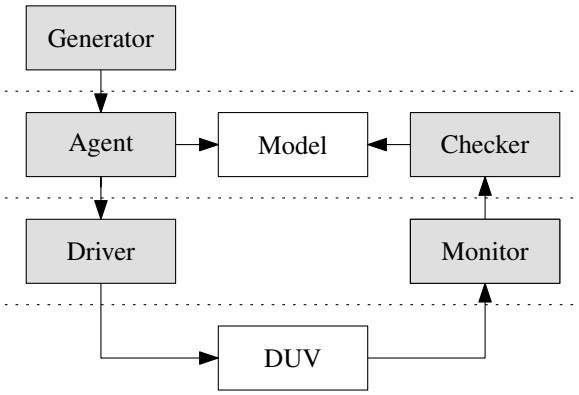
1 Verifikation des PROL16

In dieser Übung sollen Sie Ihren PROL16-Entwurf aus dem fünften Semester mit Hilfe des in der letzten Übung erstellten Modells (*"Golden Model"*) verifizieren. Dazu soll jeder generierte Befehl (also jede Instanz der Klasse `Prol16Opcode`) nicht nur von dem Modell, sondern auch auf dem VHDL-Entwurf exekutiert werden. Nach jedem Befehl kann der Registerinhalt der beiden Beschreibungen verglichen und damit geprüft werden, ob ein Fehler in der Befehlsausführung aufgetreten ist.

Die Testbench ist also wie im Folgenden dargestellt aufgebaut:



Erweitern Sie Ihre Testbench also aus der letzten Übung also so, dass das VHDL-Modell des PROL16 instantiiert und mit den Befehlen des Stimulus-Generators gespeist wird. Dabei wird ein Befehl nach dem anderen an den PROL16 angelegt, die vom Prozessor vorgegebene Adresse muss *nicht* beachtet werden! Die Befehle **SLEEP**, **LOAD** und **STORE** werden von dem Modell nicht unterstützt und daher in dieser Simulation vernachlässigt.

1.1 Zugriff auf VHDL-Signale aus der Testbench

Um den Vergleich des Zustands des Modells mit dem des VHDL-Entwurfs zu ermöglichen, muss aus der SystemVerilog-Testbench auf Signale des VHDL-Entwurfs zugegriffen werden. Da dies von VHDL nicht unterstützt wird, müssen dazu simulatorspezifische Funktionen verwendet werden. So können Sie mit *QuestaSim* auf ein Signal `pc`, das in der Instanz `/top/duv/datapath_inst` existiert, aus der Testbench folgendermaßen zugreifen:

```
module top;
    ...
    Prol16Test tb(...);
    cpu duv(...);
endmodule


program Prol16Test(...);
  logic [15:0] cpu_pc;
  ...
  initial begin
    $init_signal_spy("/top/duv/datapath_inst/pc", "/top/tb/cpu_pc");
    ...
    assert (cpu_pc == 0);
  end
endprogram
```

Beachten Sie, dass die *SignalSpy*-Funktion von *QuestaSim* keine RAMs (zweidimensionale Arrays) unterstützt – Sie müssen also jedes Register der Registerbank einzeln in die Testbench einbinden.

1.2 Reset

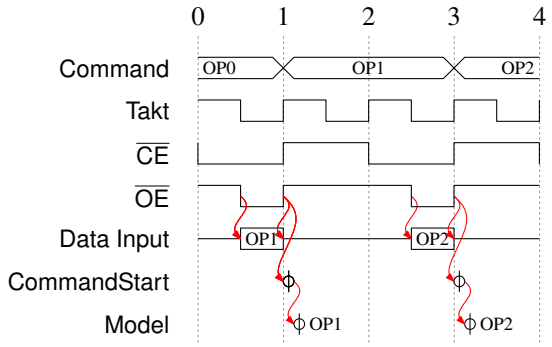
Das Register des PROL16 werden beim Reset nicht zurückgesetzt, haben also zu Beginn der Simulation keinen definierten Wert. Mit simulatorspezifischen Funktionen können Sie einen definitieren Initialwert setzen (Beispiel *QuestaSim*):

```
program Prol16Test(...);
  initial begin
    $signal_force("/top/duv/datapath_inst/thereg_file/registers(0)",
            "16#0000", 0, 1);
    ...
  end
endprogram
```

□ Erklären Sie die Parameter dieser Funktion!

1.3 Synchronisation

Die Herausforderung an dieser Testbench liegt in der Synchronistation der Testbench, des DUVs und des Modells. Eine einfache Lösung kann dabei mit *Events* realisiert werden, wie im Folgenden dargestellt:



- Mit der fallenden Flanke von $\overline{\text{OE}}$ muss der *nächste* Befehl (OP1) an das DUV angelegt werden.
- Mit der steigenden Taktflanke wird der Befehl in der CPU übernommen und gleichzeitig die Ergebnisse des *letzten* Befehls (OP0) in die Register geschrieben. Direkt danach wird $\overline{\text{OE}}$ wieder auf *high* gesetzt.
- Es ist also sinnvoll, direkt nach der steigenden Taktflanke ein Event (CommandStart) zu generieren, aufgrund dessen der Registerinhalt (Ergebnis von OP0) überprüft wird. *Nach* dieser Überprüfung kann das Modell den neuen Befehl (OP1) ausführen (Event Model).
- Mit der nächsten fallenden Flanke von $\overline{\text{OE}}$ wird wieder beim ersten Punkt begonnen. Die einzige Ausnahme bildet der Befehl **LOADI**, bei dem zu diesem Zeitpunkt die Daten angelegt werden müssen, und erst bei der nächsten fallenden Flanke wieder von vorne begonnen wird.

"There has never been an unexpectedly short debugging period in the history of computers."

Steven Levy

1 Beantwortung der Frage

Die Funktion hat die Form `$signal_force(<dest_object>, <value>, <rel_time>, <force_type>, <cancel_period>, <verbose>)`. In der Angabe wurden die Parameter `<dest_object>`, `<value>`, `<rel_time>` und `<force_type>` verwendet. `<dest_object>` ist dabei der Pfad zum Signal, `<value>` der Wert auf den das Signal gesetzt werden soll, `<rel_time>` die relative Zeit wann das Setzen des Signals stattfinden soll und `<force_type>` gibt an wie das Signal getrieben werden soll.

2 Testfälle

Die Operationen werden auf der CPU, das heißt dem DUV, ausgeführt. Danach werden die Werte vom PC, den Registern und die Flags mit dem Modell des Prol16 verglichen.
Es gibt zwei Möglichkeiten die Register und die Carry Flags zu setzen: Einerseits mithilfe von `signal_force` die Signale direkt in der CPU setzen oder andererseits die Register mit `loadi` zu laden bzw. die Flags mit z.B. Add.
In den Testfällen wurden die Signale mit `signal_force` gesetzt. Das Problem hier ist allerdings dass die Register und Flags dann den Wert behalten und nicht bei der Operation die gerade ausgeführt wird geändert werden. Dies führt dazu, dass die Register und Flags dann beim nächsten Befehl falsch sind.

3 Source Code

Der Sourcecode des Prol16 wurde nicht hinzugefügt, da der von der Elearning Plattform verwendet wurde.

../../src/sv/ifProl16.sv

```
1  interface ifProl16 #(parameter int gDataWidth = 16);
2
3     logic [gDataWidth-1:0] mem_addr;     // unused
4     logic [gDataWidth-1:0] mem_data_cpu;    // data from cpu
5     logic [gDataWidth-1:0] mem_data_tb;     // data from tb
6     logic mem_ce_n;
7     logic mem_oe_n;
8     logic mem_we_n;                      // unused
9     logic illegal_inst;
10    logic cpu_halt;
11
12    modport master (
13       output mem_data_tb,
14       input mem_data_cpu, mem_ce_n, mem_oe_n, illegal_inst, cpu_halt
15    );
16
17 endinterface
```

../../src/sv/pkgProl16.sv

```
1  package pkgProl16;
2     typedef bit [15:0] data_v;
3  endpackage
```

../../src/sv/Prol16Command.sv

```
1  typedef enum bit [5:0]
2     {
```

```
3        Nop=6'b000000,
4        Loadi=6'b000010,
5        Jump=6'b001000,
6        Jumpc=6'b001010,
7        Jumpz=6'b001011,
8        Move=6'b001100,
9        And=6'b010000,
10       Or=6'b010001,
11       Xor=6'b010010,
12       Not=6'b010011,
13       Add=6'b010100,
14       Addc=6'b010101,
15       Sub=6'b010110,
16       Subc=6'b010111,
17       Comp=6'b011000,
18       Inc=6'b011010,
19       Dec=6'b011011,
20       Shl=6'b011100,
21       Shr=6'b011101,
22       Shlc=6'b011110,
23       Shrc=6'b011111,
24       Invalid=6'b111111
25    } Prol16Command;
```

../../src/sv/Prol16Opcode.sv

```
1  `include "Prol16Command.sv"
2
3  class Prol16Opcode;
4     int ra;
5     int rb;
6     Prol16Command cmd;
7     pkgProl16::data_v data;
8
9     function new(int ra, int rb, Prol16Command cmd, pkgProl16::data_v data);
10       this.ra = ra;
11       this.rb = rb;
12       this.cmd = cmd;
13       this.data = data;
14    endfunction
15 endclass
```

../../src/sv/Prol16State.sv

```
1  class Prol16State #(parameter int gRegs = 32);
2     pkgProl16::data_v regs[gRegs];
3     int programCounter;
4     bit cFlag;
5     bit zFlag;
6
7     function new();
8        for (int i = 0; i < gRegs; i++) begin
9           regs[i] = '0;
10       end
11       programCounter = 0;
12       cFlag = 0;
13       zFlag = 0;
14    endfunction
15
16    task reset();
17       for (int i = 0; i < gRegs; i++) begin
18          regs[i] = '0;
```

```
19        end
20        programCounter = 0;
21        cFlag = 0;
22        zFlag = 0;
23     endtask
24  endclass
```

../../src/sv/Prol16Model.sv

```
1   `include "Prol16State.sv"
2   `include "Prol16Opcode.sv"
3
4   class Prol16Model#(parameter int gNumRegs);
5      Prol16State state;
6
7      function new(Prol16State _state);
8         state = _state;
9      endfunction
10
11     task reset();
12        state.reset();
13     endtask
14
15     task calcCarryFlag(int data);
16        if (data > (2**$size(pkgProl16::data_v) - 1) || data < 0)
17           state.cFlag = 1;
18        else
19           state.cFlag = 0;
20     endtask
21
22     task calcZeroFlag(pkgProl16::data_v data);
23        if (data == '0)
24           state.zFlag = 1;
25        else
26           state.zFlag = 0;
27     endtask
28
29     task execute(Prol16Opcode opcode);
30        int data = 0;  // tmp data for carry calc
31        bit tmpCFlag = 0; // tmp carry flag
32
33        // inc pc here, overwrite in jumps
34        state.programCounter++;
35
36       if (opcode.ra > (gNumRegs - 1) || opcode.ra < 0 ||
37           opcode.rb > (gNumRegs - 1) || opcode.rb < 0)
38           $display("Illegal register, doing nop");
39       else
40        begin
41
42        case (opcode.cmd)
43           Nop:
44           begin
45           end
46
47           Loadi:
48           begin
49              state.regs[opcode.ra] = opcode.data;
50              state.programCounter++; // TODO: necessary?
51           end
52
```

```
53          Jump:
54          begin
55             state.programCounter = state.regs[opcode.ra];
56          end
57
58          Jumpc:
59          begin
60             if (state.cFlag == 1)
61                state.programCounter = state.regs[opcode.ra];
62          end
63
64          Jumpz:
65          begin
66             if (state.zFlag == 1)
67                state.programCounter = state.regs[opcode.ra];
68          end
69
70          Move:
71          begin
72             state.regs[opcode.ra] = state.regs[opcode.rb];
73          end
74
75          And:
76          begin
77             state.regs[opcode.ra] = state.regs[opcode.ra] & state.regs[
                   opcode.rb];
78             state.cFlag = 0;
79             calcZeroFlag(state.regs[opcode.ra]);
80          end
81
82          Or:
83          begin
84             state.regs[opcode.ra] = state.regs[opcode.ra] | state.regs[
                   opcode.rb];
85             state.cFlag = 0;
86             calcZeroFlag(state.regs[opcode.ra]);
87          end
88
89          Xor:
90          begin
91             state.regs[opcode.ra] = state.regs[opcode.ra] ^ state.regs[
                   opcode.rb];
92             state.cFlag = 0;
93             calcZeroFlag(state.regs[opcode.ra]);
94          end
95
96          Not:
97          begin
98             state.regs[opcode.ra] = ~(state.regs[opcode.ra]);
99             state.cFlag = 0;
100            calcZeroFlag(state.regs[opcode.ra]);
101         end
102
103         Add:
104         begin
105            data = state.regs[opcode.ra] + state.regs[opcode.rb];
106            calcCarryFlag(data);
107            state.regs[opcode.ra] = data;
108            calcZeroFlag(state.regs[opcode.ra]);
109         end
```

```
110
111         Addc:
112         begin
113            data = state.regs[opcode.ra] + state.regs[opcode.rb] + state.
                  cFlag;
114            calcCarryFlag(data);
115            state.regs[opcode.ra] = data;
116            calcZeroFlag(state.regs[opcode.ra]);
117         end
118
119         Sub:
120         begin
121            data = state.regs[opcode.ra] - state.regs[opcode.rb];
122            calcCarryFlag(data);
123            state.regs[opcode.ra] = data;
124            calcZeroFlag(state.regs[opcode.ra]);
125         end
126
127         Subc:
128         begin
129            data = state.regs[opcode.ra] - state.regs[opcode.rb] - state.
                  cFlag;
130            calcCarryFlag(data);
131            state.regs[opcode.ra] = data;
132            calcZeroFlag(state.regs[opcode.ra]);
133         end
134
135         Comp:
136         begin
137            data = state.regs[opcode.ra] - state.regs[opcode.rb];
138            calcCarryFlag(data);
139            calcZeroFlag(data);
140         end
141
142         Inc:
143         begin
144            data = state.regs[opcode.ra] + 1;
145            calcCarryFlag(data);
146            state.regs[opcode.ra] = data;
147            calcZeroFlag(state.regs[opcode.ra]);
148         end
149
150         Dec:
151         begin
152            data = state.regs[opcode.ra] - 1;
153            calcCarryFlag(data);
154            state.regs[opcode.ra] = data;
155            calcZeroFlag(state.regs[opcode.ra]);
156         end
157
158         Shl:
159         begin
160            if (state.regs[opcode.ra][$size(pkgProl16::data_v) - 1] == 1)
161              state.cFlag = 1;
162            else
163              state.cFlag = 0;
164
165            state.regs[opcode.ra] = state.regs[opcode.ra] << 1;
166            calcZeroFlag(state.regs[opcode.ra]);
167         end
```

```
168
169          Shr:
170          begin
171                if (state.regs[opcode.ra][0] == 1)
172               state.cFlag = 1;
173             else
174               state.cFlag = 0;
175
176                state.regs[opcode.ra] = state.regs[opcode.ra] >> 1;
177             calcZeroFlag(state.regs[opcode.ra]);
178          end
179
180          Shlc:
181          begin
182            if (state.regs[opcode.ra][$size(pkgProl16::data_v) - 1] == 1)
183               tmpCFlag = 1;
184             else
185               tmpCFlag = 0;
186
187             state.regs[opcode.ra] = state.regs[opcode.ra] << 1;
188             state.regs[opcode.ra][0] = state.cFlag;
189             state.cFlag = tmpCFlag;
190             calcZeroFlag(state.regs[opcode.ra]);
191          end
192
193          Shrc:
194          begin
195            if (state.regs[opcode.ra][0] == 1)
196               tmpCFlag = 1;
197             else
198               tmpCFlag = 0;
199
200             state.regs[opcode.ra] = state.regs[opcode.ra] >> 1;
201             state.regs[opcode.ra][$size(pkgProl16::data_v) - 1] = state.
                   cFlag;
202             state.cFlag = tmpCFlag;
203             calcZeroFlag(state.regs[opcode.ra]);
204          end
205
206          default : $display("Wrong opcode: doing nop");
207       endcase
208       end
209    endtask
210 endclass
```

../../src/sv/testProl16Model.sv

```
1  `include "Prol16Model.sv"
2
3  class TestClass;
4     task assertWithoutFlags(int expectedRegisterValue, int expectedPCValue,
          Prol16State state, int register, string text);
5        assert (expectedRegisterValue == state.regs[register])
6        else $error("Expected Register Value: %d, Actual Register Value: %d,
             Info: %s", expectedRegisterValue, state.regs[register], text);
7
8        assert (expectedPCValue == state.programCounter)
9        else $error("Expected PC Value: %d, Actual PC Value: %d, Info: %s",
             expectedPCValue, state.programCounter, text);
10    endtask
11
```

```
12    task assertWithFlags(int expectedRegisterValue, int expectedPCValue, int
          expectedCarryFlag, int expectedZeroFlag, Prol16State state, int
         register, string text);
13       assert (expectedRegisterValue == state.regs[register])
14       else $error("Expected Register Value: %d, Actual Register Value: %d,
            Info: %s", expectedRegisterValue, state.regs[register], text);
15
16       assert (expectedPCValue == state.programCounter)
17       else $error("Expected PC Value: %d, Actual PC Value: %d, Info: %s",
            expectedPCValue, state.programCounter, text);
18
19       assert (expectedCarryFlag == state.cFlag)
20       else $error("Expected Carry Flag Value: %d, Actual Carry Flag Value:
            %d, Info: %s", expectedCarryFlag, state.cFlag, text);
21
22       assert (expectedZeroFlag == state.zFlag)
23       else $error("Expected Zero Flag Value: %d, Actual Zero Flag Value: %d
            , Info: %s", expectedZeroFlag, state.zFlag, text);
24    endtask
25
26    task assertWithDuv(Prol16State state, int register, logic [15:0] cpuRegs
          [31:0], logic[15:0] cpuPc, logic cpuCFlag, logic cpuZFlag, string
         text);
27       assert (state.regs[register] == cpuRegs[register])
28       else $error("WithDuv: Expected Register Value: %d, Actual Register
            Value: %d, Info: %s", state.regs[register], cpuRegs[register],
            text);
29
30       assert (state.programCounter == cpuPc)
31       else $error("WithDuv: Expected PC Value: %d, Actual PC Value: %d,
            Info: %s", state.programCounter, cpuPc, text);
32
33       assert (state.cFlag == cpuCFlag)
34       else $error("WithDuv: Expected Carry Flag Value: %d, Actual Carry
            Flag Value: %d, Info: %s", state.cFlag, cpuCFlag, text);
35
36       assert (state.zFlag == cpuZFlag)
37       else $error("WithDuv: Expected Zero Flag Value: %d, Actual Zero Flag
            Value: %d, Info: %s", state.zFlag, cpuZFlag, text);
38    endtask
39 endclass
40
41 program testProl16Model(ifProl16.master cpu, output logic rst, input logic
       clk);
42    logic [15:0] cpuRegs [31:0];
43    logic [15:0] cpuPc;
44    logic cpuCFlag;
45    logic cpuZFlag;
46
47    event CommandStart;
48    event End;
49
50    Prol16Opcode opcode = new(0, 0, Nop, 0);
51
52    bit LoadiOccurred = 0;
53
54    task trigger();
55       while (!End.triggered) begin
56         @(negedge cpu.mem_oe_n)
57         begin
```

```
58            $display("negEdge oe");
59           if (LoadiOccurred == 0)
60           begin
61              cpu.mem_data_tb[15:10] <= opcode.cmd;
62              cpu.mem_data_tb[9:5] <= opcode.ra;
63              cpu.mem_data_tb[4:0] <= opcode.rb;
64
65              if (opcode.cmd == Loadi)
66              begin
67                 LoadiOccurred = 1;
68              end
69           end
70           else
71           begin
72              cpu.mem_data_tb <= opcode.data;
73              LoadiOccurred = 0;
74           end
75          end
76          @(posedge cpu.mem_oe_n)
77          begin
78            $display("posEdge oe");
79           -> CommandStart;
80          end
81       end
82    endtask
83
84    initial begin : stimuli
85       Prol16State state = new();
86       Prol16Model#(32) model = new(state);
87
88       TestClass testClass = new();
89
90       Prol16Opcode opcode_Nop = new(0, 0, Nop, 0);
91       Prol16Opcode opcode_Loadi = new(0, 3, Loadi, 50);
92       Prol16Opcode opcode_Loadi2 = new(1, 3, Loadi, 20);
93       Prol16Opcode opcode_Jump = new(0, 0, Jump, 0);
94       Prol16Opcode opcode_Jumpc = new(0, 0, Jumpc, 0);
95       Prol16Opcode opcode_Jumpz = new(1, 0, Jumpz, 0);
96       Prol16Opcode opcode_Move = new(2, 0, Move, 0);
97       Prol16Opcode opcode_Move2 = new(2, 1, Move, 0);
98       Prol16Opcode opcode_And = new(2, 0, And, 0);
99       Prol16Opcode opcode_And2 = new(2, 3, And, 0);
100      Prol16Opcode opcode_Or = new(2, 4, Or, 0);
101      Prol16Opcode opcode_Or2 = new(2, 3, Or, 0);
102      Prol16Opcode opcode_Xor = new(2, 3, Xor, 0);
103      Prol16Opcode opcode_Xor2 = new(3, 4, Xor, 0);
104      Prol16Opcode opcode_Not = new(3, 7, Not, 0);
105      Prol16Opcode opcode_Not2 = new(5, 8, Not, 0);
106      Prol16Opcode opcode_Add = new(6, 7, Add, 0);
107      Prol16Opcode opcode_Addc = new(8, 9, Addc, 0);
108      Prol16Opcode opcode_Sub = new(10, 11, Sub, 0);
109      Prol16Opcode opcode_Subc = new(12, 13, Subc, 0);
110      Prol16Opcode opcode_Comp = new(14, 15, Comp, 0);
111      Prol16Opcode opcode_Inc = new(16, 17, Inc, 0);
112      Prol16Opcode opcode_Dec = new(18, 19, Dec, 0);
113      Prol16Opcode opcode_Shl = new(20, 21, Shl, 0);
114      Prol16Opcode opcode_Shr = new(22, 23, Shr, 0);
115      Prol16Opcode opcode_Shlc = new(24, 25, Shlc, 0);
116      Prol16Opcode opcode_Shrc = new(26, 27, Shrc, 0);
117      Prol16Opcode opcode_Invalid = new(28, 29, Invalid, 0);
```

```
118         Prol16Opcode opcode_InvalidRegister = new(31, 32, Move, 0);
119
120
121         $init_signal_spy("/top/TheCpu/datapath_inst/thereg_file/registers(0)"
                , "/top/TheTest/cpuRegs(0)");
122         $init_signal_spy("/top/TheCpu/datapath_inst/thereg_file/registers(1)"
                , "/top/TheTest/cpuRegs(1)");
123         $init_signal_spy("/top/TheCpu/datapath_inst/thereg_file/registers(2)"
                , "/top/TheTest/cpuRegs(2)");
124         $init_signal_spy("/top/TheCpu/datapath_inst/thereg_file/registers(3)"
                , "/top/TheTest/cpuRegs(3)");
125         $init_signal_spy("/top/TheCpu/datapath_inst/thereg_file/registers(4)"
                , "/top/TheTest/cpuRegs(4)");
126         $init_signal_spy("/top/TheCpu/datapath_inst/thereg_file/registers(5)"
                , "/top/TheTest/cpuRegs(5)");
127         $init_signal_spy("/top/TheCpu/datapath_inst/thereg_file/registers(6)"
                , "/top/TheTest/cpuRegs(6)");
128         $init_signal_spy("/top/TheCpu/datapath_inst/thereg_file/registers(7)"
                , "/top/TheTest/cpuRegs(7)");
129         $init_signal_spy("/top/TheCpu/datapath_inst/thereg_file/registers(8)"
                , "/top/TheTest/cpuRegs(8)");
130         $init_signal_spy("/top/TheCpu/datapath_inst/thereg_file/registers(9)"
                , "/top/TheTest/cpuRegs(9)");
131         $init_signal_spy("/top/TheCpu/datapath_inst/thereg_file/registers(10)
                ", "/top/TheTest/cpuRegs(10)");
132         $init_signal_spy("/top/TheCpu/datapath_inst/thereg_file/registers(11)
                ", "/top/TheTest/cpuRegs(11)");
133         $init_signal_spy("/top/TheCpu/datapath_inst/thereg_file/registers(12)
                ", "/top/TheTest/cpuRegs(12)");
134         $init_signal_spy("/top/TheCpu/datapath_inst/thereg_file/registers(13)
                ", "/top/TheTest/cpuRegs(13)");
135         $init_signal_spy("/top/TheCpu/datapath_inst/thereg_file/registers(14)
                ", "/top/TheTest/cpuRegs(14)");
136         $init_signal_spy("/top/TheCpu/datapath_inst/thereg_file/registers(15)
                ", "/top/TheTest/cpuRegs(15)");
137         $init_signal_spy("/top/TheCpu/datapath_inst/thereg_file/registers(16)
                ", "/top/TheTest/cpuRegs(16)");
138         $init_signal_spy("/top/TheCpu/datapath_inst/thereg_file/registers(17)
                ", "/top/TheTest/cpuRegs(17)");
139         $init_signal_spy("/top/TheCpu/datapath_inst/thereg_file/registers(18)
                ", "/top/TheTest/cpuRegs(18)");
140         $init_signal_spy("/top/TheCpu/datapath_inst/thereg_file/registers(19)
                ", "/top/TheTest/cpuRegs(19)");
141         $init_signal_spy("/top/TheCpu/datapath_inst/thereg_file/registers(20)
                ", "/top/TheTest/cpuRegs(20)");
142         $init_signal_spy("/top/TheCpu/datapath_inst/thereg_file/registers(21)
                ", "/top/TheTest/cpuRegs(21)");
143         $init_signal_spy("/top/TheCpu/datapath_inst/thereg_file/registers(22)
                ", "/top/TheTest/cpuRegs(22)");
144         $init_signal_spy("/top/TheCpu/datapath_inst/thereg_file/registers(23)
                ", "/top/TheTest/cpuRegs(23)");
145         $init_signal_spy("/top/TheCpu/datapath_inst/thereg_file/registers(24)
                ", "/top/TheTest/cpuRegs(24)");
146         $init_signal_spy("/top/TheCpu/datapath_inst/thereg_file/registers(25)
                ", "/top/TheTest/cpuRegs(25)");
147         $init_signal_spy("/top/TheCpu/datapath_inst/thereg_file/registers(26)
                ", "/top/TheTest/cpuRegs(26)");
148         $init_signal_spy("/top/TheCpu/datapath_inst/thereg_file/registers(27)
                ", "/top/TheTest/cpuRegs(27)");
```

```
149         $init_signal_spy("/top/TheCpu/datapath_inst/thereg_file/registers(28)
                ", "/top/TheTest/cpuRegs(28)");
150         $init_signal_spy("/top/TheCpu/datapath_inst/thereg_file/registers(29)
                ", "/top/TheTest/cpuRegs(29)");
151         $init_signal_spy("/top/TheCpu/datapath_inst/thereg_file/registers(30)
                ", "/top/TheTest/cpuRegs(30)");
152         $init_signal_spy("/top/TheCpu/datapath_inst/thereg_file/registers(31)
                ", "/top/TheTest/cpuRegs(31)");
153
154         $init_signal_spy("/top/TheCpu/datapath_inst/RegPC", "/top/TheTest/
                cpuPc");
155         $init_signal_spy("/top/TheCpu/control_inst/Carry", "/top/TheTest/
                cpuCFlag");
156         $init_signal_spy("/top/TheCpu/control_inst/Zero", "/top/TheTest/
                cpuZFlag");
157
158         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(0)", "
                16#0000", 0, 1);
159         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(1)", "
                16#0000", 0, 1);
160         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(2)", "
                16#0000", 0, 1);
161         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(3)", "
                16#0000", 0, 1);
162         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(4)", "
                16#0000", 0, 1);
163         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(5)", "
                16#0000", 0, 1);
164         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(6)", "
                16#0000", 0, 1);
165         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(7)", "
                16#0000", 0, 1);
166         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(8)", "
                16#0000", 0, 1);
167         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(9)", "
                16#0000", 0, 1);
168         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(10)",
                "16#0000", 0, 1);
169         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(11)",
                "16#0000", 0, 1);
170         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(12)",
                "16#0000", 0, 1);
171         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(13)",
                "16#0000", 0, 1);
172         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(14)",
                "16#0000", 0, 1);
173         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(15)",
                "16#0000", 0, 1);
174         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(16)",
                "16#0000", 0, 1);
175         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(17)",
                "16#0000", 0, 1);
176         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(18)",
                "16#0000", 0, 1);
177         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(19)",
                "16#0000", 0, 1);
178         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(20)",
                "16#0000", 0, 1);
179         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(21)",
                "16#0000", 0, 1);
```

```
180         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(22)",
                "16#0000", 0, 1);
181         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(23)",
                "16#0000", 0, 1);
182         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(24)",
                "16#0000", 0, 1);
183         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(25)",
                "16#0000", 0, 1);
184         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(26)",
                "16#0000", 0, 1);
185         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(27)",
                "16#0000", 0, 1);
186         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(28)",
                "16#0000", 0, 1);
187         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(29)",
                "16#0000", 0, 1);
188         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(30)",
                "16#0000", 0, 1);
189         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(31)",
                "16#0000", 0, 1);
190
191         //Reset
192         model.reset();
193         //testClass.assertWithoutFlags(0, 0, model.state, 12, "Reset test");
194
195         // generate reset
                ----------------------------------------------------
196         rst = 1;
197         #10 rst = 0;
198         #20 rst = 1;
199
200         fork
201           trigger();
202         join_none
203
204
205         //Nop
206         @(CommandStart);
207         testClass.assertWithDuv(model.state, 12, cpuRegs, cpuPc, cpuCFlag,
                cpuZFlag, "Reset test");
208
209         model.execute(opcode);
210         opcode = opcode_Loadi;
211         @(CommandStart);
212         testClass.assertWithDuv(model.state, 0, cpuRegs, cpuPc, cpuCFlag,
                cpuZFlag, "Nop test");
213
214         @(CommandStart);
215
216         //Loadi
217         model.execute(opcode);
218         opcode = opcode_Loadi2;
219         @(CommandStart);
220         testClass.assertWithDuv(model.state, 0, cpuRegs, cpuPc, cpuCFlag,
                cpuZFlag, "Loadi test 1");
221         testClass.assertWithoutFlags(50, 3, model.state, 0, "Loadi test 1");
222
223         @(CommandStart);
224
225         model.execute(opcode);
```

```
226         opcode = opcode_Jump;
227         @(CommandStart);
228         testClass.assertWithDuv(model.state, 1, cpuRegs, cpuPc, cpuCFlag,
                cpuZFlag, "Loadi test 2");
229         testClass.assertWithoutFlags(20, 5, model.state, 1, "Loadi test 2");
230
231         //Jump
232         model.execute(opcode);
233         opcode = opcode_Jumpc;
234         @(CommandStart);
235         testClass.assertWithDuv(model.state, 0, cpuRegs, cpuPc, cpuCFlag,
                cpuZFlag, "Jump test");
236         testClass.assertWithoutFlags(50, 50, model.state, 0, "Jump test");
237
238         //Jumpc
239         model.execute(opcode_Jumpc);
240         opcode = opcode_Jumpc;
241         @(CommandStart);
242         testClass.assertWithDuv(model.state, 0, cpuRegs, cpuPc, cpuCFlag,
                cpuZFlag, "Jumpc test");
243         testClass.assertWithoutFlags(50, 51, model.state, 0, "Jumpc test
                Carry = 0");
244
245         model.state.cFlag = 1;
246         $signal_force("/top/TheCpu/control_inst/Carry", "1", 0, 1);
247         model.execute(opcode_Jumpc);
248         opcode = opcode_Jumpz;
249         @(CommandStart);
250         testClass.assertWithDuv(model.state, 0, cpuRegs, cpuPc, cpuCFlag,
                cpuZFlag, "Jumpc test Carry = 1");
251         testClass.assertWithoutFlags(50, 50, model.state, 0, "Jumpc test
                Carry = 1");
252
253         //Jumpz
254         model.execute(opcode_Jumpz);
255         opcode = opcode_Jumpz;
256         @(CommandStart);
257         testClass.assertWithDuv(model.state, 1, cpuRegs, cpuPc, cpuCFlag,
                cpuZFlag, "Jumpz test Zero = 0");
258         testClass.assertWithoutFlags(20, 51, model.state, 1, "Jumpz test Zero
                 = 0");
259
260         model.state.zFlag = 1;
261         $signal_force("/top/TheCpu/control_inst/Zero", "1", 0, 1);
262         model.execute(opcode_Jumpz);
263         opcode = opcode_Move;
264         @(CommandStart);
265         testClass.assertWithDuv(model.state, 1, cpuRegs, cpuPc, cpuCFlag,
                cpuZFlag, "Jumpz test Zero = 1");
266         testClass.assertWithoutFlags(20, 20, model.state, 1, "Jumpz test Zero
                 = 1");
267
268         //Move
269         model.execute(opcode_Move);
270         opcode = opcode_Move2;
271         @(CommandStart);
272         testClass.assertWithDuv(model.state, 2, cpuRegs, cpuPc, cpuCFlag,
                cpuZFlag, "Move test 1");
273         testClass.assertWithoutFlags(50, 21, model.state, 2, "Move test 1");
274
```

```
275         model.execute(opcode_Move2);
276         opcode = opcode_And;
277         @(CommandStart);
278         testClass.assertWithDuv(model.state, 2, cpuRegs, cpuPc, cpuCFlag,
                cpuZFlag, "Move test 2");
279         testClass.assertWithoutFlags(20, 22, model.state, 2, "Move test 2");
280
281         //And
282         model.state.regs[3] = 64;
283         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(3)", "
                16#0040", 0, 1);
284         model.execute(opcode_And);
285         opcode = opcode_And2;
286         @(CommandStart);
287         testClass.assertWithDuv(model.state, 2, cpuRegs, cpuPc, cpuCFlag,
                cpuZFlag, "And test 1");
288         testClass.assertWithFlags(16, 23, 0, 0, model.state, 2, "And test 1")
                ;
289
290         model.execute(opcode_And2);
291         opcode = opcode_Or;
292         @(CommandStart);
293         testClass.assertWithDuv(model.state, 2, cpuRegs, cpuPc, cpuCFlag,
                cpuZFlag, "And test 2");
294         testClass.assertWithFlags(0, 24, 0, 1, model.state, 2, "And test 2");
295
296         //Or
297         model.execute(opcode_Or);
298         opcode = opcode_Or2;
299         @(CommandStart);
300         testClass.assertWithDuv(model.state, 2, cpuRegs, cpuPc, cpuCFlag,
                cpuZFlag, "Or test 1");
301         testClass.assertWithFlags(0, 25, 0, 1, model.state, 2, "Or test 1");
302
303         model.execute(opcode_Or2);
304         opcode = opcode_Xor;
305         @(CommandStart);
306         testClass.assertWithDuv(model.state, 2, cpuRegs, cpuPc, cpuCFlag,
                cpuZFlag, "Or test 2");
307         testClass.assertWithFlags(64, 26, 0, 0, model.state, 2, "Or test 2");
308
309         //Xor
310         model.execute(opcode_Xor);
311         opcode = opcode_Xor2;
312         @(CommandStart);
313         testClass.assertWithDuv(model.state, 2, cpuRegs, cpuPc, cpuCFlag,
                cpuZFlag, "Xor test 1");
314         testClass.assertWithFlags(0, 27, 0, 1, model.state, 2, "Xor test 1");
315
316         model.state.regs[4] = 96;
317         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(3)", "
                16#0060", 0, 1);
318         model.execute(opcode_Xor2);
319         opcode = opcode_Not;
320         @(CommandStart);
321         testClass.assertWithDuv(model.state, 3, cpuRegs, cpuPc, cpuCFlag,
                cpuZFlag, "Xor test 2");
322         testClass.assertWithFlags(32, 28, 0, 0, model.state, 3, "Xor test 2")
                ;
323
```

```
324         //Not
325         model.execute(opcode_Not);
326         opcode = opcode_Not2;
327         @(CommandStart);
328         testClass.assertWithDuv(model.state, 3, cpuRegs, cpuPc, cpuCFlag,
               cpuZFlag, "Not test 1");
329         testClass.assertWithFlags(65503, 29, 0, 0, model.state, 3, "Not test
               1");
330
331         model.state.regs[5] = 65535;
332         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(3)", "
               16#FFFF", 0, 1);
333         model.execute(opcode_Not2);
334         opcode = opcode_Add;
335         @(CommandStart);
336         testClass.assertWithDuv(model.state, 5, cpuRegs, cpuPc, cpuCFlag,
               cpuZFlag, "Not test 2");
337         testClass.assertWithFlags(0, 30, 0, 1, model.state, 5, "Not test 2");
338
339         //Add
340         model.state.regs[6] = 10;
341         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(3)", "
               16#000A", 0, 1);
342         model.state.regs[7] = 30;
343         $signal_force("/top/TheCpu/datapath_inst/thereg_file/registers(3)", "
               16#001E", 0, 1);
344         model.state.cFlag = 1;
345         $signal_force("/top/TheCpu/control_inst/Carry", "1", 0, 1);
346         model.execute(opcode_Add);
347         opcode = opcode_Add;
348         @(CommandStart);
349         testClass.assertWithDuv(model.state, 6, cpuRegs, cpuPc, cpuCFlag,
               cpuZFlag, "Add test 1");
350         testClass.assertWithFlags(40, 31, 0, 0, model.state, 6, "Add test 1")
               ;
351
352         model.state.regs[6] = 65535;
353         model.state.regs[7] = 1;
354         model.execute(opcode_Add);
355         testClass.assertWithFlags(0, 32, 1, 1, model.state, 6, "Add test 2");
356
357         model.state.regs[6] = 65535;
358         model.state.regs[7] = 2;
359         model.execute(opcode_Add);
360         testClass.assertWithFlags(1, 33, 1, 0, model.state, 6, "Add test 3");
361
362         //Addc
363         model.state.regs[8] = 10;
364         model.state.regs[9] = 30;
365         model.state.cFlag = 0;
366         model.execute(opcode_Addc);
367         testClass.assertWithFlags(40, 34, 0, 0, model.state, 8, "Addc test 1"
               );
368         model.state.regs[8] = 10;
369         model.state.regs[9] = 30;
370         model.state.cFlag = 1;
371         model.execute(opcode_Addc);
372         testClass.assertWithFlags(41, 35, 0, 0, model.state, 8, "Addc test 2"
               );
373         model.state.regs[8] = 65535;
```

```
374         model.state.regs[9] = 1;
375         model.execute(opcode_Addc);
376         testClass.assertWithFlags(0, 36, 1, 1, model.state, 8, "Addc test 3")
                ;
377         model.state.regs[8] = 65535;
378         model.state.regs[9] = 1;
379         model.execute(opcode_Addc);
380         testClass.assertWithFlags(1, 37, 1, 0, model.state, 8, "Addc test 4")
                ;
381 
382         //Sub
383         model.state.regs[10] = 30;
384         model.state.regs[11] = 10;
385         model.state.cFlag = 1;
386         model.execute(opcode_Sub);
387         testClass.assertWithFlags(20, 38, 0, 0, model.state, 10, "Sub test 1"
                );
388         model.state.regs[10] = 10;
389         model.state.regs[11] = 10;
390         model.execute(opcode_Sub);
391         testClass.assertWithFlags(0, 39, 0, 1, model.state, 10, "Sub test 2")
                ;
392         model.state.regs[10] = 1;
393         model.state.regs[11] = 2;
394         model.execute(opcode_Sub);
395         testClass.assertWithFlags(65535, 40, 1, 0, model.state, 10, "Sub test
                 3");
396 
397         //Subc
398         model.state.regs[12] = 30;
399         model.state.regs[13] = 10;
400         model.state.cFlag = 0;
401         model.execute(opcode_Subc);
402         testClass.assertWithFlags(20, 41, 0, 0, model.state, 12, "Subc test 1
                ");
403         model.state.regs[12] = 30;
404         model.state.regs[13] = 10;
405         model.state.cFlag = 1;
406         model.execute(opcode_Subc);
407         testClass.assertWithFlags(19, 42, 0, 0, model.state, 12, "Subc test 2
                ");
408         model.state.regs[12] = 10;
409         model.state.regs[13] = 10;
410         model.state.cFlag = 0;
411         model.execute(opcode_Subc);
412         testClass.assertWithFlags(0, 43, 0, 1, model.state, 12, "Subc test 3"
                );
413         model.state.regs[12] = 1;
414         model.state.regs[13] = 2;
415         model.execute(opcode_Subc);
416         testClass.assertWithFlags(65535, 44, 1, 0, model.state, 12, "Subc
                test 4");
417 
418         //Comp
419         model.state.regs[14] = 30;
420         model.state.regs[15] = 10;
421         model.execute(opcode_Comp);
422         testClass.assertWithFlags(30, 45, 0, 0, model.state, 14, "Comp test 1
                ");
423         model.state.regs[14] = 30;
```

```
424         model.state.regs[15] = 10;
425         model.state.cFlag = 1;
426         model.execute(opcode_Comp);
427         testClass.assertWithFlags(30, 46, 0, 0, model.state, 14, "Comp test 2
                ");
428         model.state.regs[14] = 10;
429         model.state.regs[15] = 10;
430         model.execute(opcode_Comp);
431         testClass.assertWithFlags(10, 47, 0, 1, model.state, 14, "Comp test 3
                ");
432         model.state.regs[14] = 1;
433         model.state.regs[15] = 2;
434         model.execute(opcode_Comp);
435         testClass.assertWithFlags(1, 48, 1, 0, model.state, 14, "Comp test 4"
                );
436 
437         //Inc
438         model.state.regs[16] = 30;
439         model.execute(opcode_Inc);
440         testClass.assertWithFlags(31, 49, 0, 0, model.state, 16, "Inc test 1"
                );
441         model.state.regs[16] = 30;
442         model.state.cFlag = 1;
443         model.execute(opcode_Inc);
444         testClass.assertWithFlags(31, 50, 0, 0, model.state, 16, "Inc test 2"
                );
445         model.state.regs[16] = 0;
446         model.execute(opcode_Inc);
447         testClass.assertWithFlags(1, 51, 0, 0, model.state, 16, "Inc test 3")
                ;
448         model.state.regs[16] = 65535;
449         model.execute(opcode_Inc);
450         testClass.assertWithFlags(0, 52, 1, 1, model.state, 16, "Inc test 4")
                ;
451 
452         //Dec
453         model.state.regs[18] = 30;
454         model.execute(opcode_Dec);
455         testClass.assertWithFlags(29, 53, 0, 0, model.state, 18, "Dec test 1"
                );
456         model.state.regs[18] = 30;
457         model.state.cFlag = 1;
458         model.execute(opcode_Dec);
459         testClass.assertWithFlags(29, 54, 0, 0, model.state, 18, "Dec test 2"
                );
460         model.state.regs[18] = 0;
461         model.execute(opcode_Dec);
462         testClass.assertWithFlags(65535, 55, 1, 0, model.state, 18, "Dec test
                 3");
463         model.state.regs[18] = 1;
464         model.execute(opcode_Dec);
465         testClass.assertWithFlags(0, 56, 0, 1, model.state, 18, "Dec test 4")
                ;
466 
467         //Shl
468         model.state.regs[20] = 0;
469         model.execute(opcode_Shl);
470         testClass.assertWithFlags(0, 57, 0, 1, model.state, 20, "Shl test 1")
                ;
471         model.state.regs[20] = 1;
```

```
472         model.state.cFlag = 1;
473         model.execute(opcode_Shl);
474         testClass.assertWithFlags(2, 58, 0, 0, model.state, 20, "Shl test 2")
                ;
475         model.state.regs[20] = 65535;
476         model.execute(opcode_Shl);
477         testClass.assertWithFlags(65534, 59, 1, 0, model.state, 20, "Shl test
                 3");
478         model.state.regs[20] = 32768;
479         model.execute(opcode_Shl);
480         testClass.assertWithFlags(0, 60, 1, 1, model.state, 20, "Shl test 4")
                ;
481
482         //Shr
483         model.state.regs[22] = 0;
484         model.execute(opcode_Shr);
485         testClass.assertWithFlags(0, 61, 0, 1, model.state, 22, "Shr test 1")
                ;
486         model.state.regs[22] = 2;
487         model.state.cFlag = 1;
488         model.execute(opcode_Shr);
489         testClass.assertWithFlags(1, 62, 0, 0, model.state, 22, "Shr test 2")
                ;
490         model.state.regs[22] = 65535;
491         model.execute(opcode_Shr);
492         testClass.assertWithFlags(32767, 63, 1, 0, model.state, 22, "Shr test
                 3");
493         model.state.regs[22] = 1;
494         model.execute(opcode_Shr);
495         testClass.assertWithFlags(0, 64, 1, 1, model.state, 22, "Shr test 4")
                ;
496
497         //Shlc
498         model.state.regs[24] = 0;
499         model.state.cFlag = 0;
500         model.execute(opcode_Shlc);
501         testClass.assertWithFlags(0, 65, 0, 1, model.state, 24, "Shlc test 1"
                );
502         model.state.regs[24] = 1;
503         model.state.cFlag = 1;
504         model.execute(opcode_Shlc);
505         testClass.assertWithFlags(3, 66, 0, 0, model.state, 24, "Shlc test 2"
                );
506         model.state.regs[24] = 65535;
507         model.execute(opcode_Shlc);
508         testClass.assertWithFlags(65534, 67, 1, 0, model.state, 24, "Shlc
                test 3");
509         model.state.regs[24] = 32768;
510         model.state.cFlag = 0;
511         model.execute(opcode_Shlc);
512         testClass.assertWithFlags(0, 68, 1, 1, model.state, 24, "Shlc test 4"
                );
513
514         //Shrc
515         model.state.regs[26] = 0;
516         model.state.cFlag = 0;
517         model.execute(opcode_Shrc);
518         testClass.assertWithFlags(0, 69, 0, 1, model.state, 26, "Shrc test 1"
                );
519         model.state.regs[26] = 2;
```

```
520         model.state.cFlag = 1;
521         model.execute(opcode_Shrc);
522         testClass.assertWithFlags(32769, 70, 0, 0, model.state, 26, "Shrc
               test 2");
523         model.state.regs[26] = 65535;
524         model.execute(opcode_Shrc);
525         testClass.assertWithFlags(32767, 71, 1, 0, model.state, 26, "Shrc
               test 3");
526         model.state.regs[26] = 1;
527         model.state.cFlag = 0;
528         model.execute(opcode_Shrc);
529         testClass.assertWithFlags(0, 72, 1, 1, model.state, 26, "Shrc test 4"
               );
530
531         //Invalid
532         model.state.regs[28] = 123;
533         model.state.cFlag = 0;
534         model.state.zFlag = 0;
535         model.execute(opcode_Invalid);
536         testClass.assertWithFlags(123, 73, 0, 0, model.state, 28, "Invalid
               test");
537
538         //InvalidRegister
539         model.state.regs[31] = 1234;
540         model.execute(opcode_InvalidRegister);
541         testClass.assertWithFlags(1234, 74, 0, 0, model.state, 31, "Invalid
               register test");
542
543         -> End;
544         $stop;
545      end : stimuli
546   endprogram
```

../../src/sv/top.sv

```
1   `include "testProl16Model.sv"
2
3   module top;
4      logic clk = 0, rst;
5      always #10 clk = ~clk;
6
7      ifProl16 ifCpu();
8      cpu TheCpu(clk, rst, ifCpu.mem_addr, ifCpu.mem_data_cpu, ifCpu.
          mem_data_tb, ifCpu.mem_ce_n, ifCpu.mem_oe_n,
9                ifCpu.mem_we_n, ifCpu.illegal_inst, ifCpu.cpu_halt);
10     testProl16Model TheTest(ifCpu.master, rst, clk);
11  endmodule
```

../../sim/Compile.do

```
1   vlib work
2
3   vlog ../src/sv/ifProl16.sv
4   vlog ../src/sv/pkgProl16.sv
5   vlog ../src/sv/Prol16Command.sv
6   vlog ../src/sv/Prol16Opcode.sv +incdir+../src/sv
7   vlog ../src/sv/Prol16State.sv +incdir+../src/sv
8   vlog ../src/sv/Prol16Model.sv +incdir+../src/sv
9   vlog ../src/sv/testProl16Model.sv +incdir+../src/sv
10  vlog ../src/sv/top.sv +incdir+../src/sv
11
12  vcom -87 ../src/vhdl/prol16_pack.vhd
```

```
13 vcom -87 ../src/vhdl/reg_file.vhd
14 vcom -87 ../src/vhdl/datapath.vhd
15 vcom -87 ../src/vhdl/control.vhd
16 vcom -87 ../src/vhdl/alu.vhd
17 vcom -93 ../src/vhdl/memory.vhd
18 vcom -87 ../src/vhdl/cpu.vhd
19 #vcom -87 ../src/vhdl/cpu_tb.vhd
```

../../sim/Sim.do

```
1 vsim -novopt top
2 run -all
```