UM–SJTU Joint Institute
Introduction to Computer Organization (VE370)

Project Report Two

CPU Implementation

Group 7

Name: Liu Yihao ID: 515370910207
Name: Wu Guangzheng ID: 515370910030
Name: Jiang Yicheng ID: 515370910224

November 24, 2017

1 Objective

In this project, we will model both single cycle and pipelined implementation of MIPS computer in Verilog that support a subset of MIPS instruction set including:

- The memory-reference instructions load word (lw) and store word (sw)
- The arithmetic-logical instructions add, addi, sub, and, andi, or, and slt
- The jumping instructions branch equal (beq), branch not equal (bne), and jump (j)

2 Procedure

2.1 Single-Cycle CPU Design

For a Single-Cycle CPU, we just follows the design shown in Figure 1:

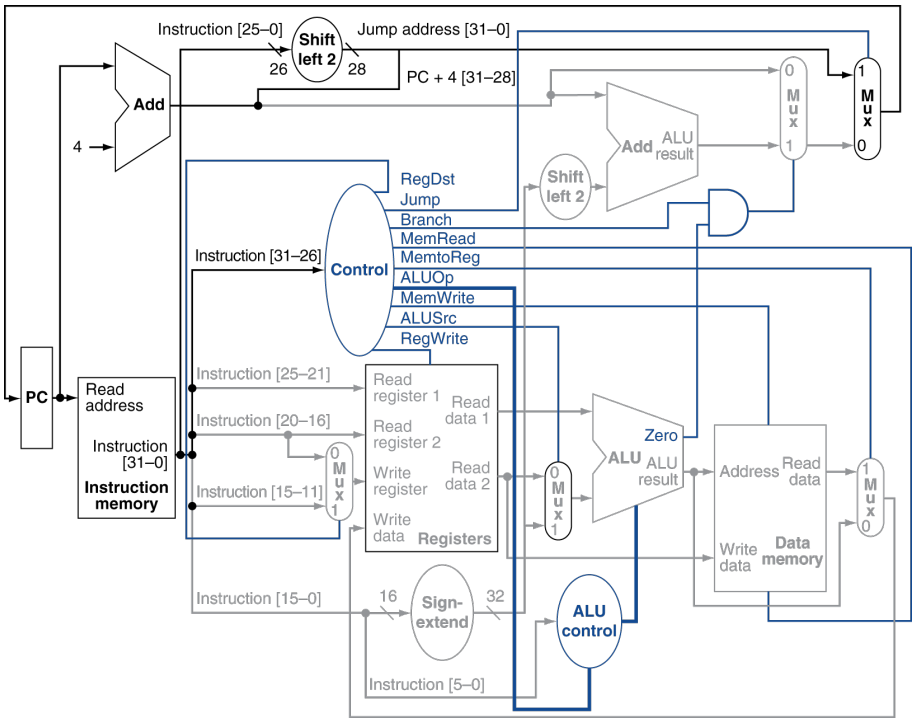


Figure 1: Design of Single-Cycle CPU

As we can see, it consists of a "**PC**" which is used to fetch the instruction from "**Instruction Memory**". Also there is an "**adder**" with one additive kept as constant 4 to promote the "**PC**" to next instruction. To deal with jumping instructions, there is a "**shifter**" which will shift the "relative address" left by 2 bits. And the final PC address will be chosed through two "**mux**"es. The signal deciding the choice of mux comes from "**ALU**" (which is used to do all arithmetic including "and", "add", "compare") and "**control**" (according to instruction, give out control signal to other blocks). The "**ALU**" is controled by a block called "**ALU Control**" (which is controled by "**Control**"). And the "**Registers**" can offers 32 registers for instructions and the data can be stored in or load from "**Data memory**". Last but not least, there is a block called "**signextend**" to tell the CPU that the data is negative (change to corresponding 2'complement number).

Some other "**mux**"es are used to choose the data needed to be used according to different kinds of instructions.

And in our design, we use these modules implemented in verilog to realize this design:

Block Name	Module Name (File name)
ALU	alu.v
ALU Control	alu_control.v
Control	control.v
Instruction Memory	im.v
mux	mux2.v
Registers	registers.v
Data Memory	dm.v
signextend	sign_extend.v
PC	pc.v
other blocks	single_cycle.v

Table 1: Module used in Single-Cycle CPU

With codes shown in appendix, finally we obtain RTL schematic of Single-cycle CPU shown in Figure 3:

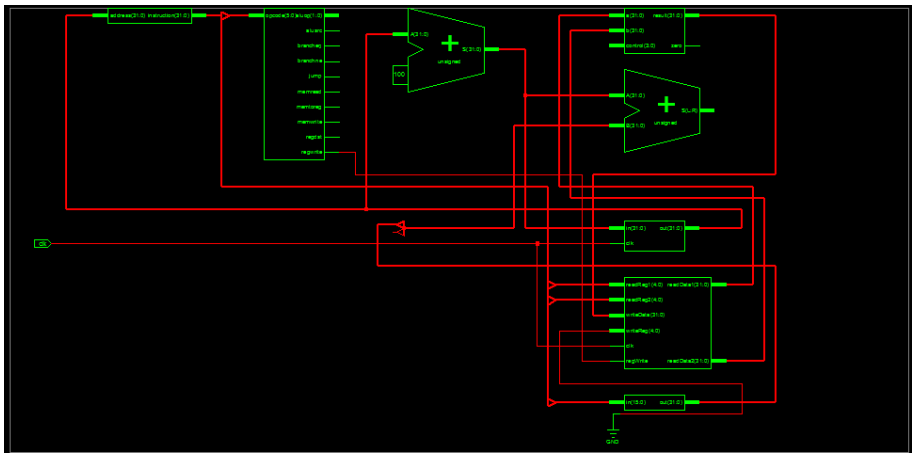


Figure 2: RTL schematic of Single-cycle CPU designed in Verilog

2.2 Pipelined CPU Design

For a Pipelined CPU, we mainly follows the design shown in Figure 2:

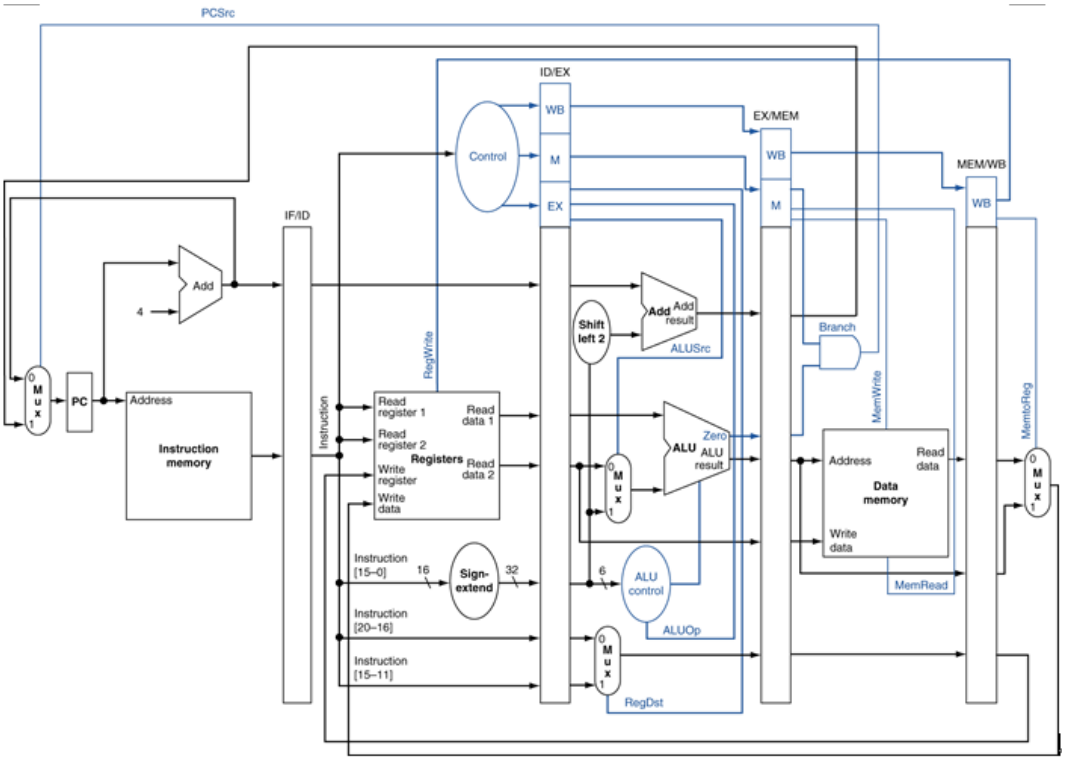


Figure 3: Design of Pipelined CPU

To improve the performance of a single-cycle CPU, we can overlap some execution. That is to divide the whole exection of an instruction into five different parts:

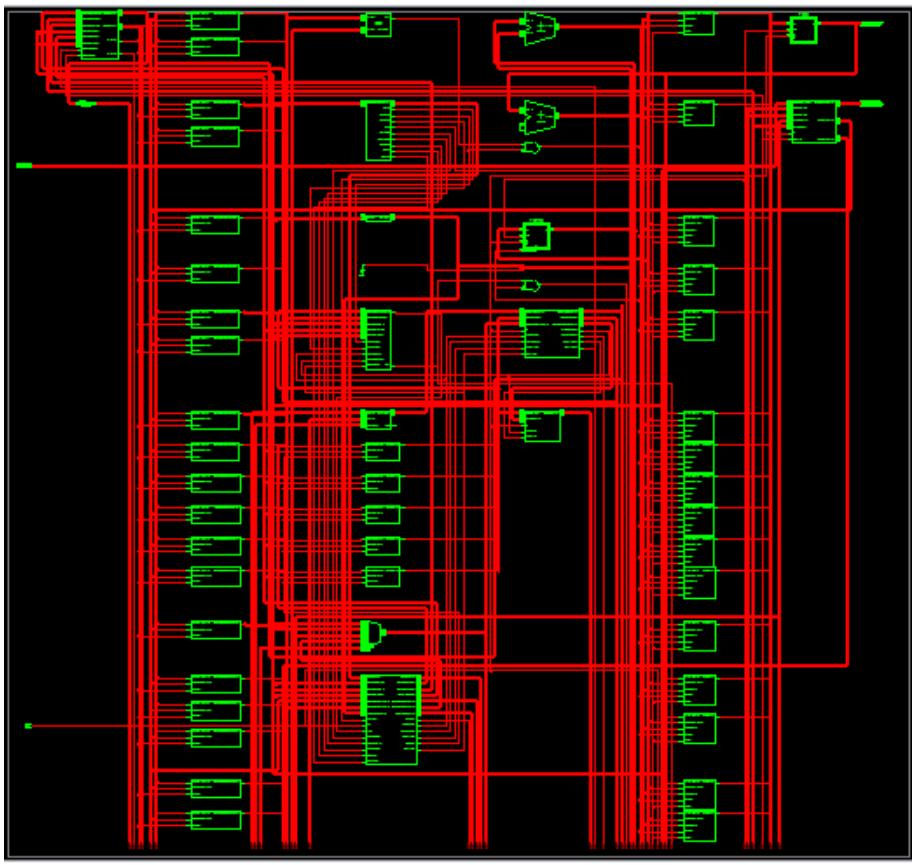
1. "IF": Instruction fetch
2. "ID": Instruction decode/register file read
3. "EX": Execute/address calculation (do all kinds of arithmetic)
4. "MEM": Memory access (called by load word and store word)
5. "WB": Write back (called by store word)

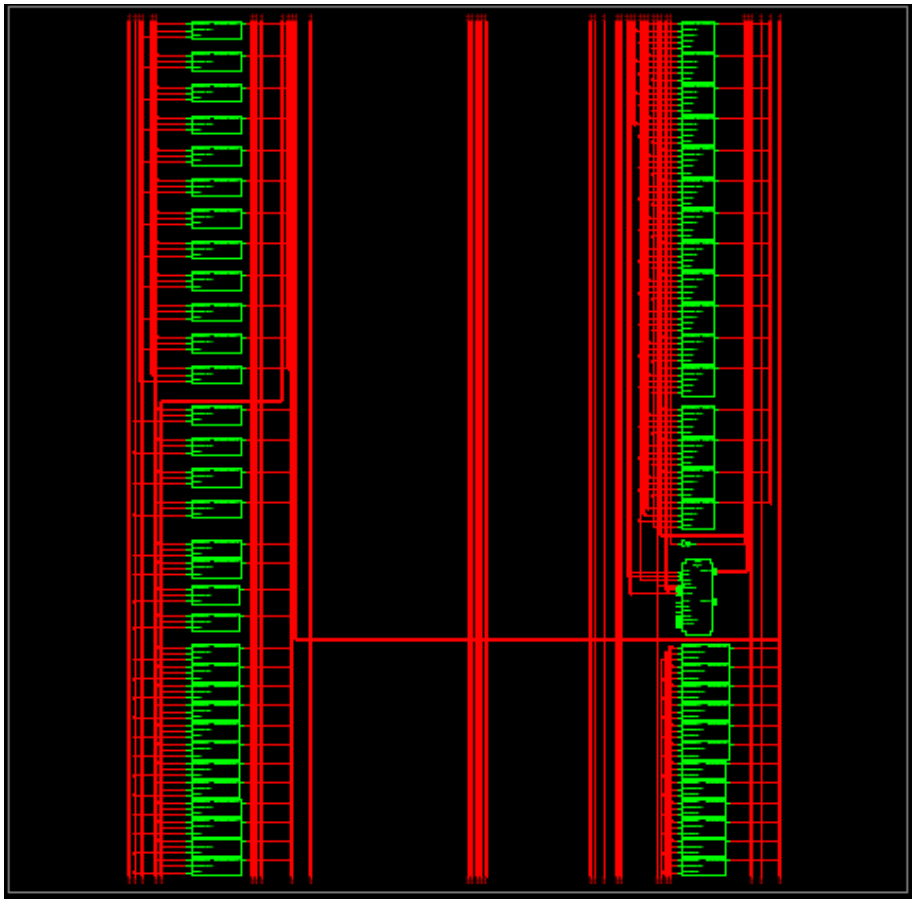
We can see that all the blocks used in a single-cycle CPU are also used in a Pipelined CPU. The main difference is that there are four large registers each used between two fields, called "**IF/ID**", "**ID/EX**", "**EX/MEM**" and "**MEM/WB**". But since we deal with several instructions at the same time, there will be data hazard, control harzard and branch hazard. To solve these problems, we have to stall the CPU or flush some parts or forward according to different needs. So we need a block called "Hazard Detection" and a block called "Forward".

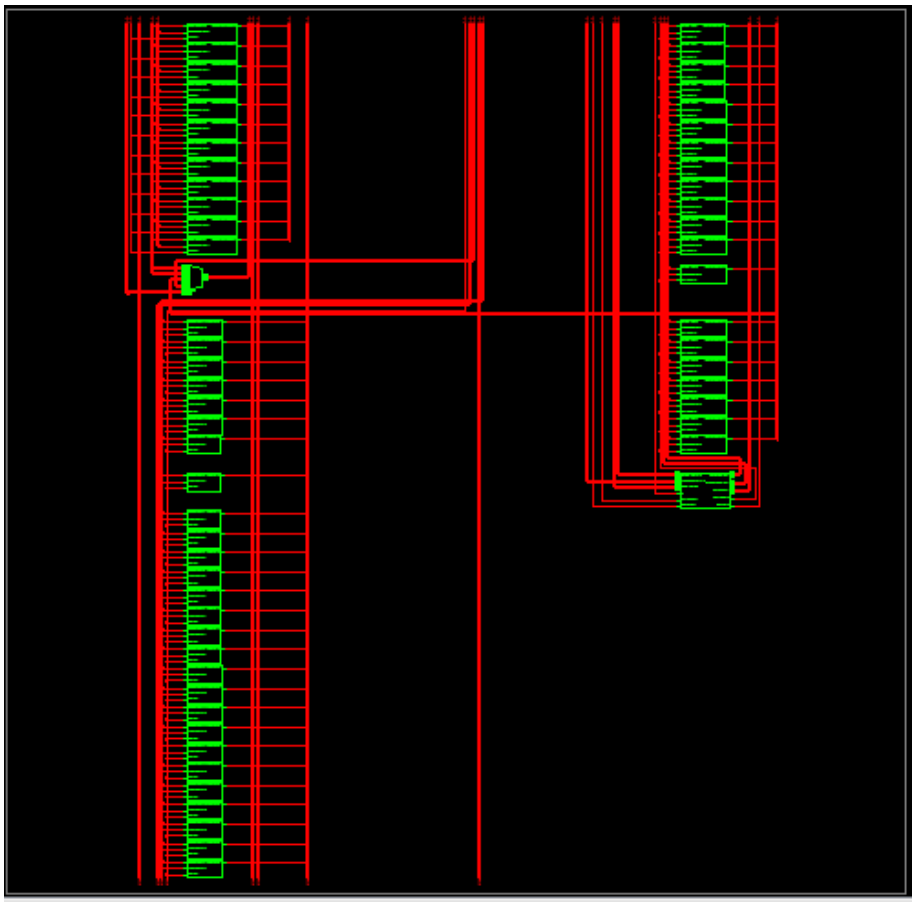
Block Name	Module Name (File name)
ALU	alu.v
ALU Control	alu_control.v
Control	control.v
Instruction Memory	im.v
mux	mux2.v
Registers	registers.v
Data Memory	dm.v
signextend	sign_extend.v
PC	pc.v
IF/ID	if_ id.v
ID/EX	id_ ex.v
EX/MEM	ex_ mem.v
MEM/WB	mem_ wb.v
Forward	forward.v
HarzardDetection	hazard_detection.v
other blocks	single_ cycle.v

Table 2: Module used in Pipelined CPU

With codes shown in appendix, finally we obtain RTL schematic of Single-cycle CPU shown in Figure 3 (4 parts together):







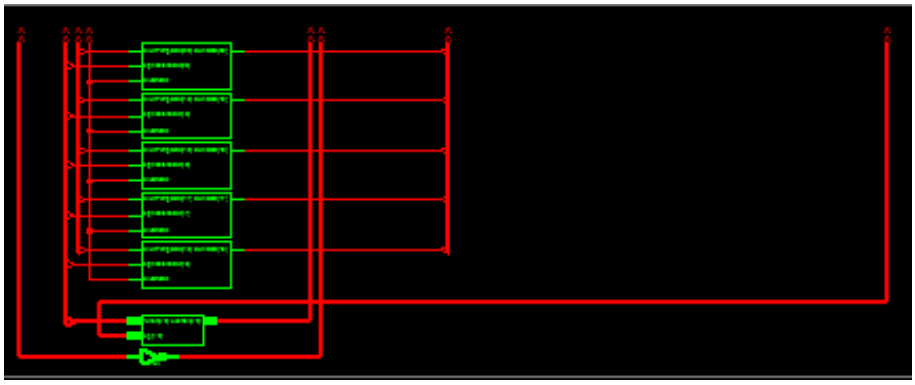


Figure 4: RTL schematic of Pipeline-CPU designed in Verilog

3 Test Plan

We use the following set of MIPS instructions to check our implementation.

```
memory[0]  = 32'b00100000000010000000000000100000; //addi $t0, $zero, 0x20
memory[1]  = 32'b00100000000010010000000000100111; //addi $t1, $zero, 0x27
memory[2]  = 32'b00000001000010011000000000100100; //and $s0, $t0, $t1
memory[3]  = 32'b00000001000010011000000000100101; //or $s0, $t0, $t1
memory[4]  = 32'b10101100000010000000000000000100; //sw $s0, 4($zero)
memory[5]  = 32'b10101100000010000000000000001000; //sw $t0, 8($zero)
memory[6]  = 32'b00000001000010011000100000100000; //add $s1, $t0, $t1
memory[7]  = 32'b00000001000010011001000000100010; //sub $s2, $t0, $t1
memory[8]  = 32'b00010010001100100000000000001001; //beq $s1, $s2, error0
memory[9]  = 32'b10001100000100010000000000000100; //lw $s1, 4($zero)
memory[10] = 32'b00110010001100100000000000011000; //andi $s2, $s1, 0x18
memory[11] = 32'b00010010001100100000000000001001; //beq $s1, $s2, error1
memory[12] = 32'b10001100000100110000000000001000; //lw $s3, 8($zero)
memory[13] = 32'b00010010000100110000000000001010; //beq $s0, $s3, error2
memory[14] = 32'b00000010010100011010000000101010; //slt $s4,$s2,$s1(Last)
memory[15] = 32'b00010010100000000000000000001111; //beq $s4, $0, EXIT
memory[16] = 32'b00000001000100000100100000100000; //add $s2, $s1, $0
memory[17] = 32'b00001000000000000000000000001110; //j Last
memory[18] = 32'b00100000000010000000000000000000; //addi $t0,$0,0(error0)
```

```
20 memory[19] = 32'b00100000000010010000000000000000; //addi $t1, $0, 0
21 memory[20] = 32'b00001000000000000000000000011111; //j EXIT
22 memory[21] = 32'b00100000000010000000000000000001; //addi
   ↪ $t0,$0,1(error1)
23 memory[22] = 32'b00100000000010010000000000000001; //addi $t1, $0, 1
24 memory[23] = 32'b00001000000000000000000000011111; //j EXIT
25 memory[24] = 32'b00100000000010000000000000000010; //addi
   ↪ $t0,$0,2(error2)
26 memory[25] = 32'b00100000000010010000000000000010; //addi $t1, $0, 2
27 memory[26] = 32'b00001000000000000000000000011111; //j EXIT
28 memory[27] = 32'b00100000000010000000000000000011; //addi
   ↪ $t0,$0,3(error3)
29 memory[28] = 32'b00100000000010010000000000000011; //addi $t1, $0, 3
30 memory[29] = 32'b00001000000000000000000000011111; //j EXIT
```

3.1 Theoretical simulation result for Single-cycle CPU

For a single-cycle CPU, we will check the value of PC (Program Counter) and register $s0-$s7 and $t0-$t9 at eack clock cycle and display them in the console window in Xilinx ISE. Then we can find out whether it works as we expects. And the result should be:

Cycle	PC	Register
0	0x00000004	$t0=0x20
1	0x00000008	$t0=0x20, $t1=0x27
2	0x0000000c	$t0=0x20, $t1=0x27, $s0=0x20
3	0x00000010	$t0=0x20, $t1=0x27, $s0=0x27
4	0x00000014	$t0=0x20, $t1=0x27, $s0=0x27
5	0x00000018	$t0=0x20, $t1=0x27, $s0=0x27
6	0x0000001c	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x47
7	0x00000020	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x47, $s2=0xfffffff9
8	0x00000024	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x47, $s2=0xfffffff9
9	0x00000028	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x27, $s2=0xfffffff9
10	0x0000002c	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x27
11	0x00000030	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x27
12	0x00000034	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x27, $s3=0x20
13	0x00000038	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x27, $s3=0x20
14	0x0000003c	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x27, $s3=0x20, $s4=0x1
15	0x00000040	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x27, $s3=0x20, $s4=0x1
16	0x00000044	$t0=0x20, $t1=0x27, $s0=$s1=$s2=0x27, $s3=0x20, $s4=0x1
17	0x00000038	$t0=0x20, $t1=0x27, $s0=$s1=$s2=0x27, $s3=0x20, $s4=0x1
18	0x0000003c	$t0=0x20, $t1=0x27, $s0=$s1=$s2=0x27,$s3=0x20
19	0x0000007c	$t0=0x20, $t1=0x27, $s0=$s1=$s2=0x27, $s3=0x20
20	0xxxxxxxxx	$t0=0x20, $t1=0x27, $s0=$s1=$s2=0x27, $s3=0x20
>21	0xxxxxxxxx	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x27, $s2=27, $s3=0x20

Table 3: Theoretical simulation result for single-cycle CPU (only focus on CLK=1)

3.2 Theoretical simulation result for Pipelined CPU

For a Pipelined CPU, we will check the value of PC and register $s0-$s7 and $t0-$t3 at eack clock cycle and the result should be:

Cycle	CLK	PC	Register
0	0	0x00000000	
0	1	0x00000004	
1	0	0x00000004	
1	1	0x00000008	
2	0	0x00000008	
2	1	0x0000000c	
3	0	0x0000000c	
3	1	0x00000010	
4	0	0x00000010	$t0=0x20
			Comment: Forwarding compared with single-cycled cpu
4	1	0x00000014	$t0=0x20
5	0	0x00000014	$t0=0x20, $t1=0x27
5	1	0x00000018	$t0=0x20, $t1=0x27
6	0	0x00000018	$t0=0x20, $t1=0x27, $s0=0x27
6	1	0x0000001c	$t0=0x20, $t1=0x27, $s0=0x27

7	0	0x0000001c	$t0=0x20, $t1=0x27, $s0=0x27
7	1	0x00000020	$t0=0x20, $t1=0x27, $s0=0x27
8	0	0x00000020	$t0=0x20, $t1=0x27, $s0=0x27
8	1	0x00000024	$t0=0x20, $t1=0x27, $s0=0x27
9	0	0x00000024	$t0=0x20, $t1=0x27, $s0=0x27
9	1	0x00000024	$t0=0x20, $t1=0x27, $s0=0x27
			Comment: Here (9) is stalled by one clock cycle due to hazard
10	0	0x00000024	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x47
10	1	0x00000028	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x47
11	0	0x00000028	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x47, $s2=ffffff9
11	1	0x0000002c	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x47, $s2=ffffff9
12	0	0x0000002c	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x47, $s2=ffffff9
12	1	0x0000002c	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x47, $s2=ffffff9
13	0	0x0000002c	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x47, $s2=ffffff9
13	1	0x00000030	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x47, $s2=ffffff9
14	0	0x00000030	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x27, $s2=ffffff9
14	1	0x00000030	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x27, $s2=ffffff9
15	0	0x00000030	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x27, $s2=ffffff9
15	1	0x00000034	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x27, $s2=ffffff9
16	0	0x00000034	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x27
16	1	0x00000038	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x27
17	0	0x00000038	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x27
17	1	0x00000038	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x27
18	0	0x00000038	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x27
18	1	0x00000038	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x27
			Comment: Here (16-18) is stalled by three clock cycles due to hazard
19	0	0x00000038	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x27, $s3=0x20
19	1	0x0000003c	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x27, $s3=0x20
20	0	0x0000003c	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x27, $s3=0x20
20	1	0x00000040	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x27, $s3=0x20
21	0	0x00000040	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x27, $s3=0x20
21	1	0x00000040	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x27, $s3=0x20
			Comment: Here (21) is stalled by three clock cycles due to hazard
22	0	0x00000040	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x27, $s3=0x20
22	1	0x00000044	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x27, $s3=0x20, $s4=0x1
23	0	0x00000044	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x27, $s3=0x20, $s4=0x1
23	1	0x00000048	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x27, $s3=0x20, $s4=0x1
24	0	0x00000048	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x27, $s3=0x20, $s4=0x1
24	1	0x00000038	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x27, $s3=0x20, $s4=0x1
25	0	0x00000038	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x27, $s3=0x20, $s4=0x1
25	1	0x0000003c	$t0=0x20, $t1=0x27, $s0=0x27, $s1=0x27, $s3=0x20, $s4=0x1
26	0	0x0000003c	$t0=0x20, $t1=$s0=$s1=$s2=0x27, $s3=0x20, $s4=0x1
26	1	0x00000040	$t0=0x20, $t1=$s0=$s1=$s2=0x27, $s3=0x20, $s4=0x1
27	0	0x00000040	$t0=0x20, $t1=$s0=$s1=$s2=0x27, $s3=0x20, $s4=0x1
27	1	0x00000040	$t0=0x20, $t1=$s0=$s1=$s2=0x27, $s3=0x20, $s4=0x1
			Comment: Here (27) is stalled by three clock cycles due to hazard
28	0	0x00000040	$t0=0x20, $t1=$s0=$s1=$s2=0x27, $s3=0x20, $s4=0x1
28	1	0x0000007c	$t0=0x20, $t1=$s0=$s1=$s2=0x27, $s3=0x20, $s4=0x1

Table 4: Theoretical simulation result for Pipelined CPU

4 Simulation Results

4.1 Single-Cycle CPU Test Result

Here is the simulation results of single-cycle CPU.

```
This is a Full version of ISE Simulator(ISim).

Simulator is doing circuit initialization process.
Finished circuit initialization process.
Time:           0, CLK = 0, PC = 0x00000000
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000000
[$t1] = 0x00000000, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
-----------------------------------------------------
Time:           0, CLK = 1, PC = 0x00000004
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000000, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
-----------------------------------------------------
Time:           1, CLK = 0, PC = 0x00000004
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000000, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
-----------------------------------------------------
Time:           1, CLK = 1, PC = 0x00000008
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
-----------------------------------------------------
Time:           2, CLK = 0, PC = 0x00000008
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
```

```
-------------------------------------------------------
Time:              2, CLK = 1, PC = 0x0000000c
[$s0] = 0x00000020, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
-------------------------------------------------------
Time:              3, CLK = 0, PC = 0x0000000c
[$s0] = 0x00000020, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
-------------------------------------------------------
Time:              3, CLK = 1, PC = 0x00000010
[$s0] = 0x00000027, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
-------------------------------------------------------
Time:              4, CLK = 0, PC = 0x00000010
[$s0] = 0x00000027, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
-------------------------------------------------------
Time:              4, CLK = 1, PC = 0x00000014
[$s0] = 0x00000027, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
-------------------------------------------------------
Time:              5, CLK = 0, PC = 0x00000014
[$s0] = 0x00000027, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
```

```
92  ----------------------------------------------------------
93  Time:            5, CLK = 1, PC = 0x00000018
94  [$s0] = 0x00000027, [$s1] = 0x00000000, [$s2] = 0x00000000
95  [$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
96  [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
97  [$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
98  [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
99  [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
100 ----------------------------------------------------------
101 Time:            6, CLK = 0, PC = 0x00000018
102 [$s0] = 0x00000027, [$s1] = 0x00000000, [$s2] = 0x00000000
103 [$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
104 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
105 [$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
106 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
107 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
108 ----------------------------------------------------------
109 Time:            6, CLK = 1, PC = 0x0000001c
110 [$s0] = 0x00000027, [$s1] = 0x00000047, [$s2] = 0x00000000
111 [$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
112 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
113 [$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
114 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
115 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
116 ----------------------------------------------------------
117 Time:            7, CLK = 0, PC = 0x0000001c
118 [$s0] = 0x00000027, [$s1] = 0x00000047, [$s2] = 0x00000000
119 [$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
120 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
121 [$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
122 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
123 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
124 ----------------------------------------------------------
125 Time:            7, CLK = 1, PC = 0x00000020
126 [$s0] = 0x00000027, [$s1] = 0x00000047, [$s2] = 0xfffffff9
127 [$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
128 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
129 [$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
130 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
131 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
132 ----------------------------------------------------------
133 Time:            8, CLK = 0, PC = 0x00000020
134 [$s0] = 0x00000027, [$s1] = 0x00000047, [$s2] = 0xfffffff9
135 [$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
136 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
137 [$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
138 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
139 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
```

```
140 ----------------------------------------------------------
141 Time:           8, CLK = 1, PC = 0x00000024
142 [$s0] = 0x00000027, [$s1] = 0x00000047, [$s2] = 0xfffffff9
143 [$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
144 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
145 [$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
146 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
147 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
148 ----------------------------------------------------------
149 Time:           9, CLK = 0, PC = 0x00000024
150 [$s0] = 0x00000027, [$s1] = 0x00000047, [$s2] = 0xfffffff9
151 [$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
152 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
153 [$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
154 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
155 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
156 ----------------------------------------------------------
157 Time:           9, CLK = 1, PC = 0x00000028
158 [$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0xfffffff9
159 [$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
160 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
161 [$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
162 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
163 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
164 ----------------------------------------------------------
165 Time:          10, CLK = 0, PC = 0x00000028
166 [$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0xfffffff9
167 [$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
168 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
169 [$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
170 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
171 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
172 ----------------------------------------------------------
173 Time:          10, CLK = 1, PC = 0x0000002c
174 [$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000000
175 [$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
176 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
177 [$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
178 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
179 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
180 ----------------------------------------------------------
181 Time:          11, CLK = 0, PC = 0x0000002c
182 [$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000000
183 [$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
184 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
185 [$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
186 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
187 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
```

```
188 ----------------------------------------------------------
189 Time:           11, CLK = 1, PC = 0x00000030
190 [$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000000
191 [$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
192 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
193 [$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
194 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
195 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
196 ----------------------------------------------------------
197 Time:           12, CLK = 0, PC = 0x00000030
198 [$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000000
199 [$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
200 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
201 [$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
202 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
203 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
204 ----------------------------------------------------------
205 Time:           12, CLK = 1, PC = 0x00000034
206 [$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000000
207 [$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
208 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
209 [$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
210 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
211 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
212 ----------------------------------------------------------
213 Time:           13, CLK = 0, PC = 0x00000034
214 [$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000000
215 [$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
216 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
217 [$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
218 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
219 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
220 ----------------------------------------------------------
221 Time:           13, CLK = 1, PC = 0x00000038
222 [$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000000
223 [$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
224 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
225 [$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
226 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
227 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
228 ----------------------------------------------------------
229 Time:           14, CLK = 0, PC = 0x00000038
230 [$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000000
231 [$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
232 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
233 [$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
234 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
235 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
```

```
236 ----------------------------------------------------
237 Time:          14, CLK = 1, PC = 0x0000003c
238 [$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000000
239 [$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
240 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
241 [$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
242 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
243 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
244 ----------------------------------------------------
245 Time:          15, CLK = 0, PC = 0x0000003c
246 [$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000000
247 [$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
248 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
249 [$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
250 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
251 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
252 ----------------------------------------------------
253 Time:          15, CLK = 1, PC = 0x00000040
254 [$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000000
255 [$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
256 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
257 [$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
258 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
259 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
260 ----------------------------------------------------
261 Time:          16, CLK = 0, PC = 0x00000040
262 [$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000000
263 [$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
264 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
265 [$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
266 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
267 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
268 ----------------------------------------------------
269 Time:          16, CLK = 1, PC = 0x00000044
270 [$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000027
271 [$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
272 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
273 [$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
274 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
275 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
276 ----------------------------------------------------
277 Time:          17, CLK = 0, PC = 0x00000044
278 [$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000027
279 [$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
280 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
281 [$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
282 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
283 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
```

```
284 ----------------------------------------------------------
285 Time:           17, CLK = 1, PC = 0x00000038
286 [$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000027
287 [$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
288 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
289 [$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
290 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
291 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
292 ----------------------------------------------------------
293 Time:           18, CLK = 0, PC = 0x00000038
294 [$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000027
295 [$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
296 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
297 [$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
298 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
299 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
300 ----------------------------------------------------------
301 Time:           18, CLK = 1, PC = 0x0000003c
302 [$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000027
303 [$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
304 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
305 [$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
306 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
307 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
308 ----------------------------------------------------------
309 Time:           19, CLK = 0, PC = 0x0000003c
310 [$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000027
311 [$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
312 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
313 [$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
314 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
315 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
316 ----------------------------------------------------------
317 Time:           19, CLK = 1, PC = 0x0000007c
318 [$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000027
319 [$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
320 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
321 [$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
322 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
323 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
324 ----------------------------------------------------------
325 Time:           20, CLK = 0, PC = 0x0000007c
326 [$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000027
327 [$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
328 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
329 [$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
330 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
331 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
```

```
------------------------------------------------------
Time:          20, CLK = 1, PC = 0xxxxxxxxx
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000027
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
------------------------------------------------------
Time:          21, CLK = 0, PC = 0xxxxxxxxx
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000027
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
------------------------------------------------------
Time:          21, CLK = 1, PC = 0xxxxxxxxx
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000027
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
------------------------------------------------------

```

We can see that the result conforms to our expectation.

4.2 Pipelined CPU Test Result

Here is the simulation results of Pipelined CPU.

```
This is a Full version of ISE Simulator(ISim).

Simulator is doing circuit initialization process.
Finished circuit initialization process.
Time:           0, CLK = 0, PC = 0x00000000
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000000
[$t1] = 0x00000000, [$t2] = 0x00000000, [$t3] = 0x00000000
------------------------------------------------------
Time:           0, CLK = 1, PC = 0x00000004
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000000
[$t1] = 0x00000000, [$t2] = 0x00000000, [$t3] = 0x00000000
------------------------------------------------------
```

```
Time:            1, CLK = 0, PC = 0x00000004
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000000
[$t1] = 0x00000000, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:            1, CLK = 1, PC = 0x00000008
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000000
[$t1] = 0x00000000, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:            2, CLK = 0, PC = 0x00000008
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000000
[$t1] = 0x00000000, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:            2, CLK = 1, PC = 0x0000000c
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000000
[$t1] = 0x00000000, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:            3, CLK = 0, PC = 0x0000000c
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000000
[$t1] = 0x00000000, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:            3, CLK = 1, PC = 0x00000010
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000000
[$t1] = 0x00000000, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:            4, CLK = 0, PC = 0x00000010
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000000, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:            4, CLK = 1, PC = 0x00000014
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000000, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
```

```
Time:            5, CLK = 0, PC = 0x00000014
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
--------------------------------------------------------------
Time:            5, CLK = 1, PC = 0x00000018
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
--------------------------------------------------------------
Time:            6, CLK = 0, PC = 0x00000018
[$s0] = 0x00000020, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
--------------------------------------------------------------
Time:            6, CLK = 1, PC = 0x0000001c
[$s0] = 0x00000020, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
--------------------------------------------------------------
Time:            7, CLK = 0, PC = 0x0000001c
[$s0] = 0x00000027, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
--------------------------------------------------------------
Time:            7, CLK = 1, PC = 0x00000020
[$s0] = 0x00000027, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
--------------------------------------------------------------
Time:            8, CLK = 0, PC = 0x00000020
[$s0] = 0x00000027, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
--------------------------------------------------------------
Time:            8, CLK = 1, PC = 0x00000024
[$s0] = 0x00000027, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
--------------------------------------------------------------
```

```
Time:           9, CLK = 0, PC = 0x00000024
[$s0] = 0x00000027, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:           9, CLK = 1, PC = 0x00000024
[$s0] = 0x00000027, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:          10, CLK = 0, PC = 0x00000024
[$s0] = 0x00000027, [$s1] = 0x00000047, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:          10, CLK = 1, PC = 0x00000028
[$s0] = 0x00000027, [$s1] = 0x00000047, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:          11, CLK = 0, PC = 0x00000028
[$s0] = 0x00000027, [$s1] = 0x00000047, [$s2] = 0xfffffff9
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:          11, CLK = 1, PC = 0x0000002c
[$s0] = 0x00000027, [$s1] = 0x00000047, [$s2] = 0xfffffff9
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:          12, CLK = 0, PC = 0x0000002c
[$s0] = 0x00000027, [$s1] = 0x00000047, [$s2] = 0xfffffff9
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:          12, CLK = 1, PC = 0x0000002c
[$s0] = 0x00000027, [$s1] = 0x00000047, [$s2] = 0xfffffff9
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
```

```
Time:           13, CLK = 0, PC = 0x0000002c
[$s0] = 0x00000027, [$s1] = 0x00000047, [$s2] = 0xfffffff9
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:           13, CLK = 1, PC = 0x00000030
[$s0] = 0x00000027, [$s1] = 0x00000047, [$s2] = 0xfffffff9
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:           14, CLK = 0, PC = 0x00000030
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0xfffffff9
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:           14, CLK = 1, PC = 0x00000030
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0xfffffff9
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:           15, CLK = 0, PC = 0x00000030
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0xfffffff9
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:           15, CLK = 1, PC = 0x00000034
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0xfffffff9
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:           16, CLK = 0, PC = 0x00000034
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:           16, CLK = 1, PC = 0x00000038
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
```

```
Time:           17, CLK = 0, PC = 0x00000038
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:           17, CLK = 1, PC = 0x00000038
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:           18, CLK = 0, PC = 0x00000038
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:           18, CLK = 1, PC = 0x00000038
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:           19, CLK = 0, PC = 0x00000038
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:           19, CLK = 1, PC = 0x0000003c
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:           20, CLK = 0, PC = 0x0000003c
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:           20, CLK = 1, PC = 0x00000040
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
```

```
Time:           21, CLK = 0, PC = 0x00000040
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:           21, CLK = 1, PC = 0x00000040
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:           22, CLK = 0, PC = 0x00000040
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:           22, CLK = 1, PC = 0x00000044
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:           23, CLK = 0, PC = 0x00000044
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:           23, CLK = 1, PC = 0x00000048
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:           24, CLK = 0, PC = 0x00000048
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time:           24, CLK = 1, PC = 0x00000038
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
```

```
Time:           25, CLK = 0, PC = 0x00000038
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
--------------------------------------------------------------
Time:           25, CLK = 1, PC = 0x0000003c
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
--------------------------------------------------------------
Time:           26, CLK = 0, PC = 0x0000003c
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000027
[$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
--------------------------------------------------------------
Time:           26, CLK = 1, PC = 0x00000040
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000027
[$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
--------------------------------------------------------------
Time:           27, CLK = 0, PC = 0x00000040
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000027
[$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
--------------------------------------------------------------
Time:           27, CLK = 1, PC = 0x00000040
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000027
[$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
--------------------------------------------------------------
Time:           28, CLK = 0, PC = 0x00000040
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000027
[$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000000
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
--------------------------------------------------------------
Time:           28, CLK = 1, PC = 0x0000007c
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000027
[$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000000
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
--------------------------------------------------------------
```

```
Time:          29, CLK = 0, PC = 0x0000007c
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000027
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000000
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
--------------------------------------------------------
Time:          29, CLK = 1, PC = 0x00000080
[$s0] = 0x00000027, [$s1] = 0x00000027, [$s2] = 0x00000027
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000000
[$t1] = 0x00000027, [$t2] = 0x00000000, [$t3] = 0x00000000
```

We can see that the result conforms to our expectation.

5 Conclusion

In this project, we write the single cycle and pipeline implementation of MIPS computer in Verilog. From the project, we can see how the MIPS instructions are based on the CPU. The design supports a subset of MIPS instruction set including the memory-reference instructions load word (lw) and store word (sw), the arithmetic-logical instructions add, addi, sub, and, andi,or, and slt,and the jumping instructions branch equal (beq), branch not equal (bne),and jump (j).

For the single cycle, we just simply write how an instruction can be dealt by the CPU. For a x32 Edition computers, every instruction includes 32 digit, and we can deal with them one by one. It is a quite basic idea, and rather naive idea to change the programs only known by human to binary instructions known by both human and computers. But however, it can only deal with a single instruction at a time, which is rather slow.

So to improve this idea, we decide the pipeline. The idea of pipeline is to divide a cycle in the single-cycle in different parts, and execute different instructions at the same time. This do improve the time efficiency, but cause some problems. In order to solve them, we should add some more parts to the CPU. J-type instructions are can not satisfies the current design, and if the data we needed is still executing in the instructions above, we may calling some data before they exist. So a hazard detection unit and a forwarding unit is needed here.

The result of our design is shown in the appendix shown below. From the project, we know how a cpu work, and how the history of CPU's development, which will helps us in the later study.

Appendix A. Code for Single-Cycle CPU

single_cycle.v

```
1  `include "alu.v"
2  `include "alu_control.v"
3  `include "control.v"
4  `include "im.v"
5  `include "mux2.v"
6  `include "registers.v"
7  `include "dm.v"
8  `include "sign_extend.v"
9  `include "pc.v"
10
11 module single_cycle (input clk);
12
13     wire        [31:0]  pcIn,
14                         pcOut,
15                         pcAdd4,
16                         instruction,
17                         jumpAddress,
18                         branchResult,
19                         addResult;
20
21     wire                branch;
22
23     wire                regdst,
24                         jump,
25                         brancheq,
26                         branchne,
27                         memread,
28                         memtoreg,
29                         memwrite,
30                         alusrc,
31                         regwrite;
32     wire        [1:0]   aluop;
33
34     wire        [4:0]   regWriteReg;
35     wire        [31:0]  regWriteData,
36                         regReadData1,
37                         regReadData2;
38
39     wire        [31:0]  signExtendOut,
40                         aluMainIn,
41                         aluMainResult;
42     wire        [3:0]   aluMainControl;
43     wire                aluMainZero;
44
```

```
45      wire        [31:0]  dmReadData;
46
47
48      assign pcAdd4 = pcOut + 4;
49      assign addResult = pcAdd4 + (signExtendOut << 2);
50      assign branch = (brancheq & aluMainZero) | (branchne &
        ↪  ~aluMainZero);
51      assign branchResult = (branch == 1'b0) ? pcAdd4 : addResult;
52      assign jumpAddress = {pcAdd4[31:28], instruction[25:0], 2'b0};
53      assign pcIn = (jump == 1'b0) ? branchResult : jumpAddress;
54
55      assign regWriteReg = (regdst == 1'b0) ? instruction[20:16] :
        ↪  instruction[15:11];
56      assign regWriteData = (memtoreg == 1'b0) ? aluMainResult :
        ↪  dmReadData;
57      assign aluMainIn = (alusrc == 1'b0) ? regReadData2 :
        ↪  signExtendOut;
58
59
60
61      PC pc(
62          .clk(clk),
63          .in(pcIn),
64          .out(pcOut)
65      );
66
67      InstructionMemory im(
68          .address(pcOut),
69          .instruction(instruction)
70      );
71
72      Control control(
73          .opcode(instruction[31:26]),
74          .regdst(regdst),
75          .jump(jump),
76          .brancheq(brancheq),
77          .branchne(branchne),
78          .memread(memread),
79          .memtoreg(memtoreg),
80          .memwrite(memwrite),
81          .alusrc(alusrc),
82          .regwrite(regwrite),
83          .aluop(aluop)
84      );
85
86      SignExtend signExtend(
87          .in(instruction[15:0]),
88          .out(signExtendOut)
```

```
89      );
90
91      ALUControl aluControl(
92          .funct(instruction[5:0]),
93          .op(aluop),
94          .control(aluMainControl)
95      );
96
97      Registers registers(
98          .clk(clk),
99          .readReg1(instruction[25:21]),
100         .readReg2(instruction[20:16]),
101         .readData1(regReadData1),
102         .readData2(regReadData2),
103         .writeReg(regWriteReg),
104         .writeData(regWriteData),
105         .regWrite(regwrite)
106     );
107
108     ALU aluMain(
109         .a(regReadData1),
110         .b(aluMainIn),
111         .control(aluMainControl),
112         .zero(aluMainZero),
113         .result(aluMainResult)
114     );
115
116     DataMemory dm(
117         .clk(clk),
118         .address(aluMainResult),
119         .writeData(regReadData2),
120         .readData(dmReadData),
121         .memWrite(memwrite),
122         .memRead(memread)
123     );
124
125 endmodule // single_cycle
```

alu.v

```
1  `ifndef MODULE_ALU
2  `define MODULE_ALU
3  `timescale 1ns / 1ps
4
5  module ALU (
6      input       [3:0]   control,
7      input       [31:0]  a, b,
8      output              zero,
```

```
    output reg  [31:0]  result
);

    assign zero = (result == 0);

    initial begin
        result = 32'b0;
    end

    always @ ( control, a, b ) begin
        case (control)
            4'b0000: // AND
                result = a & b;
            4'b0001: // OR
                result = a | b;
            4'b0010: // ADD
                result = a + b;
            4'b0110: // SUB
                result = a - b;
            4'b0111: // SLT
                result = (a < b) ? 1 : 0;
            4'b1100: // NOR
                result = ~(a | b);
            default: ;
        endcase
    end

endmodule // ALU

`endif // MODULE_ALU
```

alu_control.v

```
`ifndef MODULE_ALU_CONTROL
`define MODULE_ALU_CONTROL
`timescale 1ns / 1ps

module ALUControl (
    input       [5:0]   funct,
    input       [1:0]   op,
    output reg  [3:0]   control
);

    always @ ( funct, op ) begin
        case (op)
            2'b00: // ADD
                control = 4'b0010;
            2'b01: // SUB
```

```
                control = 4'b0110;
            2'b10: // R-type
                case (funct)
                    6'b100000: // ADD
                        control = 4'b0010;
                    6'b100010: // SUB
                        control = 4'b0110;
                    6'b100100: // AND
                        control = 4'b0000;
                    6'b100101: // OR
                        control = 4'b0001;
                    6'b101010: // SLT
                        control = 4'b0111;
                    default: ;
                endcase
            2'b11: // AND
                control = 4'b0000;
            default: ;
        endcase
    end

endmodule // ALUControl

`endif // MODULE_ALU_CONTROL
```

control.v

```
`ifndef MODULE_CONTROL
`define MODULE_CONTROL
`timescale 1ns / 1ps

module Control (
    input       [5:0]   opCode,
    output reg          regDst,
                        jump,
                        branchEq,
                        branchNe,
                        memRead,
                        memtoReg,
                        memWrite,
                        aluSrc,
                        regWrite,
    output reg  [1:0]   aluOp
);

    initial begin
        regDst      = 1'b0;
        jump        = 1'b0;
```

```
22          branchEq    = 1'b0;
23          branchNe    = 1'b0;
24          memRead     = 1'b0;
25          memtoReg    = 1'b0;
26          memWrite    = 1'b0;
27          aluSrc      = 1'b0;
28          regWrite    = 1'b0;
29          aluOp       = 2'b00;
30      end
31
32      always @ ( opCode ) begin
33          case (opCode)
34              6'b000000: begin // R-type
35                  regDst      <= 1'b1;
36                  jump        <= 1'b0;
37                  branchEq    <= 1'b0;
38                  branchNe    <= 1'b0;
39                  memRead     <= 1'b0;
40                  memtoReg    <= 1'b0;
41                  memWrite    <= 1'b0;
42                  aluSrc      <= 1'b0;
43                  regWrite    <= 1'b1;
44                  aluOp       <= 2'b10;
45              end
46              6'b000010: begin // j
47                  regDst      <= 1'b1;
48                  jump        <= 1'b1;
49                  branchEq    <= 1'b0;
50                  branchNe    <= 1'b0;
51                  memRead     <= 1'b0;
52                  memtoReg    <= 1'b0;
53                  memWrite    <= 1'b0;
54                  aluSrc      <= 1'b0;
55                  regWrite    <= 1'b0;
56                  aluOp       <= 2'b10;
57              end
58              6'b000100: begin // beq
59                  regDst      <= 1'b1;
60                  jump        <= 1'b0;
61                  branchEq    <= 1'b1;
62                  branchNe    <= 1'b0;
63                  memRead     <= 1'b0;
64                  memtoReg    <= 1'b0;
65                  memWrite    <= 1'b0;
66                  aluSrc      <= 1'b0;
67                  regWrite    <= 1'b0;
68                  aluOp       <= 2'b01;
69              end
```

```
70              6'b000100: begin // bne
71                  regDst      <= 1'b1;
72                  jump        <= 1'b0;
73                  branchEq    <= 1'b0;
74                  branchNe    <= 1'b1;
75                  memRead     <= 1'b0;
76                  memtoReg    <= 1'b0;
77                  memWrite    <= 1'b0;
78                  aluSrc      <= 1'b0;
79                  regWrite    <= 1'b0;
80                  aluOp       <= 2'b01;
81              end
82              6'b001000: begin // addi
83                  regDst      <= 1'b0;
84                  jump        <= 1'b0;
85                  branchEq    <= 1'b0;
86                  branchNe    <= 1'b0;
87                  memRead     <= 1'b0;
88                  memtoReg    <= 1'b0;
89                  memWrite    <= 1'b0;
90                  aluSrc      <= 1'b1;
91                  regWrite    <= 1'b1;
92                  aluOp       <= 2'b00;
93              end
94              6'b001100: begin // andi
95                  regDst      <= 1'b0;
96                  jump        <= 1'b0;
97                  branchEq    <= 1'b0;
98                  branchNe    <= 1'b0;
99                  memRead     <= 1'b0;
100                 memtoReg    <= 1'b0;
101                 memWrite    <= 1'b0;
102                 aluSrc      <= 1'b1;
103                 regWrite    <= 1'b1;
104                 aluOp       <= 2'b11;
105             end
106             6'b100011: begin // lw
107                 regDst      <= 1'b0;
108                 jump        <= 1'b0;
109                 branchEq    <= 1'b0;
110                 branchNe    <= 1'b0;
111                 memRead     <= 1'b1;
112                 memtoReg    <= 1'b1;
113                 memWrite    <= 1'b0;
114                 aluSrc      <= 1'b1;
115                 regWrite    <= 1'b1;
116                 aluOp       <= 2'b00;
117             end
```

```
118             6'b101011: begin // sw
119                 regDst      <= 1'b0;
120                 jump        <= 1'b0;
121                 branchEq    <= 1'b0;
122                 branchNe    <= 1'b0;
123                 memRead     <= 1'b0;
124                 memtoReg    <= 1'b0;
125                 memWrite    <= 1'b1;
126                 aluSrc      <= 1'b1;
127                 regWrite    <= 1'b0;
128                 aluOp       <= 2'b00;
129             end
130 
131             default: ;
132         endcase
133     end
134 
135 endmodule // control
136 
137 `endif // MODULE_CONTROL
```

im.v

```
1  `ifndef MODULE_IM
2  `define MODULE_IM
3  
4  module InstructionMemory (
5      input       [31:0]  address,
6      output      [31:0]  instruction
7  );
8  
9      parameter size = 64;
10     integer i;
11 
12     reg [31:0] memory [0:size-1];
13 
14     initial begin
15         memory[0]  = 32'b00100000000010000000000000100000; //addi $t0,
           ↪ $zero, 0x20
16 memory[1]  = 32'b00100000000010010000000000100111; //addi $t1, $zero,
   ↪ 0x27
17 memory[2]  = 32'b00000001000010011100000000100100; //and $s0, $t0,
   ↪ $t1
18 memory[3]  = 32'b00000001000010011100000000100101; //or $s0, $t0, $t1
19 memory[4]  = 32'b10101100000010000000000000000100; //sw $s0, 4($zero)
20 memory[5]  = 32'b10101100000010000000000000001000; //sw $t0, 8($zero)
21 memory[6]  = 32'b00000001000010011000100000100000; //add $s1, $t0,
   ↪ $t1
```

```
22  memory[7]  = 32'b00000001000010011001000000100010; //sub $s2, $t0,
    ↪  $t1
23  memory[8]  = 32'b00010010001100100000000000001001; //beq $s1, $s2,
    ↪  error0
24  memory[9]  = 32'b10001100000100010000000000000100; //lw $s1, 4($zero)
25  memory[10] = 32'b00110010001100100000000000011000; //andi $s2, $s1,
    ↪  0x18
26  memory[11] = 32'b00010010001100100000000000001001; //beq $s1, $s2,
    ↪  error1
27  memory[12] = 32'b10001100000100110000000000001000; //lw $s3, 8($zero)
28  memory[13] = 32'b00010010000100110000000000001010; //beq $s0, $s3,
    ↪  error2
29  memory[14] = 32'b00000010010100011010000000101010; //slt $s4, $s2, $s1
    ↪  (Last)
30  memory[15] = 32'b00010010100000000000000000001111; //beq $s4, $0,
    ↪  EXIT
31  memory[16] = 32'b00000010001000001001000000100000; //add $s2, $s1, $0
32  memory[17] = 32'b00001000000000000000000000001110; //j Last
33  memory[18] = 32'b00100000000010000000000000000000; //addi $t0, $0,
    ↪  0(error0)
34  memory[19] = 32'b00100000000010010000000000000000; //addi $t1, $0, 0
35  memory[20] = 32'b00001000000000000000000000011111; //j EXIT
36  memory[21] = 32'b00100000000010000000000000000001; //addi $t0, $0,
    ↪  1(error1)
37  memory[22] = 32'b00100000000010010000000000000001; //addi $t1, $0, 1
38  memory[23] = 32'b00001000000000000000000000011111; //j EXIT
39  memory[24] = 32'b00100000000010000000000000000010; //addi $t0, $0,
    ↪  2(error2)
40  memory[25] = 32'b00100000000010010000000000000010; //addi $t1, $0, 2
41  memory[26] = 32'b00001000000000000000000000011111; //j EXIT
42  memory[27] = 32'b00100000000010000000000000000011; //addi $t0, $0,
    ↪  3(error3)
43  memory[28] = 32'b00100000000010010000000000000011; //addi $t1, $0, 3
44  memory[29] = 32'b00001000000000000000000000011111; //j EXIT
45
46      end
47
48      assign instruction = memory[address >> 2];
49
50  endmodule // InstructionMemory
51
52  `endif // MODULE_IM
```

mux2.v

```
1  `ifndef MODULE_MUX2
2  `define MODULE_MUX2
3
```

```
module Mux2 (in0, in1, sel, out);

    parameter   size = 32;
    input                   sel;
    input       [size-1:0]  in0, in1;
    output      [size-1:0]  out;

    assign out = (sel == 1'b0) ? in0 : in1;

endmodule // Mux2

`endif // MODULE_MUX2
```

registers.v

```
`ifndef MODULE_REGISTERS
`define MODULE_REGISTERS
`timescale 1ns / 1ps

module Registers (
    input               clk, regWrite,
    input       [4:0]   readReg1, readReg2, readRegExtra,
    input       [4:0]   writeReg,
    output      [31:0]  readData1, readData2, readDataExtra,
    input       [31:0]  writeData
);

    reg [31:0] regs [0:31];
    integer i;

    initial begin
        for (i = 0; i < 32; i = i + 1)
            regs[i] = 32'b0;
    end

    assign readData1 = regs[readReg1];
    assign readData2 = regs[readReg2];
    assign readDataExtra = regs[readRegExtra];

    always @ (negedge clk) begin
        if (regWrite == 1)
            regs[writeReg] <= writeData;
    end

endmodule // registers

`endif
```

dm.v

```
`ifndef MODULE_DM
`define MODULE_DM
`timescale 1ns / 1ps

module DataMemory (
    input               clk,
    input       [31:0]  address,
                        writeData,
    input               memRead,
                        memWrite,
    output      [31:0]  readData
);

    wire        [31:0]  index;
    parameter           size = 8;
    integer             i;
    reg         [31:0]  memory [0:size-1];

    assign index = address >> 2;


    initial begin
        for (i = 0; i < size; i = i + 1)
            memory[i] = 32'b0;
        //readData = 32'b0;
    end

    always @ ( posedge clk ) begin
        if (memWrite == 1'b1) begin
            memory[index] = writeData;
        end
    end

    assign readData = (memRead == 1'b1) ? memory[index] : 32'b0;

endmodule // DataMemory


`endif
```

sign_extend.v

```
`ifndef MODULE_SIGN_EXTEND
`define MODULE_SIGN_EXTEND
`timescale 1ns / 1ps

module SignExtend (
```

```
    input           [15:0]  in,
    output          [31:0]  out
);

    assign out = {{16{in[15]}}, in[15:0]};

endmodule // SignExtend

`endif
```

pc.v

```
`ifndef MODULE_PC
`define MODULE_PC
`timescale 1ns / 1ps

module PC (
    input               clk,
                        stall,
    input       [31:0]  in,
    output reg  [31:0]  out
);

    initial begin
        out = 32'b0;
    end

    always @ (posedge clk) begin
        if (!stall)
            out <= in;
    end

endmodule // PC

`endif
```

single_cycle_tb.v

```
`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:   00:39:53 11/20/2017
// Design Name:   single_cycle
// Module Name:   /home/liu/VE370/p2/single_cycle_tb.v
// Project Name:  p2
```

```
11 // Target Device:
12 // Tool versions:
13 // Description:
14 //
15 // Verilog Test Fixture created by ISE for module: single_cycle
16 //
17 // Dependencies:
18 //
19 // Revision:
20 // Revision 0.01 - File Created
21 // Additional Comments:
22 //
23 ////////////////////////////////////////////////////////////////////////////////
24
25 `include "single_cycle.v"
26
27 module single_cycle_tb;
28
29     integer i = 0;
30     // Inputs
31      reg clk;
32
33     // Instantiate the Unit Under Test (UUT)
34     single_cycle uut (
35         .clk(clk)
36     );
37
38     initial begin
39         // Initialize Inputs
40         clk = 0;
41     end
42
43     always #20 begin
44         $display("Time: %d, CLK = %d, PC = 0x%H", i, clk, uut.pcOut);
45         $display("[$s0] = 0x%H, [$s1] = 0x%H, [$s2] = 0x%H",
           ↪  uut.registers.regs[16], uut.registers.regs[17],
           ↪  uut.registers.regs[18]);
46         $display("[$s3] = 0x%H, [$s4] = 0x%H, [$s5] = 0x%H",
           ↪  uut.registers.regs[19], uut.registers.regs[20],
           ↪  uut.registers.regs[21]);
47         $display("[$s6] = 0x%H, [$s7] = 0x%H, [$t0] = 0x%H",
           ↪  uut.registers.regs[22], uut.registers.regs[23],
           ↪  uut.registers.regs[8]);
48         $display("[$t1] = 0x%H, [$t2] = 0x%H, [$t3] = 0x%H",
           ↪  uut.registers.regs[9], uut.registers.regs[10],
           ↪  uut.registers.regs[11]);
```

```
49              $display("[$t4] = 0x%H, [$t5] = 0x%H, [$t6] = 0x%H",
                ↪  uut.registers.regs[12], uut.registers.regs[13],
                ↪  uut.registers.regs[14]);
50              $display("[$t7] = 0x%H, [$t8] = 0x%H, [$t9] = 0x%H",
                ↪  uut.registers.regs[15], uut.registers.regs[24],
                ↪  uut.registers.regs[25]);
51
             ↪  $display("-------------------------------------------------------");
52          clk = ~clk;
53          if (~clk) i = i + 1;
54      end
55  endmodule
```

Appendix B. Code for Pipelined CPU

pipeline.v

```
1   `ifndef MODULE_PIPELINE
2   `define MODULE_PIPELINE
3   `timescale 1ns / 1ps
4
5   `include "alu.v"
6   `include "alu_control.v"
7   `include "control.v"
8   `include "im.v"
9   `include "mux2.v"
10  `include "registers.v"
11  `include "dm.v"
12  `include "sign_extend.v"
13  `include "pc.v"
14  `include "if_id.v"
15  `include "id_ex.v"
16  `include "ex_mem.v"
17  `include "mem_wb.v"
18  `include "forward.v"
19  `include "hazard_detection.v"
20
21  module Pipeline (
22      input               clk,
23      input       [4:0]   regIn,
24      output      [31:0]  pcOut,
25                          regOut
26  );
27
28      wire                flushIFID,
29                          flushIDEX,
30                          stall;
31
```

```
32      // IF stage
33      wire        [31:0]  pcInIF,
34                          pcOutIF,
35                          pcAdd4IF,
36                          instructionIF,
37                          branchResultIF;
38
39      // ID stage
40      wire        [31:0]  pcAdd4ID,
41                          pcAddResultID,
42                          instructionID,
43                          regReadData1ID,
44                          regReadData2ID,
45                          regReadData1NewID,
46                          regReadData2NewID,
47                          signExtendID,
48                          jumpAddressID;
49      wire        [4:0]   registerRsID,
50                          registerRtID,
51                          registerRdID;
52      wire        [1:0]   aluOpID;
53      wire                regDstID,
54                          jumpID,
55                          branchEqID,
56                          branchNeID,
57                          memReadID,
58                          memtoRegID,
59                          memWriteID,
60                          aluSrcID,
61                          regWriteID,
62                          branchID,
63                          regReadDataEqID;
64
65      // EX stage
66      wire        [31:0]  regReadData1EX,
67                          regReadData2EX,
68                          signExtendEX,
69                          aluInAEX,
70                          aluInTempBEX,
71                          aluInBEX,
72                          aluResultEX;
73      wire        [4:0]   registerRsEX,
74                          registerRtEX,
75                          registerRdEX,
76                          registerEX;
77      wire        [3:0]   aluControlEX;
78      wire        [1:0]   aluOpEX;
79      wire                regDstEX,
```

```
80                      memReadEX,
81                      memtoRegEX,
82                      memWriteEX,
83                      aluSrcEX,
84                      regWriteEX,
85                      aluZeroEX;
86
87      // MEM stage
88      wire        [31:0]  aluResultMEM,
89                          regReadData2MEM,
90                          dmReadDataMEM;
91      wire        [4:0]   registerMEM;
92      wire                memReadMEM,
93                          memtoRegMEM,
94                          memWriteMEM,
95                          regWriteMEM;
96      // WB stage
97      wire        [31:0]  aluResultWB,
98                          dmReadDataWB,
99                          regWriteDataWB;
100     wire        [4:0]   registerWB;
101     wire                memtoRegWB,
102                         regWriteWB;
103
104     // Data Hazard
105     wire        [1:0]   forwardA,
106                         forwardB;
107     wire                forwardC,
108                         forwardD;
109
110     // IF stage
111     PC pc(
112         .clk(clk), .stall(stall),
113         .in(pcInIF), .out(pcOutIF)
114     );
115     InstructionMemory im(.address(pcOutIF),
        ↪  .instruction(instructionIF));
116     assign pcAdd4IF = pcOutIF + 4;
117
118     // IF/ID
119     IF_ID ifid (
120         .clk(clk), .stall(stall), .flush(flushIFID),
121         .pcAdd4IF(pcAdd4IF), .pcAdd4ID(pcAdd4ID),
122         .instructionIF(instructionIF), .instructionID(instructionID)
123     );
124
125     // ID stage
126     assign registerRsID = instructionID[25:21];
```

```
127     assign registerRtID = instructionID[20:16];
128     assign registerRdID = instructionID[15:11];
129     Control control (
130         .opCode(instructionID[31:26]),
131         .regDst(regDstID),
132         .jump(jumpID),
133         .branchEq(branchEqID),
134         .branchNe(branchNeID),
135         .memRead(memReadID),
136         .memtoReg(memtoRegID),
137         .memWrite(memWriteID),
138         .aluSrc(aluSrcID),
139         .regWrite(regWriteID),
140         .aluOp(aluOpID)
141     );
142     Registers registers (
143         .clk(clk),
144         .readReg1(registerRsID),
145         .readReg2(registerRtID),
146         .readData1(regReadData1ID),
147         .readData2(regReadData2ID),
148         .writeReg(registerWB),
149         .writeData(regWriteDataWB),
150         .regWrite(regWriteWB),
151         .readRegExtra(regIn),
152         .readDataExtra(regOut)
153     );
154     SignExtend signExtend(.in(instructionID[15:0]),
        ↪  .out(signExtendID));
155     assign pcAddResultID = pcAdd4ID + (signExtendID << 2);
156     assign regReadData1NewID = (forwardC) ? aluResultMEM :
        ↪  regReadData1ID;
157     assign regReadData2NewID = (forwardD) ? aluResultMEM :
        ↪  regReadData2ID;
158     assign regReadDataEqID = (regReadData1NewID == regReadData2NewID);
159     assign branchID = (branchEqID && regReadDataEqID) || (branchNeID
        ↪  && !regReadDataEqID);
160     assign branchResultIF = (!branchID) ? pcAdd4IF : pcAddResultID;
161     assign jumpAddressID = {pcAdd4ID[31:28], instructionID[25:0],
        ↪  2'b0};
162     assign pcInIF = (!jumpID) ? branchResultIF : jumpAddressID;
163     assign flushIFID = branchID;
164
165     // ID/EX
166     ID_EX idex (
167         .clk(clk), .flush(flushIDEX),
168         .regReadData1ID(regReadData1ID),
            ↪  .regReadData1EX(regReadData1EX),
```

```
169         .regReadData2ID(regReadData2ID),
            ↪  .regReadData2EX(regReadData2EX),
170         .signExtendID(signExtendID), .signExtendEX(signExtendEX),
171         .registerRsID(registerRsID), .registerRsEX(registerRsEX),
172         .registerRtID(registerRtID), .registerRtEX(registerRtEX),
173         .registerRdID(registerRdID), .registerRdEX(registerRdEX),
174         .aluOpID(aluOpID), .aluOpEX(aluOpEX),
175         .regDstID(regDstID), .regDstEX(regDstEX),
176         .memReadID(memReadID), .memReadEX(memReadEX),
177         .memtoRegID(memtoRegID), .memtoRegEX(memtoRegEX),
178         .memWriteID(memWriteID), .memWriteEX(memWriteEX),
179         .aluSrcID(aluSrcID), .aluSrcEX(aluSrcEX),
180         .regWriteID(regWriteID), .regWriteEX(regWriteEX)
181     );
182
183     // EX stage
184     ALUControl aluControl (
185         .funct(signExtendEX[5:0]),
186         .op(aluOpEX),
187         .control(aluControlEX)
188     );
189     assign aluInAEX = (forwardA == 2'b00) ? regReadData1EX :
190         ((forwardA == 2'b01) ? regWriteDataWB : aluResultMEM);
191     assign aluInTempBEX = (forwardB == 2'b00) ? regReadData2EX :
192         ((forwardB == 2'b01) ? regWriteDataWB : aluResultMEM);
193     assign aluInBEX = (!aluSrcEX) ? aluInTempBEX : signExtendEX;
194     ALU alu (
195         .a(aluInAEX),
196         .b(aluInBEX),
197         .control(aluControlEX),
198         .zero(aluZeroEX),
199         .result(aluResultEX)
200     );
201     assign registerEX = (!regDstEX) ? registerRtEX : registerRdEX;
202
203     // EX/MEM
204     EX_MEM exmem (
205         .clk(clk),
206         .aluResultEX(aluResultEX), .aluResultMEM(aluResultMEM),
207         .regReadData2EX(aluInTempBEX),
            ↪  .regReadData2MEM(regReadData2MEM),
208         .registerEX(registerEX), .registerMEM(registerMEM),
209         .memReadEX(memReadEX), .memReadMEM(memReadMEM),
210         .memtoRegEX(memtoRegEX), .memtoRegMEM(memtoRegMEM),
211         .memWriteEX(memWriteEX), .memWriteMEM(memWriteMEM),
212         .regWriteEX(regWriteEX), .regWriteMEM(regWriteMEM)
213     );
214
```

```
215     // MEM stage
216     DataMemory dm (
217         .clk(clk),
218         .address(aluResultMEM),
219         .writeData(regReadData2MEM),
220         .readData(dmReadDataMEM),
221         .memWrite(memWriteMEM),
222         .memRead(memReadMEM)
223     );
224 
225     // MEM/WB
226     MEM_WB memwb (
227         .clk(clk),
228         .dmReadDataMEM(dmReadDataMEM), .dmReadDataWB(dmReadDataWB),
229         .aluResultMEM(aluResultMEM), .aluResultWB(aluResultWB),
230         .registerMEM(registerMEM), .registerWB(registerWB),
231         .memtoRegMEM(memtoRegMEM), .memtoRegWB(memtoRegWB),
232         .regWriteMEM(regWriteMEM), .regWriteWB(regWriteWB)
233     );
234 
235     // WB stage
236     assign regWriteDataWB = (!memtoRegWB) ? aluResultWB :
        ↪  dmReadDataWB;
237     //assign regWriteDataWB = aluResultWB;
238 
239     // Data Hazard
240     Forward forward (
241         .registerRsID(registerRsID),
242         .registerRtID(registerRtID),
243         .registerRsEX(registerRsEX),
244         .registerRtEX(registerRtEX),
245         .registerRdMEM(registerMEM),
246         .registerRdWB(registerWB),
247         .regWriteMEM(regWriteMEM),
248         .regWriteWB(regWriteWB),
249         .forwardA(forwardA),
250         .forwardB(forwardB),
251         .forwardC(forwardC),
252         .forwardD(forwardD)
253     );
254 
255     HazardDetection hazardDetection (
256         .branchEqID(branchEqID),
257         .branchNeID(branchNeID),
258         .memReadEX(memReadEX),
259         .regWriteEX(regWriteEX),
260         .memReadMEM(memReadMEM),
261         .registerRsID(registerRsID),
```

```
262         .registerRtID(registerRtID),
263         .registerRtEX(registerRtEX),
264         .registerRdEX(registerEX),
265         .registerRdMEM(registerMEM),
266         .stall(stall),
267         .flush(flushIDEX)
268     );
269
270     // Output
271     assign pcOut = pcOutIF;
272
273
274 endmodule // pipeline
275
276 `endif
```

alu.v

```
1  `ifndef MODULE_ALU
2  `define MODULE_ALU
3  `timescale 1ns / 1ps
4
5  module ALU (
6      input       [3:0]   control,
7      input       [31:0]  a, b,
8      output              zero,
9      output reg  [31:0]  result
10 );
11
12     assign zero = (result == 0);
13
14     initial begin
15         result = 32'b0;
16     end
17
18     always @ ( control, a, b ) begin
19         case (control)
20             4'b0000: // AND
21                 result = a & b;
22             4'b0001: // OR
23                 result = a | b;
24             4'b0010: // ADD
25                 result = a + b;
26             4'b0110: // SUB
27                 result = a - b;
28             4'b0111: // SLT
29                 result = (a < b) ? 1 : 0;
30             4'b1100: // NOR
```

```
                result = ~(a | b);
            default: ;
        endcase
    end

endmodule // ALU

`endif // MODULE_ALU
```

alu_control.v

```
`ifndef MODULE_ALU_CONTROL
`define MODULE_ALU_CONTROL
`timescale 1ns / 1ps

module ALUControl (
    input       [5:0]   funct,
    input       [1:0]   op,
    output reg  [3:0]   control
);

    always @ ( funct, op ) begin
        case (op)
            2'b00: // ADD
                control = 4'b0010;
            2'b01: // SUB
                control = 4'b0110;
            2'b10: // R-type
                case (funct)
                    6'b100000: // ADD
                        control = 4'b0010;
                    6'b100010: // SUB
                        control = 4'b0110;
                    6'b100100: // AND
                        control = 4'b0000;
                    6'b100101: // OR
                        control = 4'b0001;
                    6'b101010: // SLT
                        control = 4'b0111;
                    default: ;
                endcase
            2'b11: // AND
                control = 4'b0000;
            default: ;
        endcase
    end

endmodule // ALUControl
```

```
38
39 `endif // MODULE_ALU_CONTROL
```

control.v

```
1  `ifndef MODULE_CONTROL
2  `define MODULE_CONTROL
3  `timescale 1ns / 1ps
4
5  module Control (
6      input       [5:0]   opCode,
7      output reg          regDst,
8                          jump,
9                          branchEq,
10                         branchNe,
11                         memRead,
12                         memtoReg,
13                         memWrite,
14                         aluSrc,
15                         regWrite,
16     output reg  [1:0]   aluOp
17 );
18
19     initial begin
20         regDst      = 1'b0;
21         jump        = 1'b0;
22         branchEq    = 1'b0;
23         branchNe    = 1'b0;
24         memRead     = 1'b0;
25         memtoReg    = 1'b0;
26         memWrite    = 1'b0;
27         aluSrc      = 1'b0;
28         regWrite    = 1'b0;
29         aluOp       = 2'b00;
30     end
31
32     always @ ( opCode ) begin
33         case (opCode)
34             6'b000000: begin // R-type
35                 regDst      <= 1'b1;
36                 jump        <= 1'b0;
37                 branchEq    <= 1'b0;
38                 branchNe    <= 1'b0;
39                 memRead     <= 1'b0;
40                 memtoReg    <= 1'b0;
41                 memWrite    <= 1'b0;
42                 aluSrc      <= 1'b0;
43                 regWrite    <= 1'b1;
```

```
44                  aluOp       <= 2'b10;
45              end
46              6'b000010: begin // j
47                  regDst      <= 1'b1;
48                  jump        <= 1'b1;
49                  branchEq    <= 1'b0;
50                  branchNe    <= 1'b0;
51                  memRead     <= 1'b0;
52                  memtoReg    <= 1'b0;
53                  memWrite    <= 1'b0;
54                  aluSrc      <= 1'b0;
55                  regWrite    <= 1'b0;
56                  aluOp       <= 2'b10;
57              end
58              6'b000100: begin // beq
59                  regDst      <= 1'b1;
60                  jump        <= 1'b0;
61                  branchEq    <= 1'b1;
62                  branchNe    <= 1'b0;
63                  memRead     <= 1'b0;
64                  memtoReg    <= 1'b0;
65                  memWrite    <= 1'b0;
66                  aluSrc      <= 1'b0;
67                  regWrite    <= 1'b0;
68                  aluOp       <= 2'b01;
69              end
70              6'b000100: begin // bne
71                  regDst      <= 1'b1;
72                  jump        <= 1'b0;
73                  branchEq    <= 1'b0;
74                  branchNe    <= 1'b1;
75                  memRead     <= 1'b0;
76                  memtoReg    <= 1'b0;
77                  memWrite    <= 1'b0;
78                  aluSrc      <= 1'b0;
79                  regWrite    <= 1'b0;
80                  aluOp       <= 2'b01;
81              end
82              6'b001000: begin // addi
83                  regDst      <= 1'b0;
84                  jump        <= 1'b0;
85                  branchEq    <= 1'b0;
86                  branchNe    <= 1'b0;
87                  memRead     <= 1'b0;
88                  memtoReg    <= 1'b0;
89                  memWrite    <= 1'b0;
90                  aluSrc      <= 1'b1;
91                  regWrite    <= 1'b1;
```

```
92                  aluOp       <= 2'b00;
93              end
94              6'b001100: begin // andi
95                  regDst      <= 1'b0;
96                  jump        <= 1'b0;
97                  branchEq    <= 1'b0;
98                  branchNe    <= 1'b0;
99                  memRead     <= 1'b0;
100                 memtoReg    <= 1'b0;
101                 memWrite    <= 1'b0;
102                 aluSrc      <= 1'b1;
103                 regWrite    <= 1'b1;
104                 aluOp       <= 2'b11;
105             end
106             6'b100011: begin // lw
107                 regDst      <= 1'b0;
108                 jump        <= 1'b0;
109                 branchEq    <= 1'b0;
110                 branchNe    <= 1'b0;
111                 memRead     <= 1'b1;
112                 memtoReg    <= 1'b1;
113                 memWrite    <= 1'b0;
114                 aluSrc      <= 1'b1;
115                 regWrite    <= 1'b1;
116                 aluOp       <= 2'b00;
117             end
118             6'b101011: begin // sw
119                 regDst      <= 1'b0;
120                 jump        <= 1'b0;
121                 branchEq    <= 1'b0;
122                 branchNe    <= 1'b0;
123                 memRead     <= 1'b0;
124                 memtoReg    <= 1'b0;
125                 memWrite    <= 1'b1;
126                 aluSrc      <= 1'b1;
127                 regWrite    <= 1'b0;
128                 aluOp       <= 2'b00;
129             end
130
131             default: ;
132         endcase
133     end
134
135 endmodule // control
136
137 `endif // MODULE_CONTROL
```

im.v

```
`ifndef MODULE_IM
`define MODULE_IM

module InstructionMemory (
    input       [31:0]  address,
    output      [31:0]  instruction
);

    parameter size = 64;
    integer i;

    reg [31:0] memory [0:size-1];

    initial begin
        memory[0]  = 32'b00100000000010000000000000100000; //addi $t0, $zero, 0x20
memory[1]  = 32'b00100000000010010000000000100111; //addi $t1, $zero, 0x27
memory[2]  = 32'b00000001000010011100000000100100; //and $s0, $t0, $t1
memory[3]  = 32'b00000001000010011100000000100101; //or $s0, $t0, $t1
memory[4]  = 32'b10101100000010000000000000000100; //sw $s0, 4($zero)
memory[5]  = 32'b10101100000010000000000000001000; //sw $t0, 8($zero)
memory[6]  = 32'b00000001000010011100100000100000; //add $s1, $t0, $t1
memory[7]  = 32'b00000001000010011001000000100010; //sub $s2, $t0, $t1
memory[8]  = 32'b00010010001100100000000000001001; //beq $s1, $s2, error0
memory[9]  = 32'b10001100000010001000000000000100; //lw $s1, 4($zero)
memory[10] = 32'b00110010001100100000000000011000; //andi $s2, $s1, 0x18
memory[11] = 32'b00010010001100100000000000001001; //beq $s1, $s2, error1
memory[12] = 32'b10001100000010011000000000001000; //lw $s3, 8($zero)
memory[13] = 32'b00010010000010011000000000001010; //beq $s0, $s3, error2
memory[14] = 32'b00000001001010001101000000101010; //slt $s4, $s2, $s1 (Last)
memory[15] = 32'b00010010100000000000000000001111; //beq $s4, $0, EXIT
memory[16] = 32'b00000001000100000100100000100000; //add $s2, $s1, $0
memory[17] = 32'b00001000000000000000000000001110; //j Last
memory[18] = 32'b00100000000010000000000000000000; //addi $t0, $0, 0(error0)
memory[19] = 32'b00100000000010010000000000000000; //addi $t1, $0, 0
```

```
35  memory[20] = 32'b00001000000000000000000000011111; //j EXIT
36  memory[21] = 32'b00100000000010000000000000000001; //addi $t0, $0,
    ↪  1(error1)
37  memory[22] = 32'b00100000000010010000000000000001; //addi $t1, $0, 1
38  memory[23] = 32'b00001000000000000000000000011111; //j EXIT
39  memory[24] = 32'b00100000000010000000000000000010; //addi $t0, $0,
    ↪  2(error2)
40  memory[25] = 32'b00100000000010010000000000000010; //addi $t1, $0, 2
41  memory[26] = 32'b00001000000000000000000000011111; //j EXIT
42  memory[27] = 32'b00100000000010000000000000000011; //addi $t0, $0,
    ↪  3(error3)
43  memory[28] = 32'b00100000000010010000000000000011; //addi $t1, $0, 3
44  memory[29] = 32'b00001000000000000000000000011111; //j EXIT
45
46      end
47
48      assign instruction = memory[address >> 2];
49
50  endmodule // InstructionMemory
51
52  `endif // MODULE_IM
```

mux2.v

```
1   `ifndef MODULE_MUX2
2   `define MODULE_MUX2
3
4   module Mux2 (in0, in1, sel, out);
5
6       parameter   size = 32;
7       input                   sel;
8       input       [size-1:0]  in0, in1;
9       output      [size-1:0]  out;
10
11      assign out = (sel == 1'b0) ? in0 : in1;
12
13  endmodule // Mux2
14
15  `endif // MODULE_MUX2
```

registers.v

```
1   `ifndef MODULE_REGISTERS
2   `define MODULE_REGISTERS
3   `timescale 1ns / 1ps
4
5   module Registers (
6       input               clk, regWrite,
```

```
    input       [4:0]   readReg1, readReg2, readRegExtra,
    input       [4:0]   writeReg,
    output      [31:0]  readData1, readData2, readDataExtra,
    input       [31:0]  writeData
);

    reg [31:0] regs [0:31];
    integer i;

    initial begin
        for (i = 0; i < 32; i = i + 1)
            regs[i] = 32'b0;
    end

    assign readData1 = regs[readReg1];
    assign readData2 = regs[readReg2];
    assign readDataExtra = regs[readRegExtra];

    always @ (negedge clk) begin
        if (regWrite == 1)
            regs[writeReg] <= writeData;
    end

endmodule // registers

`endif
```

dm.v

```
`ifndef MODULE_DM
`define MODULE_DM
`timescale 1ns / 1ps

module DataMemory (
    input               clk,
    input       [31:0]  address,
                        writeData,
    input               memRead,
                        memWrite,
    output      [31:0]  readData
);

    wire        [31:0]  index;
    parameter           size = 8;
    integer             i;
    reg         [31:0]  memory [0:size-1];

    assign index = address >> 2;
```

```
20
21
22      initial begin
23          for (i = 0; i < size; i = i + 1)
24              memory[i] = 32'b0;
25          //readData = 32'b0;
26      end
27
28      always @ ( posedge clk ) begin
29          if (memWrite == 1'b1) begin
30              memory[index] = writeData;
31          end
32      end
33
34      assign readData = (memRead == 1'b1) ? memory[index] : 32'b0;
35
36  endmodule // DataMemory
37
38
39  `endif
```

sign_extend.v

```
1   `ifndef MODULE_SIGN_EXTEND
2   `define MODULE_SIGN_EXTEND
3   `timescale 1ns / 1ps
4
5   module SignExtend (
6       input           [15:0]  in,
7       output          [31:0]  out
8   );
9
10      assign out = {{16{in[15]}}, in[15:0]};
11
12  endmodule // SignExtend
13
14  `endif
```

pc.v

```
1   `ifndef MODULE_PC
2   `define MODULE_PC
3   `timescale 1ns / 1ps
4
5   module PC (
6       input               clk,
7                           stall,
8       input       [31:0]  in,
```

```
    output reg  [31:0]  out
);

    initial begin
        out = 32'b0;
    end

    always @ (posedge clk) begin
        if (!stall)
            out <= in;
    end

endmodule // PC

`endif
```

ex_mem.v

```
`ifndef MODULE_EX_MEM
`define MODULE_EX_MEM
`timescale 1ns / 1ps

module EX_MEM (
    input               clk,

    input       [31:0]  aluResultEX,
                        regReadData2EX,
    input       [4:0]   registerEX,
    input               memReadEX,
                        memtoRegEX,
                        memWriteEX,
                        regWriteEX,

    output reg  [31:0]  aluResultMEM,
                        regReadData2MEM,
    output reg  [4:0]   registerMEM,
    output reg          memReadMEM,
                        memtoRegMEM,
                        memWriteMEM,
                        regWriteMEM
);

    initial begin
        aluResultMEM    = 32'b0;
        regReadData2MEM = 32'b0;
        registerMEM     = 5'b0;
        memReadMEM      = 1'b0;
        memtoRegMEM     = 1'b0;
```

```
31          memWriteMEM     = 1'b0;
32          regWriteMEM     = 1'b0;
33      end
34
35      always @ (posedge clk) begin
36          aluResultMEM    <= aluResultEX;
37          regReadData2MEM <= regReadData2EX;
38          registerMEM     <= registerEX;
39          memReadMEM      <= memReadEX;
40          memtoRegMEM     <= memtoRegEX;
41          memWriteMEM     <= memWriteEX;
42          regWriteMEM     <= regWriteEX;
43      end
44
45  endmodule // EX_MEM
46
47  `endif // MODULE_EX_MEM
```

forward.v

```
1   `ifndef MODULE_FORWARD
2   `define MODULE_FORWARD
3   `timescale 1ns / 1ps
4
5   module Forward (
6       input       [4:0]   registerRsID,
7                           registerRtID,
8                           registerRsEX,
9                           registerRtEX,
10                          registerRdMEM,
11                          registerRdWB,
12      input               regWriteMEM,
13                          regWriteWB,
14      output reg  [1:0]   forwardA,
15                          forwardB,
16      output reg          forwardC,
17                          forwardD
18  );
19
20      initial begin
21          forwardA = 2'b00;
22          forwardB = 2'b00;
23          forwardC = 1'b0;
24          forwardD = 1'b0;
25      end
26
27      always @ ( * ) begin
```

```
28          if (regWriteMEM && registerRdMEM && registerRdMEM ==
            ↪ registerRsEX)
29              forwardA = 2'b10;
30          else if (regWriteWB && registerRdWB && registerRdWB ==
            ↪ registerRsEX)
31              forwardA = 2'b01;
32          else
33              forwardA = 2'b00;
34
35          if (regWriteMEM && registerRdMEM && registerRdMEM ==
            ↪ registerRtEX)
36              forwardB = 2'b10;
37          else if (regWriteWB && registerRdWB && registerRdWB ==
            ↪ registerRtEX)
38              forwardB = 2'b01;
39          else
40              forwardB = 2'b00;
41
42          if (regWriteMEM && registerRdMEM && registerRdMEM ==
            ↪ registerRsID)
43              forwardC = 1'b1;
44          else
45              forwardC = 1'b0;
46
47          if (regWriteMEM && registerRdMEM && registerRdMEM ==
            ↪ registerRtID)
48              forwardD = 1'b1;
49          else
50              forwardD = 1'b0;
51      end
52
53  endmodule // Forward
54
55  `endif // MODULE_FORWARD
```

harzard_detection.v

```
1   `ifndef MODULE_HAZARD_DETECTION
2   `define MODULE_HAZARD_DETECTION
3   `timescale 1ns / 1ps
4
5   module HazardDetection (
6       input               branchEqID,
7                           branchNeID,
8                           memReadEX,
9                           regWriteEX,
10                          memReadMEM,
11      input       [4:0]   registerRsID,
```

```
                        registerRtID,
                        registerRtEX,
                        registerRdEX,
                        registerRdMEM,
    output reg          stall,
                        flush
);

    initial begin
        stall = 1'b0;
        flush = 1'b0;
    end

    always @ ( * ) begin
        if (memReadEX && registerRtEX && (registerRtEX == registerRsID
        ↪  || registerRtEX == registerRtID)) begin
            stall = 1'b1;
            flush = 1'b1;
        end else if (branchEqID || branchNeID) begin
            if (regWriteEX && registerRdEX && (registerRdEX ==
            ↪  registerRsID || registerRdEX == registerRtID)) begin
                stall = 1'b1;
                flush = 1'b1;
            end else if (memReadMEM && registerRdMEM && (registerRdMEM
            ↪  == registerRsID || registerRdMEM == registerRtID))
            ↪  begin
                stall = 1'b1;
                flush = 1'b1;
            end else begin
                stall = 1'b0;
                flush = 1'b0;
            end
        end else begin
            stall = 1'b0;
            flush = 1'b0;
        end
    end

endmodule // HazardDetection

`endif
```

id_ex.v

```
`ifndef MODULE_ID_EX
`define MODULE_ID_EX
`timescale 1ns / 1ps

```

```
5  module ID_EX (
6      input                clk,
7                           flush,
8
9      input        [31:0]  regReadData1ID,
10                          regReadData2ID,
11                          signExtendID,
12     input        [4:0]   registerRsID,
13                          registerRtID,
14                          registerRdID,
15     input        [1:0]   aluOpID,
16     input                regDstID,
17                          memReadID,
18                          memtoRegID,
19                          memWriteID,
20                          aluSrcID,
21                          regWriteID,
22
23     output reg   [31:0]  regReadData1EX,
24                          regReadData2EX,
25                          signExtendEX,
26     output reg   [4:0]   registerRsEX,
27                          registerRtEX,
28                          registerRdEX,
29     output reg   [1:0]   aluOpEX,
30     output reg           regDstEX,
31                          memReadEX,
32                          memtoRegEX,
33                          memWriteEX,
34                          aluSrcEX,
35                          regWriteEX
36 );
37
38     initial begin
39         regReadData1EX  = 32'b0;
40         regReadData2EX  = 32'b0;
41         signExtendEX    = 32'b0;
42         registerRsEX    = 5'b0;
43         registerRtEX    = 5'b0;
44         registerRdEX    = 5'b0;
45         aluOpEX         = 2'b0;
46         regDstEX        = 1'b0;
47         memReadEX       = 1'b0;
48         memtoRegEX      = 1'b0;
49         memWriteEX      = 1'b0;
50         aluSrcEX        = 1'b0;
51         regWriteEX      = 1'b0;
52     end
```

```
53
54      always @ (posedge clk) begin
55          if (flush) begin
56              aluOpEX         <= 2'b0;
57              regDstEX        <= 1'b0;
58              memReadEX       <= 1'b0;
59              memtoRegEX      <= 1'b0;
60              memWriteEX      <= 1'b0;
61              aluSrcEX        <= 1'b0;
62              regWriteEX      <= 1'b0;
63          end else begin
64              regReadData1EX  <= regReadData1ID;
65              regReadData2EX  <= regReadData2ID;
66              signExtendEX    <= signExtendID;
67              registerRsEX    <= registerRsID;
68              registerRtEX    <= registerRtID;
69              registerRdEX    <= registerRdID;
70              aluOpEX         <= aluOpID;
71              regDstEX        <= regDstID;
72              memReadEX       <= memReadID;
73              memtoRegEX      <= memtoRegID;
74              memWriteEX      <= memWriteID;
75              aluSrcEX        <= aluSrcID;
76              regWriteEX      <= regWriteID;
77          end
78      end
79
80  endmodule // ID_EX
81
82  `endif // MODULE_ID_EX
```

if_id.v

```
1   `ifndef MODULE_IF_ID
2   `define MODULE_IF_ID
3   `timescale 1ns / 1ps
4
5   module IF_ID (
6       input               clk,
7                           stall,
8                           flush,
9       input       [31:0]  pcAdd4IF,
10                          instructionIF,
11      output reg  [31:0]  pcAdd4ID,
12                          instructionID
13  );
14
15      initial begin
```

```
16          pcAdd4ID = 32'b0;
17          instructionID = 32'b0;
18      end
19
20      always @ (posedge clk) begin
21          if (flush) begin
22              pcAdd4ID <= 32'b0;
23              instructionID <= 32'b0;
24          end else if (!stall) begin
25              pcAdd4ID <= pcAdd4IF;
26              instructionID <= instructionIF;
27          end
28      end
29
30  endmodule // IF_ID
31
32
33  `endif // MODULE_IF_ID
```

mem_wb.v

```
1   `ifndef MODULE_MEM_WB
2   `define MODULE_MEM_WB
3   `timescale 1ns / 1ps
4
5   module MEM_WB (
6       input               clk,
7
8       input       [31:0]  dmReadDataMEM,
9                           aluResultMEM,
10      input       [4:0]   registerMEM,
11      input               memtoRegMEM,
12                          regWriteMEM,
13
14      output reg  [31:0]  dmReadDataWB,
15                          aluResultWB,
16      output reg  [4:0]   registerWB,
17      output reg          memtoRegWB,
18                          regWriteWB
19  );
20
21      initial begin
22          dmReadDataWB    = 32'b0;
23          aluResultWB     = 32'b0;
24          registerWB      = 5'b0;
25          memtoRegWB      = 1'b0;
26          regWriteWB      = 1'b0;
27      end
```

```
28
29      always @ (posedge clk) begin
30          dmReadDataWB    <= dmReadDataMEM;
31          aluResultWB     <= aluResultMEM;
32          registerWB      <= registerMEM;
33          memtoRegWB      <= memtoRegMEM;
34          regWriteWB      <= regWriteMEM;
35      end
36
37  endmodule // MEM_WB
38
39  `endif // MODULE_MEM_WB
```

pipeline_tb.v

```
1   `timescale 1ns / 1ps
2
3   ////////////////////////////////////////////////////////////////////////////////
4   // Company:
5   // Engineer:
6   //
7   // Create Date:   21:34:58 11/21/2017
8   // Design Name:   pipeline
9   // Module Name:   /home/liu/VE370/p2/pipeline_tb.v
10  // Project Name:  p2
11  // Target Device:
12  // Tool versions:
13  // Description:
14  //
15  // Verilog Test Fixture created by ISE for module: pipeline
16  //
17  // Dependencies:
18  //
19  // Revision:
20  // Revision 0.01 - File Created
21  // Additional Comments:
22  //
23  ////////////////////////////////////////////////////////////////////////////////
24
25  `include "pipeline.v"
26
27  module pipeline_tb;
28
29      // Inputs
30      reg clk;
31      integer i = 0;
32
33      // Instantiate the Unit Under Test (UUT)
```

```
34      Pipeline uut (
35          .clk(clk)
36      );
37
38      initial begin
39          // Initialize Inputs
40          clk = 0;
41      end
42
43      always #10 begin
44          $display("Time: %d, CLK = %d, PC = 0x%H", i, clk,
            ↪ uut.pcOutIF);
45          $display("[$s0] = 0x%H, [$s1] = 0x%H, [$s2] = 0x%H",
            ↪ uut.registers.regs[16], uut.registers.regs[17],
            ↪ uut.registers.regs[18]);
46          $display("[$s3] = 0x%H, [$s4] = 0x%H, [$s5] = 0x%H",
            ↪ uut.registers.regs[19], uut.registers.regs[20],
            ↪ uut.registers.regs[21]);
47          $display("[$s6] = 0x%H, [$s7] = 0x%H, [$t0] = 0x%H",
            ↪ uut.registers.regs[22], uut.registers.regs[23],
            ↪ uut.registers.regs[8]);
48          $display("[$t1] = 0x%H, [$t2] = 0x%H, [$t3] = 0x%H",
            ↪ uut.registers.regs[9], uut.registers.regs[10],
            ↪ uut.registers.regs[11]);
49
            ↪ $display("--------------------------------------------------------");
50          clk = ~clk;
51          if (~clk) i = i + 1;
52      end
53
54  endmodule
```