

VE370 Project Report

Project 2 - Fall 2020

Name: Qinhang Wu

Id: 518370910041

Email: william_wu@sjtu.edu.cn

Last modified: Nov.12, 2020

Table of Contents

- Single-cycle Processor Design
- Verilog Implementation
- Verilog Source Code
- Peer Evaluation

VE370 Project 2 Individual Report

VE370 Project 2 Individual Report

Overview

Verilog Simulation

Verilog Source Code

main module

program counter

instruction memory

next pc

control

register

alu control

alu

data memory

testbench

Peer Evaluation

Overview

VE370 Project 2 is designed for students to get a better understanding of **Single-cycle and Pipelined Processor Design**.

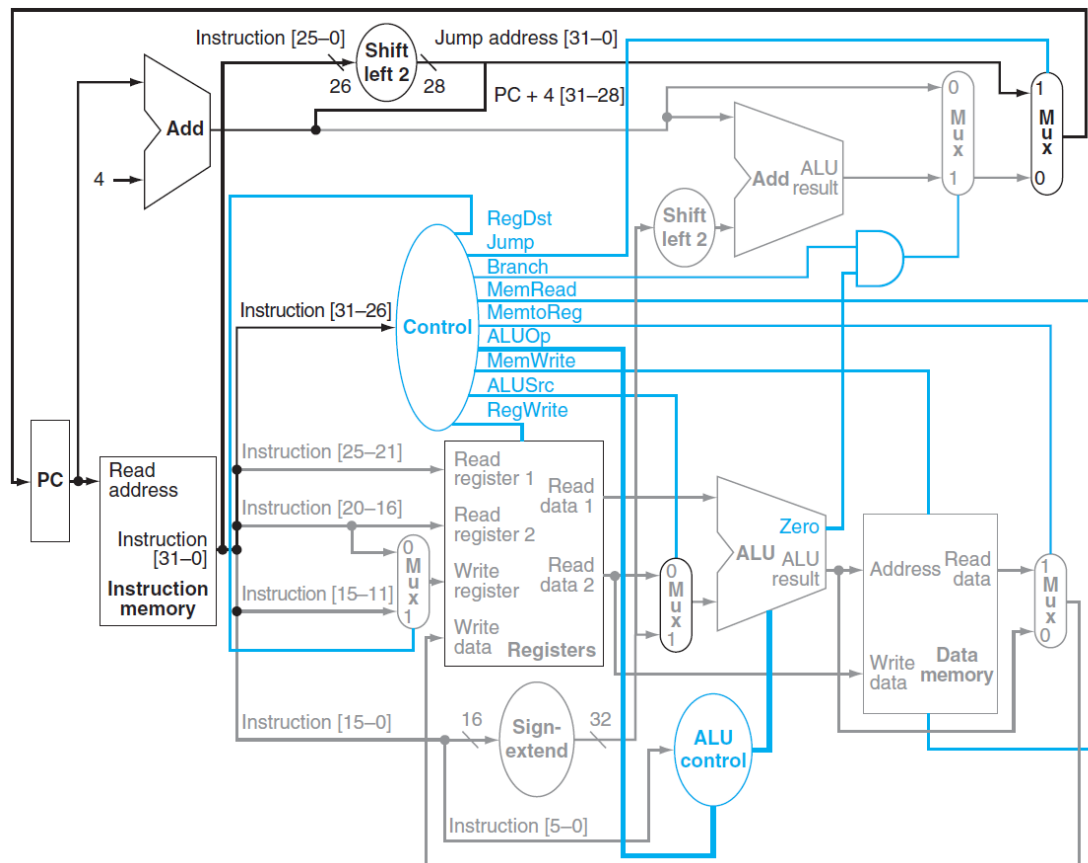


Figure 1. Single Cycle Diagram (MIPS)

Verilog Simulation

A set of instructions are used to test the single-cycle implementation.

Testcase:

```

1  00100000000010000000000000100000 //addi $t0, $zero, 0x20
2  00100000000010010000000000110111 //addi $t1, $zero, 0x37
3  00000001000010011000000000100100 //and $s0, $t0, $t1
4  00000001000010011000000000100101 //or $s0, $t0, $t1
5  101011000001000000000000000000100 //sw $s0, 4($zero)
6  101011000001000000000000000000100 //sw $t0, 8($zero)
7  00000001000010011000100000100000 //add $s1, $t0, $t1
8  00000001000010011001000000100010 //sub $s2, $t0, $t1
9  00100000000010000000000000100000 //addi $t0, $zero, 0x20
10 00100000000010000000000000100000 //addi $t0, $zero, 0x20
11 00100000000010000000000000100000 //addi $t0, $zero, 0x20
12 0001001000110010000000000010010 //beq $s1, $s2, error0
13 100011000001000100000000000000100 //lw $s1, 4($zero)
14 00110010001100100000000000100100 //andi $s2, $s1, 0x48
15 00100000000010000000000000100000 //addi $t0, $zero, 0x20
16 00100000000010000000000000100000 //addi $t0, $zero, 0x20
17 00100000000010000000000000100000 //addi $t0, $zero, 0x20
18 000100100011001000000000001111 //beq $s1, $s2, error1
19 1000110000010011000000000000100 //lw $s3, 8($zero)
20 00100000000010000000000000100000 //addi $t0, $zero, 0x20
21 00100000000010000000000000100000 //addi $t0, $zero, 0x20

```

```

22 00100000000010000000000000100000 //addi $t0, $zero, 0x20
23 00010010000100110000000000001101 //beq $s0, $s3, error2
24 00000010010100011010000000101010 //slt $s4, $s2, $s1 (Last)
25 00100000000010000000000000100000 //addi $t0, $zero, 0x20
26 00100000000010000000000000100000 //addi $t0, $zero, 0x20
27 00100000000010000000000000100000 //addi $t0, $zero, 0x20
28 00010010100000000000000000001111 //beq $s4, $0, EXIT
29 00000010001000001001000000100000 //add $s2, $s1, $0
30 0000100000000000000000000010111 //j Last
31 00100000000010000000000000000000 //addi $t0, $0, 0(error0)
32 00100000000010010000000000000000 //addi $t1, $0, 0
33 00001000000000000000000000111111 //j EXIT
34 00100000000010000000000000000001 //addi $t0, $0, 1(error1)
35 00100000000010010000000000000001 //addi $t1, $0, 1
36 00001000000000000000000000111111 //j EXIT
37 00100000000010000000000000000010 //addi $t0, $0, 2(error2)
38 00100000000010010000000000000010 //addi $t1, $0, 2
39 00001000000000000000000000111111 //j EXIT
40 00100000000010000000000000000011 //addi $t0, $0, 3(error3)
41 00100000000010010000000000000011 //addi $t1, $0, 3
42 00001000000000000000000000111111 //j EXIT

```

Simulation result:

```

1 =====
2 -----
3 Time:          0, CLK = 1, PC = 0x00000000
4 [$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
5 [$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
6 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000000
7 [$t1] = 0x00000000, [$t2] = 0x00000000, [$t3] = 0x00000000
8 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
9 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
10 -----
11 Time:          20, CLK = 1, PC = 0x00000004
12 [$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
13 [$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
14 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
15 [$t1] = 0x00000000, [$t2] = 0x00000000, [$t3] = 0x00000000
16 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
17 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
18 -----
19 Time:          40, CLK = 1, PC = 0x00000008
20 [$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
21 [$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
22 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
23 [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
24 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
25 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
26 -----
27 Time:          60, CLK = 1, PC = 0x0000000c
28 [$s0] = 0x00000020, [$s1] = 0x00000000, [$s2] = 0x00000000

```

```

29  [$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
30  [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
31  [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
32  [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
33  [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
34  -----
35  Time:           80, CLK = 1, PC = 0x00000010
36  [$s0] = 0x00000037, [$s1] = 0x00000000, [$s2] = 0x00000000
37  [$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
38  [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
39  [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
40  [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
41  [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
42  -----
43  Time:           100, CLK = 1, PC = 0x00000014
44  [$s0] = 0x00000037, [$s1] = 0x00000000, [$s2] = 0x00000000
45  [$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
46  [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
47  [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
48  [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
49  [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
50  -----
51  Time:           120, CLK = 1, PC = 0x00000018
52  [$s0] = 0x00000037, [$s1] = 0x00000000, [$s2] = 0x00000000
53  [$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
54  [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
55  [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
56  [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
57  [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
58  -----
59  Time:           140, CLK = 1, PC = 0x0000001c
60  [$s0] = 0x00000037, [$s1] = 0x00000057, [$s2] = 0x00000000
61  [$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
62  [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
63  [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
64  [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
65  [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
66  -----
67  Time:           160, CLK = 1, PC = 0x00000020
68  [$s0] = 0x00000037, [$s1] = 0x00000057, [$s2] = 0xffffffffe9
69  [$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
70  [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
71  [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
72  [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
73  [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
74  -----
75  Time:           180, CLK = 1, PC = 0x00000024
76  [$s0] = 0x00000037, [$s1] = 0x00000057, [$s2] = 0xffffffffe9
77  [$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
78  [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
79  [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
80  [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000

```

```

81  [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
82  -----
83  Time:           200, CLK = 1, PC = 0x00000028
84  [$s0] = 0x00000037, [$s1] = 0x00000057, [$s2] = 0xfffffffffe9
85  [$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
86  [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
87  [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
88  [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
89  [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
90  -----
91  Time:           220, CLK = 1, PC = 0x0000002c
92  [$s0] = 0x00000037, [$s1] = 0x00000057, [$s2] = 0xfffffffffe9
93  [$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
94  [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
95  [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
96  [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
97  [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
98  -----
99  Time:           240, CLK = 1, PC = 0x00000030
100 [$s0] = 0x00000037, [$s1] = 0x00000057, [$s2] = 0xfffffffffe9
101 [$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
102 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
103 [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
104 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
105 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
106 -----
107 Time:           260, CLK = 1, PC = 0x00000034
108 [$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0xfffffffffe9
109 [$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
110 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
111 [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
112 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
113 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
114 -----
115 Time:           280, CLK = 1, PC = 0x00000038
116 [$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
117 [$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
118 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
119 [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
120 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
121 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
122 -----
123 Time:           300, CLK = 1, PC = 0x0000003c
124 [$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
125 [$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
126 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
127 [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
128 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
129 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
130 -----
131 Time:           320, CLK = 1, PC = 0x00000040
132 [$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000

```

```

133  [$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
134  [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
135  [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
136  [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
137  [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
138  -----
139  Time:           340, CLK = 1, PC = 0x00000044
140  [$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
141  [$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
142  [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
143  [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
144  [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
145  [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
146  -----
147  Time:           360, CLK = 1, PC = 0x00000048
148  [$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
149  [$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
150  [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
151  [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
152  [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
153  [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
154  -----
155  Time:           380, CLK = 1, PC = 0x0000004c
156  [$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
157  [$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
158  [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
159  [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
160  [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
161  [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
162  -----
163  Time:           400, CLK = 1, PC = 0x00000050
164  [$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
165  [$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
166  [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
167  [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
168  [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
169  [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
170  -----
171  Time:           420, CLK = 1, PC = 0x00000054
172  [$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
173  [$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
174  [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
175  [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
176  [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
177  [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
178  -----
179  Time:           440, CLK = 1, PC = 0x00000058
180  [$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
181  [$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
182  [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
183  [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
184  [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000

```

```

185  [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
186  -----
187  Time:           460, CLK = 1, PC = 0x0000005c
188  [$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
189  [$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
190  [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
191  [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
192  [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
193  [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
194  -----
195  Time:           480, CLK = 1, PC = 0x00000060
196  [$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
197  [$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
198  [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
199  [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
200  [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
201  [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
202  -----
203  Time:           500, CLK = 1, PC = 0x00000064
204  [$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
205  [$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
206  [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
207  [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
208  [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
209  [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
210  -----
211  Time:           520, CLK = 1, PC = 0x00000068
212  [$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
213  [$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
214  [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
215  [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
216  [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
217  [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
218  -----
219  Time:           540, CLK = 1, PC = 0x0000006c
220  [$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
221  [$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
222  [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
223  [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
224  [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
225  [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
226  -----
227  Time:           560, CLK = 1, PC = 0x00000070
228  [$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
229  [$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
230  [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
231  [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
232  [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
233  [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
234  -----
235  Time:           580, CLK = 1, PC = 0x00000074
236  [$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037

```

```

237 [$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
238 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
239 [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
240 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
241 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
242 -----
243 Time:           600, CLK = 1, PC = 0x0000005c
244 [$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
245 [$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
246 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
247 [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
248 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
249 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
250 -----
251 Time:           620, CLK = 1, PC = 0x00000060
252 [$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
253 [$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
254 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
255 [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
256 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
257 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
258 -----
259 Time:           640, CLK = 1, PC = 0x00000064
260 [$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
261 [$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
262 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
263 [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
264 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
265 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
266 -----
267 Time:           660, CLK = 1, PC = 0x00000068
268 [$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
269 [$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
270 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
271 [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
272 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
273 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
274 -----
275 Time:           680, CLK = 1, PC = 0x0000006c
276 [$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
277 [$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
278 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
279 [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
280 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
281 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
282 -----
283 Time:           700, CLK = 1, PC = 0x000000ac
284 [$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
285 [$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
286 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
287 [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
288 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000

```



```

289 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
290 -----
291 Time:          720, CLK = 1, PC = 0x000000b0
292 [$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
293 [$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
294 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
295 [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
296 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
297 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
298 -----
299 Time:          740, CLK = 1, PC = 0x000000b4
300 [$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
301 [$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
302 [$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
303 [$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
304 [$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
305 [$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
306 -----
307 =====

```

Note: since `syscall` is not implemented here, a series of NOP instructions will be run by default after PC reaches EXIT. Several repeated NOP results are omitted here.

Verilog Source Code

main module

On top of the program, `main.v` interconnects different modules:

```

1  // main driver program for single-cycle processor
2
3  `timescale 1ns / 1ps
4  `include "alu_control.v"
5  `include "alu.v"
6  `include "control.v"
7  `include "data_memory.v"
8  `include "instru_memory.v"
9  `include "next_pc.v"
10 `include "program_counter.v"
11 `include "register.v"
12
13 module main(
14     input clk // clock signal for PC and RD
15 );
16
17     wire [31:0] pc_in,
18                pc_out;
19
20     wire [5:0] im_ctr;
21     wire [5:0] im_funcode;
22     wire [31:0] im_instru;

```

```

23
24     wire [31:0] r_wbdata, // dm_out
25               r_read1,
26               r_read2;
27
28     wire  c_RegDst,
29           c_Jump,
30           c_Branch,
31           c_Bne,
32           c_MemRead,
33           c_MemtoReg,
34           c_MemWrite,
35           c_ALUSrc,
36           c_RegWrite;
37     wire [1:0] c_ALUOp;
38
39     wire [3:0] c_ALUcontrol;
40
41     wire c_zero;
42     wire [31:0] alu_result;
43
44     // wire [31:0] dm_out;
45
46     program_counter asset_pc(
47         .clk (clk),
48         .next (pc_in),
49         .out (pc_out)
50     );
51
52     instru_memory asset_im(
53         .addr (pc_out),
54         .ctr (im_ctr),
55         .funcode (im_funcode),
56         .instru (im_instru)
57     );
58
59     register asset_reg(
60         .clk (clk),
61         .instru (im_instru),
62         .RegWrite (c_RegWrite),
63         .RegDst (c_RegDst),
64         .WriteData (r_wbdata),
65         .ReadData1 (r_read1),
66         .ReadData2 (r_read2)
67     );
68
69     alu asset_alu(
70         .data1 (r_read1),
71         .read2 (r_read2),
72         .instru (im_instru),
73         .ALUSrc (c_ALUSrc),
74         .ALUcontrol (c_ALUcontrol),

```

```

75     .zero (c_zero),
76     .ALUresult (alu_result)
77 );
78
79 alu_control asset_aluControl(
80     .ALUOp (c_ALUOp),
81     .instru (im_funcode),
82     .ALUcontrol (c_ALUcontrol)
83 );
84
85 control asset_control(
86     .instru (im_instru),
87     .RegDst (c_RegDst),
88     .Jump (c_Jump),
89     .Branch (c_Branch),
90     .Bne (c_Bne),
91     .MemRead (c_MemRead),
92     .MemtoReg (c_MemtoReg),
93     .ALUOp (c_ALUOp),
94     .MemWrite (c_MemWrite),
95     .ALUSrc (c_ALUSrc),
96     .RegWrite (c_RegWrite)
97 );
98
99 data_memory asset_dm(
100     .clk (clk),
101     .addr (alu_result), // im_instru
102     .wData (r_read2),
103     .ALUresult (alu_result),
104     .MemWrite (c_MemWrite),
105     .MemRead (c_MemRead),
106     .MemtoReg (c_MemtoReg),
107     .rData (r_wbdata)
108 );
109
110 next_pc asset_nextPc(
111     .old (pc_out),
112     .instru (im_instru),
113     .Jump (c_Jump),
114     .Branch (c_Branch),
115     .Bne (c_Bne),
116     .zero (c_zero),
117     .next (pc_in)
118 );
119
120 endmodule

```

program counter

`program_counter.v` describes the functionality of the program counter:

```
1  `timescale 1ns / 1ps
2
3  module program_counter(
4      input clk,
5      input [31:0] next, // the input address
6      output reg [31:0] out // the output address
7  );
8
9      initial begin
10         out = -4; // NEVER REACHED ADDRESS
11     end
12
13     always @(posedge clk) begin
14         out = next;
15     end
16
17 endmodule
```

instruction memory

`instru_memory.v` describes the functionality of the instruction memory:

```
1  `timescale 1ns / 1ps
2
3  module instru_memory(
4      // input clk,
5      input [31:0] addr,
6      output reg [5:0] ctr, // [31-26]
7      output reg [5:0] funcode, // [5-0]
8      // output reg [4:0] read1, // [25-21]
9      // output reg [4:0] read2, // [20-16]
10     // output reg [4:0] write, // [15-11]
11     output reg [31:0] instru // [31-0]
12     // output [15:0] num // [15-0]
13 );
14
15     parameter SIZE_IM = 128; // size of this memory, by default 128*32
16     reg [31:0] mem [SIZE_IM-1:0]; // instruction memory
17
18     integer n;
19     initial begin
20         for(n=0;n<SIZE_IM;n=n+1) begin
21             mem[n] = 32'b11111100000000000000000000000000;
22         end
23         $readmemb("C:\\Users\\William Wu\\Documents\\Mainframe Files\\UMJI-
24             SJTU\\1 Academy\\20
25             Fall\\VE370\\Project\\p2\\single_cycle\\testcases\\testcase.txt",mem);
26         // NOTE: the absolute path is used here.
```

```

25     instru = 32'b11111100000000000000000000000000;
26 end
27
28 always @(addr) begin
29     if (addr == -4) begin // init
30         instru = 32'b11111100000000000000000000000000;
31     end else begin
32         instru = mem[addr >> 2];
33     end
34     ctr = instru[31:26];
35     funcode = instru[5:0];
36 end
37
38 endmodule

```

next pc

`next_pc.v` describes the functionality of calculating the next address:

```

1  `timescale 1ns / 1ps
2
3  module next_pc(
4      input [31:0] old, // the original program addr.
5      input [31:0] instru, // the original instruction
6          // [15:0] used for sign-extention
7          // [25:0] used for shift-left-2
8      input Jump,
9      input Branch,
10     input Bne,
11     input zero,
12     output reg [31:0] next
13 );
14
15     reg [31:0] sign_ext;
16     reg [31:0] old_alter; // pc+4
17     reg [31:0] jump; // jump addr.
18     reg zero_alter;
19
20     initial begin
21         next = 32'b0;
22     end
23
24     always @(old) begin
25         old_alter = old + 4;
26     end
27
28     always @(zero,Bne) begin
29         zero_alter = zero;
30         if (Bne == 1) begin
31             zero_alter = ! zero_alter;
32         end
33     end

```

```

34
35     always @(instru) begin
36         // jump-shift-left
37         jump = {4'b0,instru[25:0],2'b0};
38
39         // sign-extension
40         if (instru[15] == 1'b0) begin
41             sign_ext = {16'b0,instru[15:0]};
42         end else begin
43             sign_ext = {{16{1'b1}},instru[15:0]};
44         end
45         sign_ext = {sign_ext[29:0],2'b0}; // shift left
46     end
47
48     always @(instru or old_alter or jump) begin
49         jump = {old_alter[31:28],jump[27:0]};
50     end
51
52     always @(old_alter,sign_ext,jump,Branch,zero_alter,Jump) begin
53         // assign next program counter value
54         if (Branch == 1 & zero_alter == 1) begin
55             // $display("Taking branch");
56             next = old_alter + sign_ext;
57         end else begin
58             // $display("Normal proceeding");
59             next = old_alter;
60         end
61         if (Jump == 1) begin
62             // $display("Taking jump");
63             next = jump;
64         end
65     end
66
67 endmodule

```

control

`control.v` describes the functionality of generating different control signals:

```

1  `timescale 1ns / 1ps
2
3  module control(
4      input [31:0] instru,
5      output reg RegDst,
6      output reg Jump,
7      output reg Branch,
8      output reg Bne, // 1 indicates bne
9      output reg MemRead,
10     output reg MemtoReg,
11     output reg [1:0] ALUOp,
12     output reg MemWrite,
13     output reg ALUSrc,

```

```

14     output reg RegWrite
15 );
16
17     initial begin
18         RegDst = 0;
19         Jump = 0;
20         Branch = 0;
21         MemRead = 0;
22         MemtoReg = 0;
23         ALUOp = 2'b00;
24         MemWrite = 0;
25         ALUSrc = 0;
26         RegWrite = 0;
27     end
28
29     always @(instru) begin
30         case (instru[31:26])
31             6'b000000: begin// ARITHMETIC
32                 RegDst = 1;
33                 ALUSrc = 0;
34                 MemtoReg = 0;
35                 RegWrite = 1;
36                 MemRead = 0;
37                 MemWrite = 0;
38                 Branch = 0;
39                 Bne = 0;
40                 ALUOp = 2'b10;
41                 Jump = 0;
42             end
43             6'b001000: begin// addi
44                 RegDst = 0;
45                 ALUSrc = 1;
46                 MemtoReg = 0;
47                 RegWrite = 1;
48                 MemRead = 0;
49                 MemWrite = 0;
50                 Branch = 0;
51                 Bne = 0;
52                 ALUOp = 2'b00;
53                 Jump = 0;
54             end
55             6'b001100: begin// andi
56                 RegDst = 0;
57                 ALUSrc = 1;
58                 MemtoReg = 0;
59                 RegWrite = 1;
60                 MemRead = 0;
61                 MemWrite = 0;
62                 Branch = 0;
63                 Bne = 0;
64                 ALUOp = 2'b11;
65                 Jump = 0;

```

```
66     end
67     6'b100011: begin // lw
68         RegDst = 0;
69         ALUSrc = 1;
70         MemtoReg = 1;
71         RegWrite = 1;
72         MemRead = 1;
73         MemWrite = 0;
74         Branch = 0;
75         Bne = 0;
76         ALUOp = 2'b00;
77         Jump = 0;
78     end
79     6'b101011: begin // sw
80         RegDst = 0; // x
81         ALUSrc = 1;
82         MemtoReg = 0; // x
83         RegWrite = 0;
84         MemRead = 0;
85         MemWrite = 1;
86         Branch = 0;
87         Bne = 0;
88         ALUOp = 2'b00;
89         Jump = 0;
90     end
91     6'b000100: begin // beq
92         RegDst = 0; // x
93         ALUSrc = 0;
94         MemtoReg = 0; // x
95         RegWrite = 0;
96         MemRead = 0;
97         MemWrite = 0;
98         Branch = 1;
99         Bne = 0;
100        ALUOp = 2'b01;
101        Jump = 0;
102    end
103    6'b000101: begin // bne
104        RegDst = 0; // x
105        ALUSrc = 0;
106        MemtoReg = 0; // x
107        RegWrite = 0;
108        MemRead = 0;
109        MemWrite = 0;
110        Branch = 1;
111        Bne = 1;
112        ALUOp = 2'b01;
113        Jump = 0;
114    end
115    6'b000010: begin // j
116        RegDst = 0; // x
117        ALUSrc = 0;
```



```

118     MemtoReg = 0; // x
119     RegWrite = 0;
120     MemRead = 0;
121     MemWrite = 0;
122     Branch = 0;
123     Bne = 0;
124     ALUOp = 2'b01;
125     Jump = 1;
126 end
127 default: begin
128     RegDst = 0; // x
129     ALUSrc = 0;
130     MemtoReg = 0; // x
131     RegWrite = 0;
132     MemRead = 0;
133     MemWrite = 0;
134     Branch = 0;
135     Bne = 0;
136     ALUOp = 2'b00;
137     Jump = 0;
138 end
139 endcase
140 end
141
142 endmodule

```

register

`register.v` describes the functionality of register management:

```

1  `timescale 1ns / 1ps
2
3  module register(
4      input clk,
5      input [31:0] instru, // the raw 32-bit instruction
6      input RegWrite,
7      input RegDst,
8      input [31:0] WriteData, // from WB stage
9      // input [4:0] WriteReg,
10     output [31:0] ReadData1,
11     output [31:0] ReadData2
12 );
13
14     reg [31:0] RegData [31:0]; // register data
15
16     // initialize the register data
17     integer i;
18     initial begin
19         for(i=0;i<32;i=i+1) begin
20             RegData[i] = 32'b0;
21         end
22     end

```

```

23
24     assign ReadData1 = RegData[instru[25:21]];
25     assign ReadData2 = RegData[instru[20:16]];
26
27     always @(posedge clk) begin // RegWrite, RegDst, WriteData, instru)
28         if (RegWrite == 1'b1) begin
29             if (RegDst == 1'b0) begin
30                 RegData[instru[20:16]] = WriteData;
31             end else begin
32                 RegData[instru[15:11]] = WriteData;
33             end
34         end
35     end
36
37 endmodule

```

alu control

`alu_control` describes the functionality of generating alu control signals:

```

1  `timescale 1ns / 1ps
2
3  module alu_control(
4      input [1:0] ALUOp,
5      input [5:0] instru,
6      output reg [3:0] ALUcontrol
7  );
8
9      always @(ALUOp, instru) begin
10         case (ALUOp)
11             2'b00:
12                 ALUcontrol = 4'b0010;
13             2'b01:
14                 ALUcontrol = 4'b0110;
15             2'b10: begin
16                 case (instru)
17                     6'b100000: // add
18                         ALUcontrol = 4'b0010;
19                     6'b100010: // sub
20                         ALUcontrol = 4'b0110;
21                     6'b100100: // and
22                         ALUcontrol = 4'b0000;
23                     6'b100101: // or
24                         ALUcontrol = 4'b0001;
25                     6'b101010: // slt
26                         ALUcontrol = 4'b0111;
27                     default:
28                         ;
29                 endcase
30             end
31             2'b11:
32                 ALUcontrol = 4'b0000;

```

```

33     default:
34         ;
35     endcase
36 end
37
38 endmodule

```

alu

`alu.v` describes the functionality of the alu unit:

```

1  `timescale 1ns / 1ps
2
3  module alu(
4      input [31:0] data1,
5      input [31:0] read2,
6      input [31:0] instru, // used for sign-extension
7      input ALUSrc,
8      input [3:0] ALUcontrol,
9      output reg zero,
10     output reg [31:0] ALUresult
11 );
12
13     reg [31:0] data2;
14
15     always @(ALUSrc, read2, instru) begin
16         if (ALUSrc == 0) begin
17             data2 = read2;
18         end else begin
19             // SignExt[Instru[15:0]]
20             if (instru[15] == 1'b0) begin
21                 data2 = {16'b0, instru[15:0]};
22             end else begin
23                 data2 = {{16{1'b1}}, instru[15:0]};
24             end
25         end
26     end
27
28     always @(data1, data2, ALUcontrol) begin
29         case (ALUcontrol)
30             4'b0000: // AND
31                 ALUresult = data1 & data2;
32             4'b0001: // OR
33                 ALUresult = data1 | data2;
34             4'b0010: // ADD
35                 ALUresult = data1 + data2;
36             4'b0110: // SUB
37                 ALUresult = data1 - data2;
38             4'b0111: // SLT
39                 ALUresult = (data1 < data2) ? 1 : 0;
40             4'b1100: // NOR
41                 ALUresult = data1 |~ data2;

```

```

42     default:
43         ;
44     endcase
45     if (ALUresult == 0) begin
46         zero = 1;
47     end else begin
48         zero = 0;
49     end
50 end
51
52 endmodule

```

data memory

`data_memory.v` describes the functionality of the data memory:

```

1  `timescale 1ns / 1ps
2
3  module data_memory(
4      input clk,
5      input [31:0] addr,
6      input [31:0] wData,
7      input [31:0] ALUresult,
8      input MemWrite,
9      input MemRead,
10     input MemtoReg,
11     output reg [31:0] rData
12 );
13
14     parameter SIZE_DM = 128; // size of this memory, by default 128*32
15     reg [31:0] mem [SIZE_DM-1:0]; // instruction memory
16
17     // initially set default data to 0
18     integer i;
19     initial begin
20         for(i=0; i<SIZE_DM-1; i=i+1) begin
21             mem[i] = 32'b0;
22         end
23     end
24
25     always @(addr or MemRead or MemtoReg or ALUresult) begin
26         if (MemRead == 1) begin
27             if (MemtoReg == 1) begin
28                 rData = mem[addr];
29             end else begin
30                 rData = ALUresult; // x ?
31             end
32         end else begin
33             rData = ALUresult;
34         end
35     end
36

```

```

37     always @(posedge clk) begin // MemWrite, wData, addr
38         if (MemWrite == 1) begin
39             mem[addr] = wData;
40         end
41     end
42
43 endmodule

```

testbench

Finally, a testbench is included in `testbench.v` to test and simulate the program:

```

1  // testbench for simulation of the single-cycle processor
2
3  `timescale 1ns / 1ps
4  `include "main.v"
5
6  module testbench;
7      integer currTime;
8      reg clk;
9
10     main uut(
11         .clk (clk)
12     );
13
14     initial begin
15         #0
16         clk = 0;
17         currTime = -10;
18         uut.asset_pc.out = -4;
19
20         $display("=====");
21
22         #988
23         $display("=====");
24         #989 $stop;
25     end
26
27     always @(posedge clk) begin
28         // indicating a posedge clk triggered
29         $display("-----
30
31         #1; // wait for writing back
32         $display("Time: %d, CLK = %d, PC = 0x%H",currTime, clk,
33             uut.asset_pc.out);
34         $display("[0] = 0x%H, [1] = 0x%H, [2] =
35             0x%H",uut.asset_reg.RegData[16],uut.asset_reg.RegData[17],uut.asset_reg.
36             RegData[18]);
37         $display("[3] = 0x%H, [4] = 0x%H, [5] =
38             0x%H",uut.asset_reg.RegData[19],uut.asset_reg.RegData[20],uut.asset_reg.
39             RegData[21]);

```

```

32    $display("$s6 = 0x%H, [$s7] = 0x%H, [$t0] =
    0x%H", uut.asset_reg.RegData[22], uut.asset_reg.RegData[23], uut.asset_reg.
    RegData[8]);
33    $display("$t1 = 0x%H, [$t2] = 0x%H, [$t3] =
    0x%H", uut.asset_reg.RegData[9], uut.asset_reg.RegData[10], uut.asset_reg.R
    egData[11]);
34    $display("$t4 = 0x%H, [$t5] = 0x%H, [$t6] =
    0x%H", uut.asset_reg.RegData[12], uut.asset_reg.RegData[13], uut.asset_reg.
    RegData[14]);
35    $display("$t7 = 0x%H, [$t8] = 0x%H, [$t9] =
    0x%H", uut.asset_reg.RegData[15], uut.asset_reg.RegData[24], uut.asset_reg.
    RegData[25]);
36    end
37
38    always #10 begin
39        clk = ~clk;
40        currTime = currTime + 10;
41    end
42
43    endmodule

```

Peer Evaluation

Name	Level of contribution (0~5)	Description of contribution
Wu Qinhang (me)	5	system design and debug module integration synthesis and FPGA implementation
Xu Jiaying	5	patch for pipelined system RTL schematic debug and report refinement
Liu Yuru	5	hazard detection unit branch bonus unit RTL schematic and report refinement
Yang Gengchen	5	forwarding unit FPGA implementation debug and report refinement

Each of the group member contributes to the pipelined processor equally.