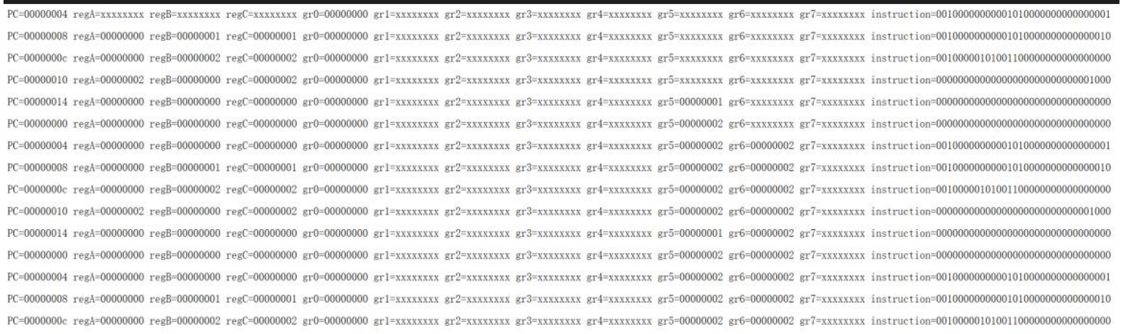


```
i_datain[319:288]={6'b001000,`gr0,`gr1,16'b1};
//addi  gr0 + 1 -> gr1
i_datain[287:256]={6'b001000,`gr0,`gr2,16'b10};
//addi  gr0 + 2 -> gr2
i_datain[255:224]={6'b000000,`gr1,`gr2,`gr4,5'b0,6'b100000};
//add   gr1 + gr2 -> gr4     (Should forward from MEM and WB)
i_datain[223:192]={6'b101011,`gr0,`gr4,16'b0};
//sw    gr4 -> d_mem[gr0]
i_datain[191:160]={6'b100011,`gr0,`gr5,16'b0};
//lw    d_mem[gr0] -> gr5
i_datain[159:128]={6'b000000,`gr2,`gr5,`gr6,5'b0,6'b100000};
//add   gr2 + gr5 -> gr6     (Should stall)
```

```
PC=00000004 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=xxxxxxxx gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00100000000000010000000000000001
PC=00000008 regA=00000000 regB=00000001 regC=00000001 gr0=00000000 gr1=xxxxxxxx gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=0010000000000010000000000000001
PC=0000000c regA=00000000 regB=00000002 regC=00000002 gr0=00000000 gr1=xxxxxxxx gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=0000000001000100010000000100000
PC=00000010 regA=00000001 regB=00000002 regC=00000003 gr0=00000000 gr1=xxxxxxxx gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=1010110000000100000000000000000
PC=00000014 regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=00000001 gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=1000110000000101000000000000000
PC=00000018 regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=00000001 gr2=00000002 gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=0000000001000101001100000010000
PC=00000018 regA=00000002 regB=00000000 regC=00000002 gr0=00000000 gr1=00000001 gr2=00000002 gr3=xxxxxxxx gr4=00000003 gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=0000000001000101001100000010000
PC=0000001c regA=00000002 regB=00000003 regC=00000005 gr0=00000000 gr1=00000001 gr2=00000002 gr3=xxxxxxxx gr4=00000003 gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000020 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=00000001 gr2=00000002 gr3=xxxxxxxx gr4=00000003 gr5=00000003 gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000024 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=00000001 gr2=00000002 gr3=xxxxxxxx gr4=00000003 gr5=00000003 gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000028 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=00000001 gr2=00000002 gr3=xxxxxxxx gr4=00000003 gr5=00000003 gr6=00000005 gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=0000002c regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=00000001 gr2=00000002 gr3=xxxxxxxx gr4=00000003 gr5=00000003 gr6=00000005 gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000030 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=00000001 gr2=00000002 gr3=xxxxxxxx gr4=00000003 gr5=00000003 gr6=00000005 gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000034 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=00000001 gr2=00000002 gr3=xxxxxxxx gr4=00000003 gr5=00000003 gr6=00000005 gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000038 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=00000001 gr2=00000002 gr3=xxxxxxxx gr4=00000003 gr5=00000003 gr6=00000005 gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
gr31=xxxxxxxx
Data Memory from word 0 to word 3
Word0=00000003
Word1=xxxxxxxx
Word2=xxxxxxxx
Word3=xxxxxxxx
```

2.4.2. Test for Forwarding, jr

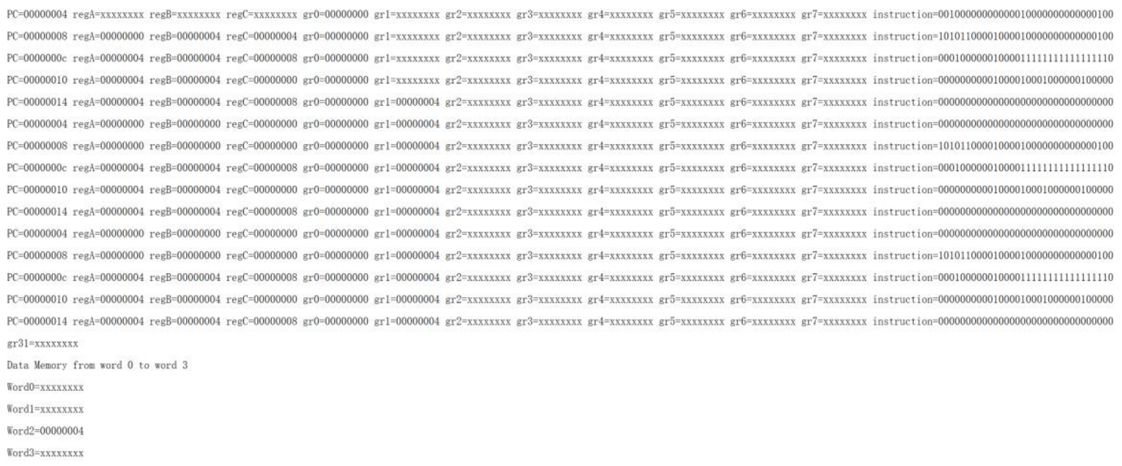
```
i_datain[319:288]={6'b001000,`gr0,`gr5,16'b1};
//addi  gr0 + 1 -> gr5
i_datain[287:256]={6'b001000,`gr0,`gr5,16'b10};
//addi  gr0 + 2 -> gr5
i_datain[255:224]={6'b001000,`gr5,`gr6,16'b0};
//addi  gr5 + 0 -> gr6      (Should forward from MEM, rather than WB)
i_datain[223:192]={6'b000000,`gr0,`gr0,`gr0,5'b0,6'b001000};
//jr    gr0                 (Should jump to PC=0)
i_datain[191:160]={6'b001000,`gr0,`gr1,16'b1};
//addi  gr0 + 1 -> gr1      (Should never be executed)
```



```
PC=00000004 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=xxxxxxxx gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00100000000001010000000000000001
PC=00000008 regA=00000000 regB=00000001 regC=00000001 gr0=00000000 gr1=xxxxxxxx gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00100000000001010000000000000010
PC=0000000c regA=00000000 regB=00000002 regC=00000002 gr0=00000000 gr1=xxxxxxxx gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00100000101001100000000000000000
PC=00000010 regA=00000002 regB=00000000 regC=00000002 gr0=00000000 gr1=xxxxxxxx gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000000000000000000000001000
PC=00000014 regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=xxxxxxxx gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=00000001 gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000000000000000000000000000
PC=00000000 regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=xxxxxxxx gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=00000002 gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000000000000000000000000000
PC=00000004 regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=xxxxxxxx gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=00000002 gr6=00000002 gr7=xxxxxxxx instruction=00100000000001010000000000000001
PC=00000008 regA=00000000 regB=00000001 regC=00000001 gr0=00000000 gr1=xxxxxxxx gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=00000002 gr6=00000002 gr7=xxxxxxxx instruction=00100000000001010000000000000010
PC=0000000c regA=00000000 regB=00000002 regC=00000002 gr0=00000000 gr1=xxxxxxxx gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=00000002 gr6=00000002 gr7=xxxxxxxx instruction=00100000101001100000000000000000
PC=00000010 regA=00000002 regB=00000000 regC=00000002 gr0=00000000 gr1=xxxxxxxx gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=00000002 gr6=00000002 gr7=xxxxxxxx instruction=00000000000000000000000000001000
PC=00000014 regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=xxxxxxxx gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=00000001 gr6=00000002 gr7=xxxxxxxx instruction=00000000000000000000000000000000
PC=00000000 regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=xxxxxxxx gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=00000002 gr6=00000002 gr7=xxxxxxxx instruction=00000000000000000000000000000000
PC=00000004 regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=xxxxxxxx gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=00000002 gr6=00000002 gr7=xxxxxxxx instruction=00100000000001010000000000000001
PC=00000008 regA=00000000 regB=00000001 regC=00000001 gr0=00000000 gr1=xxxxxxxx gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=00000002 gr6=00000002 gr7=xxxxxxxx instruction=00100000000001010000000000000010
PC=0000000c regA=00000000 regB=00000002 regC=00000002 gr0=00000000 gr1=xxxxxxxx gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=00000002 gr6=00000002 gr7=xxxxxxxx instruction=00100000101001100000000000000000
```

2.4.3. Test for Branch Flush

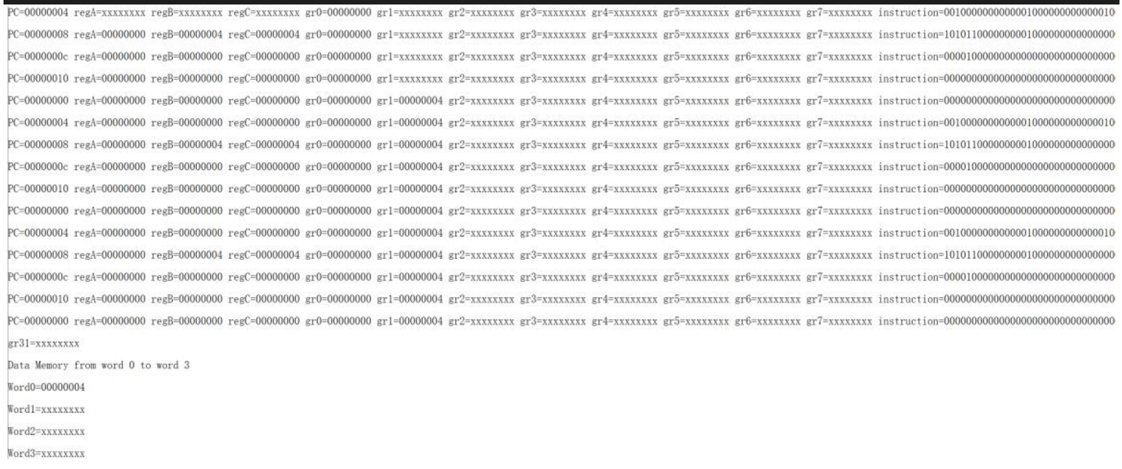
```
i_datain[319:288]={6'b001000,`gr0,`gr1,16'b100};
//addi  gr0 + 4 -> gr1
i_datain[287:256]={6'b101011,`gr1,`gr1,16'b100};
//sw    gr1 -> d_mem[gr1+4]
i_datain[255:224]={6'b000100,`gr1,`gr1,16'b1111_1111_1111_1110};
//beq   gr1, gr1 -> -2      (Should branch to PC=4)
i_datain[223:192]={6'b000000,`gr1,`gr1,`gr2,5'b0,6'b100000};
//add   gr1 + gr1 -> gr2    (Should never be executed)
i_datain[191:160]={6'b000000,`gr0,`gr1,`gr3,5'b0,6'b100000};
//add   gr0 + gr1 -> gr3    (Should never be executed)
```



```
PC=00000004 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=xxxxxxxx gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00100000000000010000000000000100
PC=00000008 regA=00000000 regB=00000004 regC=00000004 gr0=00000000 gr1=xxxxxxxx gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=10101100001000010000000000000100
PC=0000000c regA=00000004 regB=00000004 regC=00000008 gr0=00000000 gr1=xxxxxxxx gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00010000001000011111111111111110
PC=00000010 regA=00000004 regB=00000004 regC=00000000 gr0=00000000 gr1=xxxxxxxx gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000001000010001000000100000
PC=00000014 regA=00000004 regB=00000004 regC=00000008 gr0=00000000 gr1=00000004 gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000000000000000000000000000
PC=00000004 regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=00000004 gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000000000000000000000000000
PC=00000008 regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=00000004 gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=10101100001000010000000000000100
PC=0000000c regA=00000004 regB=00000004 regC=00000008 gr0=00000000 gr1=00000004 gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00010000001000011111111111111110
PC=00000010 regA=00000004 regB=00000004 regC=00000000 gr0=00000000 gr1=00000004 gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000001000010001000000100000
PC=00000014 regA=00000004 regB=00000004 regC=00000008 gr0=00000000 gr1=00000004 gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000000000000000000000000000
PC=00000004 regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=00000004 gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000000000000000000000000000
PC=00000008 regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=00000004 gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=10101100001000010000000000000100
PC=0000000c regA=00000004 regB=00000004 regC=00000008 gr0=00000000 gr1=00000004 gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00010000001000011111111111111110
PC=00000010 regA=00000004 regB=00000004 regC=00000000 gr0=00000000 gr1=00000004 gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000001000010001000000100000
PC=00000014 regA=00000004 regB=00000004 regC=00000008 gr0=00000000 gr1=00000004 gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000000000000000000000000000
gr31=xxxxxxxx
Data Memory from word 0 to word 3
Word0=xxxxxxxx
Word1=xxxxxxxx
Word2=00000004
Word3=xxxxxxxx
```

2.4.4. Test for Jump Flush

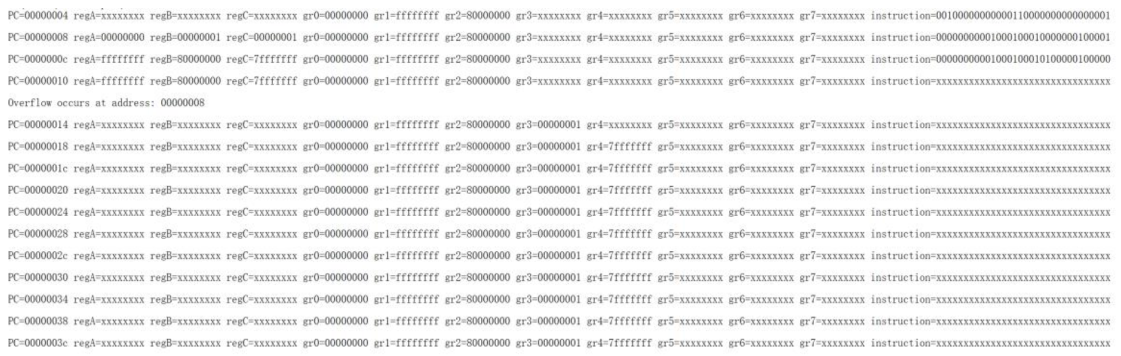
```
i_datain[319:288]={6'b001000,`gr0,`gr1,16'b100};
//addi  gr0 + 4 -> gr1
i_datain[287:256]={6'b101011,`gr0,`gr1,16'b0};
//sw    gr1 -> d_mem[gr0]
i_datain[255:224]={6'b000010,26'b0};
//j     0                   (Should jump to PC=0)
i_datain[223:192]={6'b000000,`gr1,`gr1,`gr2,5'b0,6'b100000};
//add   gr1 + gr1 -> gr2    (Should never be executed)
i_datain[191:160]={6'b000000,`gr0,`gr1,`gr3,5'b0,6'b100000};
//add   gr0 + gr1 -> gr3    (Should never be executed)
```



```
PC=00000004 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=xxxxxxxx gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=0010000000000001000000000000010
PC=00000008 regA=00000000 regB=00000004 regC=00000004 gr0=00000000 gr1=xxxxxxxx gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=1010110000000001000000000000000
PC=0000000c regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=xxxxxxxx gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=0000100000000000000000000000000
PC=00000010 regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=xxxxxxxx gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=0000000000000000000000000000000
PC=00000000 regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=00000004 gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=0000000000000000000000000000000
PC=00000004 regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=00000004 gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=0010000000000001000000000000010
PC=00000008 regA=00000000 regB=00000004 regC=00000004 gr0=00000000 gr1=00000004 gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=1010110000000001000000000000000
PC=0000000c regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=00000004 gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=0000100000000000000000000000000
PC=00000010 regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=00000004 gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=0000000000000000000000000000000
PC=00000000 regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=00000004 gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=0000000000000000000000000000000
PC=00000004 regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=00000004 gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=0010000000000001000000000000010
PC=00000008 regA=00000000 regB=00000004 regC=00000004 gr0=00000000 gr1=00000004 gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=1010110000000001000000000000000
PC=0000000c regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=00000004 gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=0000100000000000000000000000000
PC=00000010 regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=00000004 gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=0000000000000000000000000000000
PC=00000000 regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=00000004 gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=0000000000000000000000000000000
gr31=xxxxxxxx
Data Memory from word 0 to word 3
Word0=00000004
Word1=xxxxxxxx
Word2=xxxxxxxx
Word3=xxxxxxxx
```

2.4.5. Test for Overflow

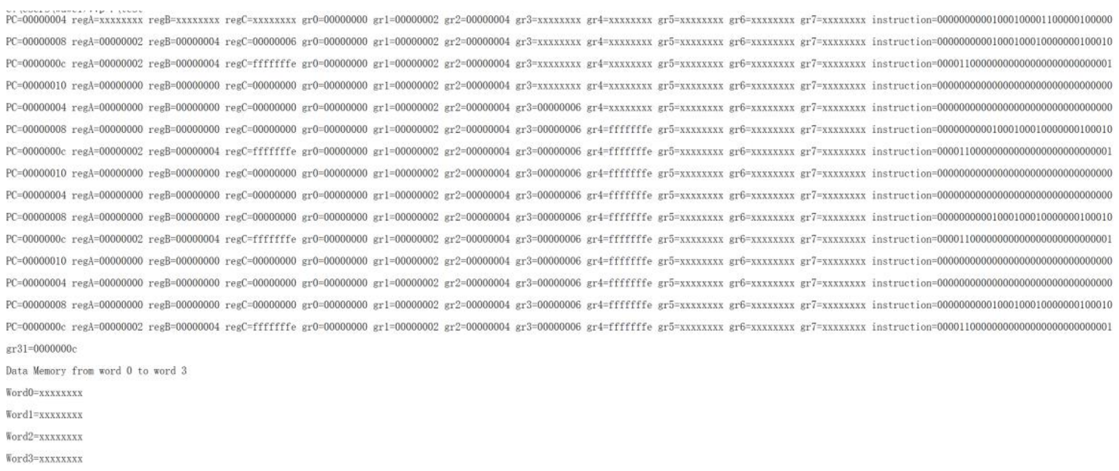
```
a. gr[1]=32'b1111_1111_1111_1111_1111_1111_1111_1111;
//gr1 = -1
a. gr[2]=32'b1000_0000_0000_0000_0000_0000_0000_0000;
//gr2 = -2147483648
i_datain[319:288]={6'b001000,`gr0,`gr3,16'b1};
//addi  gr0 + 1 -> gr3
i_datain[287:256]={6'b000000,`gr1,`gr2,`gr4,5'b0,6'b100001};
//addu  gr1 + gr2 -> gr4 (no overflow)
i_datain[255:224]={6'b000000,`gr1,`gr2,`gr5,5'b0,6'b100000};
//add   gr1 + gr2 -> gr5 (overflow)
```



```
PC=00000004 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=ffffffff gr2=80000000 gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00100000000000110000000000000001
PC=00000008 regA=00000000 regB=00000001 regC=00000001 gr0=00000000 gr1=ffffffff gr2=80000000 gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000001000100010000000100001
PC=0000000c regA=ffffffff regB=80000000 regC=7fffffff gr0=00000000 gr1=ffffffff gr2=80000000 gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000001000100010100000100000
PC=00000010 regA=ffffffff regB=80000000 regC=7fffffff gr0=00000000 gr1=ffffffff gr2=80000000 gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Overflow occurs at address: 00000008
PC=00000014 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=ffffffff gr2=80000000 gr3=00000001 gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000018 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=ffffffff gr2=80000000 gr3=00000001 gr4=7fffffff gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=0000001c regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=ffffffff gr2=80000000 gr3=00000001 gr4=7fffffff gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000020 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=ffffffff gr2=80000000 gr3=00000001 gr4=7fffffff gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000024 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=ffffffff gr2=80000000 gr3=00000001 gr4=7fffffff gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000028 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=ffffffff gr2=80000000 gr3=00000001 gr4=7fffffff gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=0000002c regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=ffffffff gr2=80000000 gr3=00000001 gr4=7fffffff gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000030 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=ffffffff gr2=80000000 gr3=00000001 gr4=7fffffff gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000034 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=ffffffff gr2=80000000 gr3=00000001 gr4=7fffffff gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000038 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=ffffffff gr2=80000000 gr3=00000001 gr4=7fffffff gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=0000003c regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=ffffffff gr2=80000000 gr3=00000001 gr4=7fffffff gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

2.4.6. Test for add, sub, jal

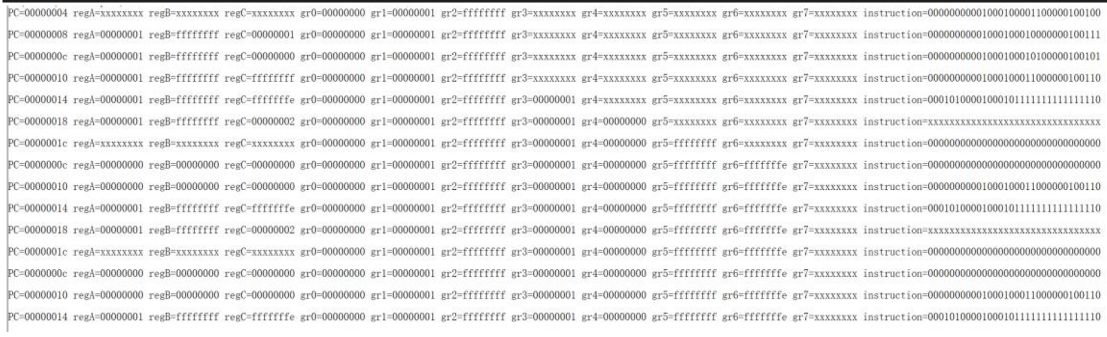
```
a. gr[1]=32'b0000_0000_0000_0000_0000_0000_0000_0010;
//gr1 = 2
a. gr[2]=32'b0000_0000_0000_0000_0000_0000_0000_0100;
//gr2 = 4
i_datain[319:288]={6'b000000,`gr1,`gr2,`gr3,5'b0,6'b100000};
//add   gr1 + gr2 -> gr3
i_datain[287:256]={6'b000000,`gr1,`gr2,`gr4,5'b0,6'b100010};
//sub   gr1 - gr2 -> gr4
i_datain[255:224]={6'b000011,26'b1};
//jal   1               (Should link PC+4 and jump to PC=4)
```



```
PC=00000004 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=00000002 gr2=00000004 gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000001000100001100000100000
PC=00000008 regA=00000002 regB=00000004 regC=00000006 gr0=00000000 gr1=00000002 gr2=00000004 gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000001000100010000000100010
PC=0000000c regA=00000002 regB=00000004 regC=fffffffe gr0=00000000 gr1=00000002 gr2=00000004 gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00011000000000000000000000000001
PC=00000010 regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=00000002 gr2=00000004 gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000000000000000000000000000
PC=00000004 regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=00000002 gr2=00000004 gr3=00000006 gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000000000000000000000000000
PC=00000008 regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=00000002 gr2=00000004 gr3=00000006 gr4=fffffffe gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000001000100010000000100010
PC=0000000c regA=00000002 regB=00000004 regC=fffffffe gr0=00000000 gr1=00000002 gr2=00000004 gr3=00000006 gr4=fffffffe gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00011000000000000000000000000001
PC=00000010 regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=00000002 gr2=00000004 gr3=00000006 gr4=fffffffe gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000000000000000000000000000
PC=00000004 regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=00000002 gr2=00000004 gr3=00000006 gr4=fffffffe gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000000000000000000000000000
PC=00000008 regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=00000002 gr2=00000004 gr3=00000006 gr4=fffffffe gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000001000100010000000100010
PC=0000000c regA=00000002 regB=00000004 regC=fffffffe gr0=00000000 gr1=00000002 gr2=00000004 gr3=00000006 gr4=fffffffe gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00011000000000000000000000000001
PC=00000010 regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=00000002 gr2=00000004 gr3=00000006 gr4=fffffffe gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000000000000000000000000000
PC=00000004 regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=00000002 gr2=00000004 gr3=00000006 gr4=fffffffe gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000000000000000000000000000
PC=00000008 regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=00000002 gr2=00000004 gr3=00000006 gr4=fffffffe gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000001000100010000000100010
PC=0000000c regA=00000002 regB=00000004 regC=fffffffe gr0=00000000 gr1=00000002 gr2=00000004 gr3=00000006 gr4=fffffffe gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00011000000000000000000000000001
gr31=0000000c
Data Memory from word 0 to word 3
Word0=xxxxxxxx
Word1=xxxxxxxx
Word2=xxxxxxxx
Word3=xxxxxxxx
```

2.4.7. Test for and, nor, or, xor, bne

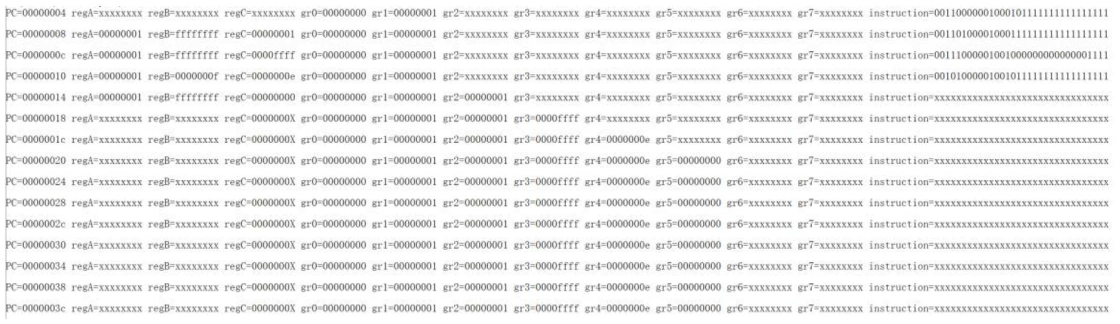
```
a. gr[1]=32'b0000_0000_0000_0000_0000_0000_0000_0001;
//gr1 = 1
a. gr[2]=32'b1111_1111_1111_1111_1111_1111_1111_1111;
//gr2 = 2**32-1
i_datain[319:288]={6'b000000,`gr1,`gr2,`gr3,5'b0,6'b100100};
//and   gr1 & gr2 -> gr3
i_datain[287:256]={6'b000000,`gr1,`gr2,`gr4,5'b0,6'b100111};
//nor   gr1 nor gr2 -> gr4
i_datain[255:224]={6'b000000,`gr1,`gr2,`gr5,5'b0,6'b100101};
//or    gr1 | gr2 -> gr5
i_datain[223:192]={6'b000000,`gr1,`gr2,`gr6,5'b0,6'b100110};
//xor   gr1 | gr2 -> gr6
i_datain[191:160]={6'b000101,`gr1,`gr2,16'b1111_1111_1111_1110};
//bne   gr1, gr1 -> -2       (Should branch to PC=c)
```



```
PC=00000004 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=00000001 gr2=ffffffff gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000001000100001100000100100
PC=00000008 regA=00000001 regB=ffffffff regC=00000001 gr0=00000000 gr1=00000001 gr2=ffffffff gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000001000100010000000100111
PC=0000000c regA=00000001 regB=ffffffff regC=00000000 gr0=00000000 gr1=00000001 gr2=ffffffff gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000001000100010100000100101
PC=00000010 regA=00000001 regB=ffffffff regC=ffffffff gr0=00000000 gr1=00000001 gr2=ffffffff gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000001000100011000000100110
PC=00000014 regA=00000001 regB=ffffffff regC=fffffffe gr0=00000000 gr1=00000001 gr2=ffffffff gr3=00000001 gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00010100001000101111111111111110
PC=00000018 regA=00000001 regB=ffffffff regC=00000002 gr0=00000000 gr1=00000001 gr2=ffffffff gr3=00000001 gr4=00000000 gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=0000001c regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=00000001 gr2=ffffffff gr3=00000001 gr4=00000000 gr5=ffffffff gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000000000000000000000000000
PC=0000000c regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=00000001 gr2=ffffffff gr3=00000001 gr4=00000000 gr5=ffffffff gr6=fffffffe gr7=xxxxxxxx instruction=00000000000000000000000000000000
PC=00000010 regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=00000001 gr2=ffffffff gr3=00000001 gr4=00000000 gr5=ffffffff gr6=fffffffe gr7=xxxxxxxx instruction=00000000001000100011000000100110
PC=00000014 regA=00000001 regB=ffffffff regC=fffffffe gr0=00000000 gr1=00000001 gr2=ffffffff gr3=00000001 gr4=00000000 gr5=ffffffff gr6=fffffffe gr7=xxxxxxxx instruction=00010100001000101111111111111110
PC=00000018 regA=00000001 regB=ffffffff regC=00000002 gr0=00000000 gr1=00000001 gr2=ffffffff gr3=00000001 gr4=00000000 gr5=ffffffff gr6=fffffffe gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=0000001c regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=00000001 gr2=ffffffff gr3=00000001 gr4=00000000 gr5=ffffffff gr6=fffffffe gr7=xxxxxxxx instruction=00000000000000000000000000000000
PC=0000000c regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=00000001 gr2=ffffffff gr3=00000001 gr4=00000000 gr5=ffffffff gr6=fffffffe gr7=xxxxxxxx instruction=00000000000000000000000000000000
PC=00000010 regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=00000001 gr2=ffffffff gr3=00000001 gr4=00000000 gr5=ffffffff gr6=fffffffe gr7=xxxxxxxx instruction=00000000001000100011000000100110
PC=00000014 regA=00000001 regB=ffffffff regC=fffffffe gr0=00000000 gr1=00000001 gr2=ffffffff gr3=00000001 gr4=00000000 gr5=ffffffff gr6=fffffffe gr7=xxxxxxxx instruction=00010100001000101111111111111110
```

2.4.8. Test for andi, ori, xori, slti

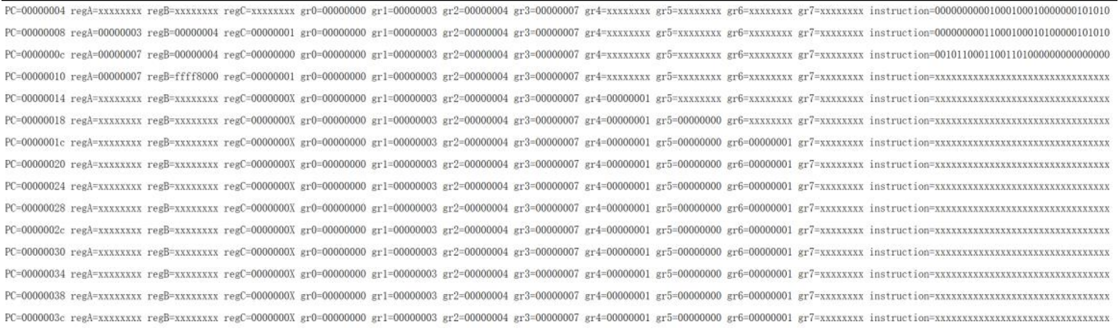
```
a. gr[1]=32'b0000_0000_0000_0000_0000_0000_0000_0001;
//gr1 = 1
i_datain[319:288]={6'b001100,`gr1,`gr2,16'b1111111111111111};
//andi   gr1 & 2**16-1 -> gr2
i_datain[287:256]={6'b001101,`gr1,`gr3,16'b1111111111111111};
//ori   gr1 | 2**16-1 -> gr3
i_datain[255:224]={6'b001110,`gr1,`gr4,16'b1111};
//xori   gr1 | 15 -> gr4
i_datain[223:192]={6'b001010,`gr1,`gr5,16'b1111111111111111};
//slti   gr2 < -1 -> gr5
```



```
PC=00000004 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=00000001 gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00110000001000101111111111111111
PC=00000008 regA=00000001 regB=ffffffff regC=00000001 gr0=00000000 gr1=00000001 gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00110100001000111111111111111111
PC=0000000c regA=00000001 regB=ffffffff regC=0000ffff gr0=00000000 gr1=00000001 gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00111000001001000000000000001111
PC=00000010 regA=00000001 regB=0000000f regC=0000000e gr0=00000000 gr1=00000001 gr2=xxxxxxxx gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00101000001001011111111111111111
PC=00000014 regA=00000001 regB=ffffffff regC=00000000 gr0=00000000 gr1=00000001 gr2=00000001 gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000018 regA=xxxxxxxx regB=xxxxxxxx regC=0000000X gr0=00000000 gr1=00000001 gr2=00000001 gr3=0000ffff gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=0000001c regA=xxxxxxxx regB=xxxxxxxx regC=0000000X gr0=00000000 gr1=00000001 gr2=00000001 gr3=0000ffff gr4=0000000e gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000020 regA=xxxxxxxx regB=xxxxxxxx regC=0000000X gr0=00000000 gr1=00000001 gr2=00000001 gr3=0000ffff gr4=0000000e gr5=00000000 gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000024 regA=xxxxxxxx regB=xxxxxxxx regC=0000000X gr0=00000000 gr1=00000001 gr2=00000001 gr3=0000ffff gr4=0000000e gr5=00000000 gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000028 regA=xxxxxxxx regB=xxxxxxxx regC=0000000X gr0=00000000 gr1=00000001 gr2=00000001 gr3=0000ffff gr4=0000000e gr5=00000000 gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=0000002c regA=xxxxxxxx regB=xxxxxxxx regC=0000000X gr0=00000000 gr1=00000001 gr2=00000001 gr3=0000ffff gr4=0000000e gr5=00000000 gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000030 regA=xxxxxxxx regB=xxxxxxxx regC=0000000X gr0=00000000 gr1=00000001 gr2=00000001 gr3=0000ffff gr4=0000000e gr5=00000000 gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000034 regA=xxxxxxxx regB=xxxxxxxx regC=0000000X gr0=00000000 gr1=00000001 gr2=00000001 gr3=0000ffff gr4=0000000e gr5=00000000 gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000038 regA=xxxxxxxx regB=xxxxxxxx regC=0000000X gr0=00000000 gr1=00000001 gr2=00000001 gr3=0000ffff gr4=0000000e gr5=00000000 gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=0000003c regA=xxxxxxxx regB=xxxxxxxx regC=0000000X gr0=00000000 gr1=00000001 gr2=00000001 gr3=0000ffff gr4=0000000e gr5=00000000 gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

2.4.9. Test for slt, sltiu

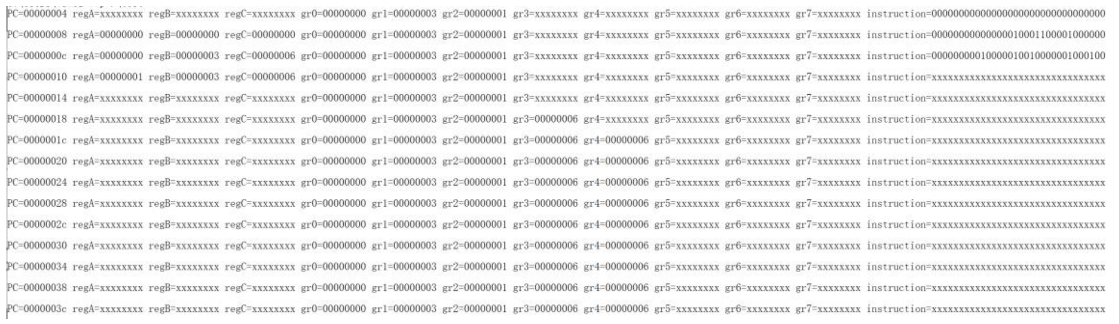
```
a. gr[1]=32'b0000_0000_0000_0000_0000_0000_0000_0011;
//gr1 = 3
a. gr[2]=32'b0000_0000_0000_0000_0000_0000_0000_0100;
//gr2 = 4
a. gr[3]=32'b0000_0000_0000_0000_0000_0000_0000_0111;
//gr3 = 7
i_datain[319:288]={6'b000000,`gr1,`gr2,`gr4,5'b0,6'b101010};
//slt  gr1 < gr2 -> gr4
i_datain[287:256]={6'b000000,`gr3,`gr2,`gr5,5'b0,6'b101010};
//slt  gr3 < gr2 -> gr5
i_datain[255:224]={6'b001011,`gr3,`gr6,16'b1000000000000000};
//sltiu  gr3 < 2**15 -> gr6
```



```
PC=00000004 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=00000003 gr2=00000004 gr3=00000007 gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000001000100010000000101010
PC=00000008 regA=00000003 regB=00000004 regC=00000001 gr0=00000000 gr1=00000003 gr2=00000004 gr3=00000007 gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000011000100010100000101010
PC=0000000c regA=00000007 regB=00000004 regC=00000000 gr0=00000000 gr1=00000003 gr2=00000004 gr3=00000007 gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00101100011001101000000000000000
PC=00000010 regA=00000007 regB=ffff8000 regC=00000001 gr0=00000000 gr1=00000003 gr2=00000004 gr3=00000007 gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000014 regA=xxxxxxxx regB=xxxxxxxx regC=0000000X gr0=00000000 gr1=00000003 gr2=00000004 gr3=00000007 gr4=00000001 gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000018 regA=xxxxxxxx regB=xxxxxxxx regC=0000000X gr0=00000000 gr1=00000003 gr2=00000004 gr3=00000007 gr4=00000001 gr5=00000000 gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=0000001c regA=xxxxxxxx regB=xxxxxxxx regC=0000000X gr0=00000000 gr1=00000003 gr2=00000004 gr3=00000007 gr4=00000001 gr5=00000000 gr6=00000001 gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000020 regA=xxxxxxxx regB=xxxxxxxx regC=0000000X gr0=00000000 gr1=00000003 gr2=00000004 gr3=00000007 gr4=00000001 gr5=00000000 gr6=00000001 gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000024 regA=xxxxxxxx regB=xxxxxxxx regC=0000000X gr0=00000000 gr1=00000003 gr2=00000004 gr3=00000007 gr4=00000001 gr5=00000000 gr6=00000001 gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000028 regA=xxxxxxxx regB=xxxxxxxx regC=0000000X gr0=00000000 gr1=00000003 gr2=00000004 gr3=00000007 gr4=00000001 gr5=00000000 gr6=00000001 gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=0000002c regA=xxxxxxxx regB=xxxxxxxx regC=0000000X gr0=00000000 gr1=00000003 gr2=00000004 gr3=00000007 gr4=00000001 gr5=00000000 gr6=00000001 gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000030 regA=xxxxxxxx regB=xxxxxxxx regC=0000000X gr0=00000000 gr1=00000003 gr2=00000004 gr3=00000007 gr4=00000001 gr5=00000000 gr6=00000001 gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000034 regA=xxxxxxxx regB=xxxxxxxx regC=0000000X gr0=00000000 gr1=00000003 gr2=00000004 gr3=00000007 gr4=00000001 gr5=00000000 gr6=00000001 gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000038 regA=xxxxxxxx regB=xxxxxxxx regC=0000000X gr0=00000000 gr1=00000003 gr2=00000004 gr3=00000007 gr4=00000001 gr5=00000000 gr6=00000001 gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=0000003c regA=xxxxxxxx regB=xxxxxxxx regC=0000000X gr0=00000000 gr1=00000003 gr2=00000004 gr3=00000007 gr4=00000001 gr5=00000000 gr6=00000001 gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

2.4.10. Test for sll, sllv

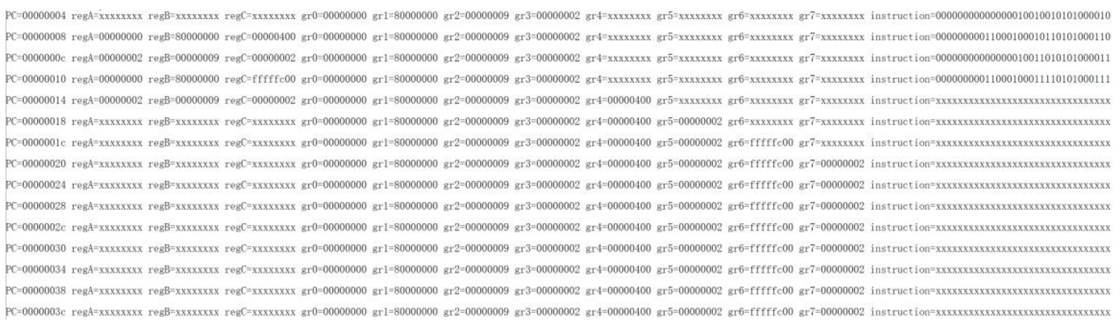
```
cpu.gr[1]=32'b0000_0000_0000_0000_0000_0000_0000_0011;
//gr1 = 3
cpu.gr[2]=32'b0000_0000_0000_0000_0000_0000_0000_0001;
//gr2 = 1
i_datain[319:288]={32'b0};
//NOP
i_datain[287:256]={6'b000000,`gr0,`gr1,`gr3,5'b1,6'b000000};
//sll  gr1 << 1 -> gr3
i_datain[255:224]={6'b000000,`gr2,`gr1,`gr4,5'b1,6'b000100};
//sllv  gr1 << gr2 -> gr4
```



```
PC=00000004 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=00000003 gr2=00000001 gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000000000000000000000000000
PC=00000008 regA=00000000 regB=00000000 regC=00000000 gr0=00000000 gr1=00000003 gr2=00000001 gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000000000010001100001000000
PC=0000000c regA=00000000 regB=00000003 regC=00000006 gr0=00000000 gr1=00000003 gr2=00000001 gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000010000010010000001000100
PC=00000010 regA=00000001 regB=00000003 regC=00000006 gr0=00000000 gr1=00000003 gr2=00000001 gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000014 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=00000003 gr2=00000001 gr3=xxxxxxxx gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000018 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=00000003 gr2=00000001 gr3=00000006 gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=0000001c regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=00000003 gr2=00000001 gr3=00000006 gr4=00000006 gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000020 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=00000003 gr2=00000001 gr3=00000006 gr4=00000006 gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000024 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=00000003 gr2=00000001 gr3=00000006 gr4=00000006 gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000028 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=00000003 gr2=00000001 gr3=00000006 gr4=00000006 gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=0000002c regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=00000003 gr2=00000001 gr3=00000006 gr4=00000006 gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000030 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=00000003 gr2=00000001 gr3=00000006 gr4=00000006 gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000034 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=00000003 gr2=00000001 gr3=00000006 gr4=00000006 gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000038 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=00000003 gr2=00000001 gr3=00000006 gr4=00000006 gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=0000003c regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=00000003 gr2=00000001 gr3=00000006 gr4=00000006 gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

2.4.11. Test for srl, srlv, sra, srav

```
cpu.gr[1]=32'b1000_0000_0000_0000_0000_0000_0000_0000;
//gr1 = 2**31
cpu.gr[2]=32'b0000_0000_0000_0000_0000_0000_0000_1001;
//gr2 = 9
cpu.gr[3]=32'b0000_0000_0000_0000_0000_0000_0000_0010;
//gr3 = 2
i_datain[319:288]={6'b000000,`gr0,`gr1,`gr4,5'b10101,6'b000010};
//srl  gr1 >> 21 -> gr4
i_datain[287:256]={6'b000000,`gr3,`gr2,`gr5,5'b10101,6'b000110};
//srlv  gr2 >> gr3 -> gr5
i_datain[255:224]={6'b000000,`gr0,`gr1,`gr6,5'b10101,6'b000011};
//sra  gr1 >>> 21 -> gr6
i_datain[223:192]={6'b000000,`gr3,`gr2,`gr7,5'b10101,6'b000111};
//srav  gr2 >>> gr3 -> gr7
```



```
PC=00000004 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=80000000 gr2=00000009 gr3=00000002 gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000000000001001001010100010
PC=00000008 regA=00000000 regB=80000000 regC=00000400 gr0=00000000 gr1=80000000 gr2=00000009 gr3=00000002 gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000011000100010110101000110
PC=0000000c regA=00000002 regB=00000009 regC=00000002 gr0=00000000 gr1=80000000 gr2=00000009 gr3=00000002 gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000000000001001101010100011
PC=00000010 regA=00000000 regB=80000000 regC=fffffc00 gr0=00000000 gr1=80000000 gr2=00000009 gr3=00000002 gr4=xxxxxxxx gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=00000000011000100011110101000111
PC=00000014 regA=00000002 regB=00000009 regC=00000002 gr0=00000000 gr1=80000000 gr2=00000009 gr3=00000002 gr4=00000400 gr5=xxxxxxxx gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000018 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=80000000 gr2=00000009 gr3=00000002 gr4=00000400 gr5=00000002 gr6=xxxxxxxx gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=0000001c regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=80000000 gr2=00000009 gr3=00000002 gr4=00000400 gr5=00000002 gr6=fffffc00 gr7=xxxxxxxx instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000020 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=80000000 gr2=00000009 gr3=00000002 gr4=00000400 gr5=00000002 gr6=fffffc00 gr7=00000002 instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000024 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=80000000 gr2=00000009 gr3=00000002 gr4=00000400 gr5=00000002 gr6=fffffc00 gr7=00000002 instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000028 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=80000000 gr2=00000009 gr3=00000002 gr4=00000400 gr5=00000002 gr6=fffffc00 gr7=00000002 instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=0000002c regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=80000000 gr2=00000009 gr3=00000002 gr4=00000400 gr5=00000002 gr6=fffffc00 gr7=00000002 instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000030 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=80000000 gr2=00000009 gr3=00000002 gr4=00000400 gr5=00000002 gr6=fffffc00 gr7=00000002 instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000034 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=80000000 gr2=00000009 gr3=00000002 gr4=00000400 gr5=00000002 gr6=fffffc00 gr7=00000002 instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=00000038 regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=80000000 gr2=00000009 gr3=00000002 gr4=00000400 gr5=00000002 gr6=fffffc00 gr7=00000002 instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PC=0000003c regA=xxxxxxxx regB=xxxxxxxx regC=xxxxxxxx gr0=00000000 gr1=80000000 gr2=00000009 gr3=00000002 gr4=00000400 gr5=00000002 gr6=fffffc00 gr7=00000002 instruction=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

2.5. Discussion & Further Improvement

In this project, I did not consider the synchronization of the general registers and the memory. In other words, I did not add the clock signal to the general registers or the memory. I just used some combinational logics to build the registers and the memory. Read/write operations will be finished instantly once the input is given. However, in reality, a clock signal should be given to synchronize the register and memory operations. Also, these operations should have some delays.

Besides, in this project, branches are issued in the Memory (MEM) stage, while jumps are issued in the Execution (EX) stage. This mechanism gives a taken branch a 4-cycle delay, while a jump a 3-cycle delay. These delays could be reduced to 2-cycle by moving the branch judgement and jump judgement to the Instruction Decode (ID) stage. Furthermore, since the latest register data are now needed in the ID stage, forwarding logic from the WB, MEM, and EX stages to the ID stage should be implemented.