



JOINT INSTITUTE
交大密西根学院

UM-SJTU Joint Institute
VE370 Introduction to Computer Organization

Project 2 Report - Sample

Based on Tianze's report

Contents

I Objective	1
II Top Level Block Diagram	1
(i) Single Cycle	1
(ii) Pipeline	1
III Design of Components	2
(i) IF Stage	2
1. PC MUX	2
2. PC Register	2
3. PC Adder 4	2
4. Instruction Memory	3
(ii) ID Stage	3
1. Register File	3
2. Control Unit	3
3. Hazard Detection Unit and ID Forwarding Unit	3
4. Comparator	3
(iii) EX Stage	4
1. Forwarding Unit and MUX	4
2. ALU Control	4
3. ALU Unit	4
(iv) Memory Stage and Write Back Stage	4
1. Data Memory	4
IV Control and Data Hazard	4
(i) EX stage	4
1. Data Hazard	4
(ii) ID stage	4
1. Control Hazard	4
2. Data Hazard	4
V Instruction Implementation	4
VI SSD and Top Module	4
VII RTL Schematic (Optional)	4
VIII Textual Result	8
IX Conclusion and Discussion	8
X Reference	9
XI Appendix	10
(i) Textual Result	10

I Objective

Fill here

- **The memory-reference instructions:** Fill here
- **The arithmetic-logical instructions:** Fill here
- **The jumping instructions:** Fill here

II Top Level Block Diagram

(i) Single Cycle

We use exactly the same figure(Figure 1) for our design from textbook.

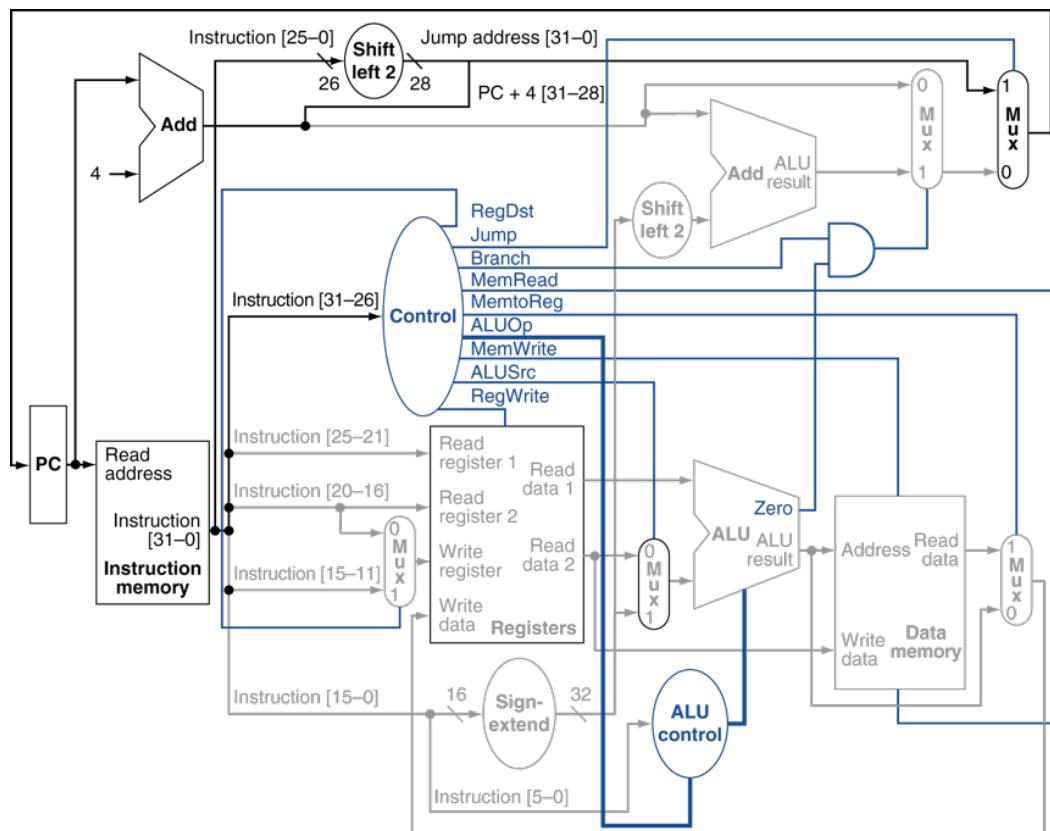


Figure 1: Single Cycle Diagram.

(ii) Pipeline

Since we cannot find a perfect diagram for pipeline, we draw a diagram by ourselves. Please see the next chapter for the figures corresponding to different stages.

III Design of Components

Since all the components used in Pipeline can be used in Single cycle, we'll only talk about the cases for pipeline.

(i) IF Stage

And the modules are given by:

1. PC MUX

This part is used to select PC source from branched address, jump address, or PC+4. The Verilog code for PC MUX is

```
module MUX_PC (PCp4, BranchPC, JumpPC, BNEorBEQ, JumpSignal, PC_in);
input [31:0] PCp4, BranchPC, JumpPC;
input BNEorBEQ, JumpSignal;
output reg [31:0] PC_in;

always @ ( * ) begin
    if (BNEorBEQ) begin
        PC_in = BranchPC;
    end else if (JumpSignal) begin
        PC_in = JumpPC;
    end else begin
        PC_in = PCp4;
    end
end

endmodule // MUX_PC
```

2. PC Register

3. PC Adder 4

4. Instruction Memory

```
module InstructionMem(PCAddress, Instruction);
// Your code here
memory[0] = 32'b0010000000001000000000000100000; //addi $t0, $zero, 0x20
memory[1] = 32'b00100000000010010000000000110111; //addi $t1, $zero, 0x37
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'b10101100000010000000000000000100; //sw $t0, 8($zero)
memory[6] = 32'b00000001000010011000100000100000; //add $s1, $t0, $t1
memory[7] = 32'b00000001000010011001000000100010; //sub $s2, $t0, $t1
memory[8] = 32'b000100100011001000000000000001001; //beq $s1, $s2, error0
memory[9] = 32'b10001100000100010000000000000100; //lw $s1, 4($zero)
memory[10] = 32'b00110010001100100000000000100100; //andi $s2, $s1, 0x48
memory[11] = 32'b000100100011001000000000000001001; //beq $s1, $s2, error1
memory[12] = 32'b10001100000100110000000000000100; //lw $s3, 8($zero)
memory[13] = 32'b000100100001001100000000000001010; //beq $s0, $s3, error2
memory[14] = 32'b00000010010100011010000000101010; //slt $s4, $s2, $s1 (Last)
memory[15] = 32'b000100101000000000000000000001111; //beq $s4, $0, EXIT
memory[16] = 32'b00000010001000001001000000100000; //add $s2, $s1, $0
memory[17] = 32'b000010000000000000000000000001110; //j Last
memory[18] = 32'b00100000000010000000000000000000; //addi $t0, $0, 0(error0)
memory[19] = 32'b00100000000010010000000000000000; //addi $t1, $0, 0
memory[20] = 32'b000010000000000000000000000001111; //j EXIT
memory[21] = 32'b00100000000010000000000000000001; //addi $t0, $0, 1(error1)
memory[22] = 32'b00100000000010010000000000000001; //addi $t1, $0, 1
memory[23] = 32'b000010000000000000000000000001111; //j EXIT
memory[24] = 32'b001000000000100000000000000000010; //addi $t0, $0, 2(error2)
memory[25] = 32'b001000000000100100000000000000010; //addi $t1, $0, 2
memory[26] = 32'b000010000000000000000000000001111; //j EXIT
memory[27] = 32'b001000000000100000000000000000011; //addi $t0, $0, 3(error3)
memory[28] = 32'b001000000000100100000000000000011; //addi $t1, $0, 3
memory[29] = 32'b000010000000000000000000000001111; //j EXIT
```

(ii) ID Stage

1. Register File
2. Control Unit
3. Hazard Detection Unit and ID Forwarding Unit
4. Comparator

(iii) EX Stage

1. Forwarding Unit and MUX
2. ALU Control
3. ALU Unit

(iv) Memory Stage and Write Back Stage

1. Data Memory

IV Control and Data Hazard

(i) EX stage

1. Data Hazard

Forwarding Unit

Forwarding MUX EX

(ii) ID stage

1. Control Hazard
2. Data Hazard

Hazard Detection Unit

Forwarding MUX ID

V Instruction Implementation

VI SSD and Top Module

VII RTL Schematic (Optional)

Please note that this part is not mandatory. *Please zoom in to check the details.*

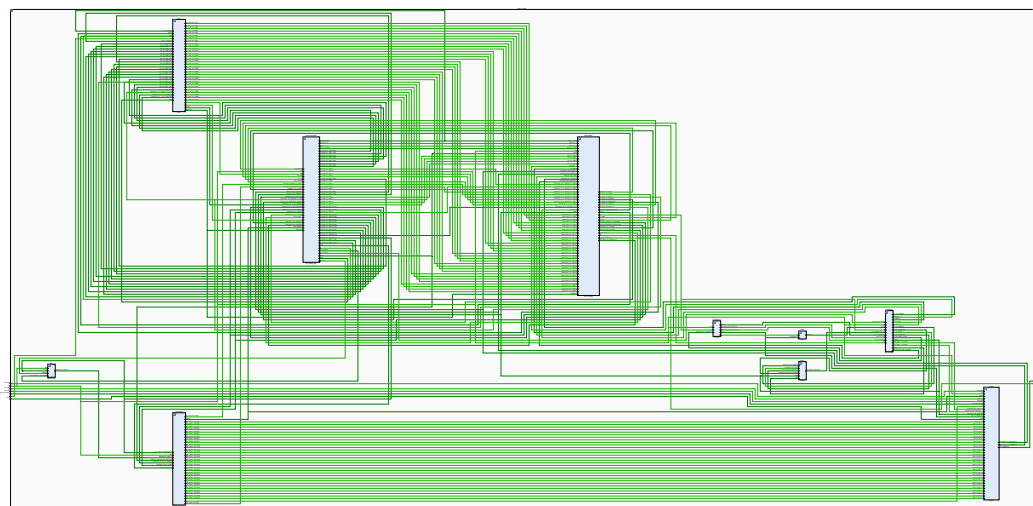


Figure 2: Schematic for Pipeline Main module.

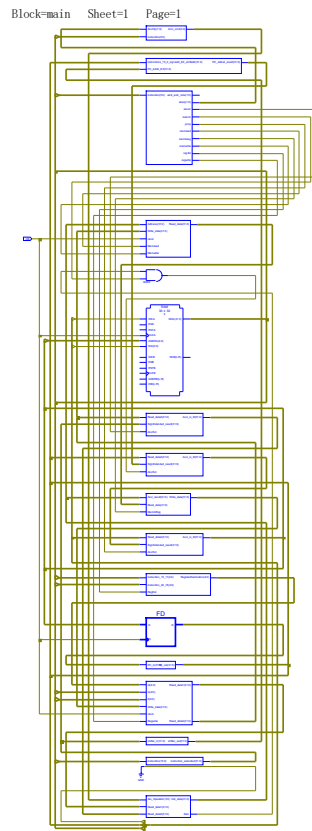


Figure 3: Schematic for Single Cycle.

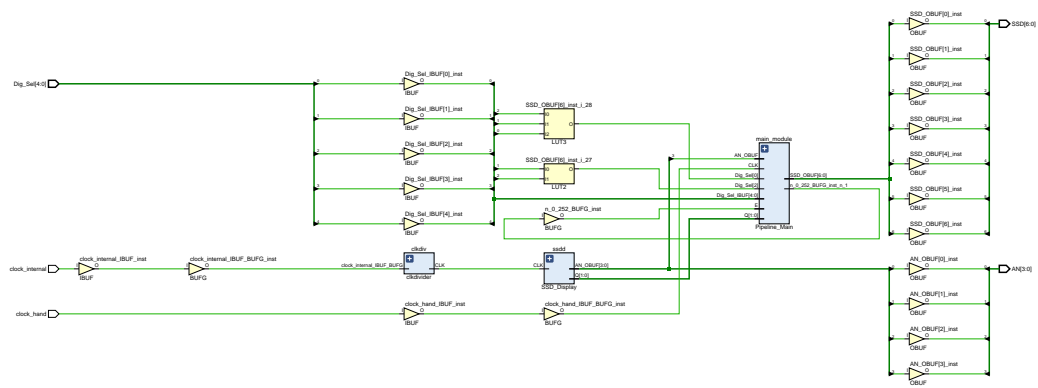


Figure 4: Schematic including SSD.

VIII Textual Result

By running the instruction memory, we can achieve the following **Textual Result**. (Too long so that we put it in the appendix part)

IX Conclusion and Discussion

Needn't be too long

X Reference

XI Appendix

(i) Textual Result

```
VCD info: dumpfile pipeline_test.vcd opened for output.
=====

Time =          0, CLK = 1, 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 =          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
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000

=====

Time =          1, 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
[$t4] = 0x00000000, [$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, [$t0] = 0x00000000
[$t1] = 0x00000000, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000

=====

Time =          2, 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
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000

=====

Time =          3, CLK = 0, PC = 0x0000000c

[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
```



```
[t7] = 0x00000000, [t8] = 0x00000000, [t9] = 0x00000000
```

Time = 24, CLK = 0, PC = 0x00000040

Time = 24, CLK = 1, PC = 0x00000040

Time = 25, CLK = 0, PC = 0x00000040

Time = 25, CLK = 1, PC = 0x00000040

Time = 26, CLK = 0, PC = 0x00000044

Time = 26, CLK = 1, PC = 0x00000044


```
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000  
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000  
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
```

```
=====
```