

# Computer system III Lab 1 Report

3200105787 张云策

## 一.实验目的

- 回顾 OS 在硬件层面的抽象
- 回顾硬件设计，仿真，调试方法
- 在硬件上运行简易 ELF 程序: Naive Kernel

## 二.实验模块

CSR设计模块：

```
1  module CSR(  
2      input clk,  
3      input rst,  
4      input [31:0] inst,  
5      input [6:0] op_code,  
6      input [2:0] funct3,  
7      input [11:0] csr_addr,  
8      input [4:0] zimm,  
9      input [31:0] csr_din,  
10     input [31:0] pc_out,  
11     output reg [31:0] csr_dout,  
12     output reg [31:0] csr_pc_next,  
13     output reg csr_pc_write,  
14     output reg csr_write  
15 );  
16     reg [31:0] mtvec;  
17     reg [31:0] mepc;  
18     reg [31:0] mstatus;  
19  
20     initial begin  
21         mtvec<=0;  
22         mepc<=0;  
23         mstatus<=0;  
24         csr_pc_next<=0;  
25         csr_write<=0;  
26     end  
27     always @(*) begin  
28         csr_write=0;  
29         if (rst == 1) begin  
30             csr_write=0;  
31         end  
32         else if (inst==32'h00000073 || inst==32'h00100073)begin//ECALL  
33             csr_write = 1;  
34         end else if (inst==32'h30200073)begin  
35             csr_write = 1;  
36         end  
37         else begin  
38             if (inst[6:0]==7'b1110011) begin  
39                 csr_write =1;  
40             end  
41         end  
42     end
```

```

41     end
42     end
43
44
45
46     always @(posedge clk or posedge rst)begin
47         if(rst==1)begin
48             mtvec<=0;
49             mepc<=0;
50             mstatus<=0;
51         end
52
53     else begin
54         if (op_code==7'b1110011) begin
55             if (csr_addr == 12'h300) begin
56                 case(funct3)
57                     3'b011:begin
58                         mstatus <= mstatus & csr_din;
59                     end
60                     3'b111:begin
61                         mstatus <= mstatus & {27'b0,zimm};
62                     end
63                     3'b010:begin
64                         mstatus <= mstatus | csr_din;
65                     end
66                     3'b110:begin
67                         mstatus <= mstatus | {27'b0,zimm};
68                     end
69                     3'b001:begin
70                         mstatus <= csr_din;
71                     end
72                     3'b101:begin
73                         mstatus <= {27'b0,zimm};
74                     end
75                 endcase
76             end
77         else if(csr_addr == 12'h305)begin
78             case(funct3)
79                 3'b011:begin
80                     mtvec <= mtvec & csr_din;
81                 end
82                 3'b111:begin
83                     mtvec <= mtvec & {27'b0,zimm};
84                 end
85                 3'b010:begin
86                     mtvec <= mtvec | csr_din;
87                 end
88                 3'b110:begin
89                     mtvec <= mtvec | {27'b0,zimm};
90                 end
91                 3'b001:begin
92                     mtvec <= csr_din;
93                 end
94                 3'b101:begin
95                     mtvec <= {27'b0,zimm};
96                 end
97             endcase
98         end

```

```

99         else if(csr_addr == 12'h341)begin
100             case(func3)
101                 3'b011:begin
102                     mepc <= mepc & csr_din;
103                 end
104                 3'b111:begin
105                     mepc <= mepc & {27'b0,zimm};
106                 end
107                 3'b010:begin
108                     mepc <= mepc | csr_din;
109                 end
110                 3'b110:begin
111                     mepc <= mepc | {27'b0,zimm};
112                 end
113                 3'b001:begin
114                     mepc <= csr_din;
115                 end
116                 3'b101:begin
117                     mepc <= {27'b0,zimm};
118                 end
119             endcase
120         end
121     else if(csr_addr == 12'h000 || csr_addr == 12'h001)begin
mepc<=pc_out;end
122     end
123 end
124 end
125
126 always @(*)begin
127     csr_pc_write=0;
128     if(op_code==7'b1110011)begin
129         if(csr_addr==12'h302)begin
130             csr_pc_next=mepc;
131             csr_pc_write=1;
132         end
133     else if(csr_addr==12'h000 || csr_addr==12'h001)begin
134         csr_pc_next=mtvec;
135         csr_pc_write=1;
136     end
137 end
138 end
139
140     always @(*) begin
141
142         if (rst == 1) begin
143             csr_dout<=0;
144         end
145
146         else begin
147             if (inst[6:0]==7'b1110011) begin
148                 if (inst[31:20] == 12'h300) begin
149                     csr_dout <= mstatus;
150                 end
151             else if(inst[31:20] == 12'h305)begin
152                 csr_dout <= mtvec;
153             end
154             else if(inst[31:20] == 12'h341)begin
155                 csr_dout <= mepc;
156             end

```

```
156         end
157     end
158 end
159 endmodule
```

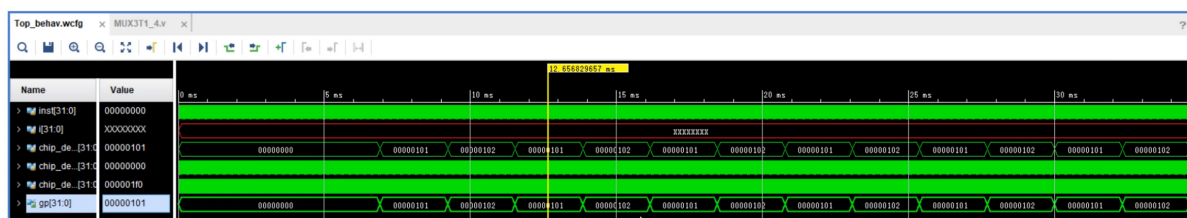
Forward选择器模块:

```
1  module Forward_mux(
2      input [31:0] I0,
3      input [31:0] I1,
4      input [31:0] I2,
5      input [31:0] I3,
6      input e,
7      input [1:0] s,
8      output reg [31:0] o
9  );
10  always @(*) begin
11      if(e!=1)
12          case(s)
13              2'b00: o = I0;
14              2'b01: o = I1;
15              2'b10: o = I2;
16              2'b11: o = I3;
17              default: ;
18          endcase
19      else o=I0;
20  end
21 endmodule
```

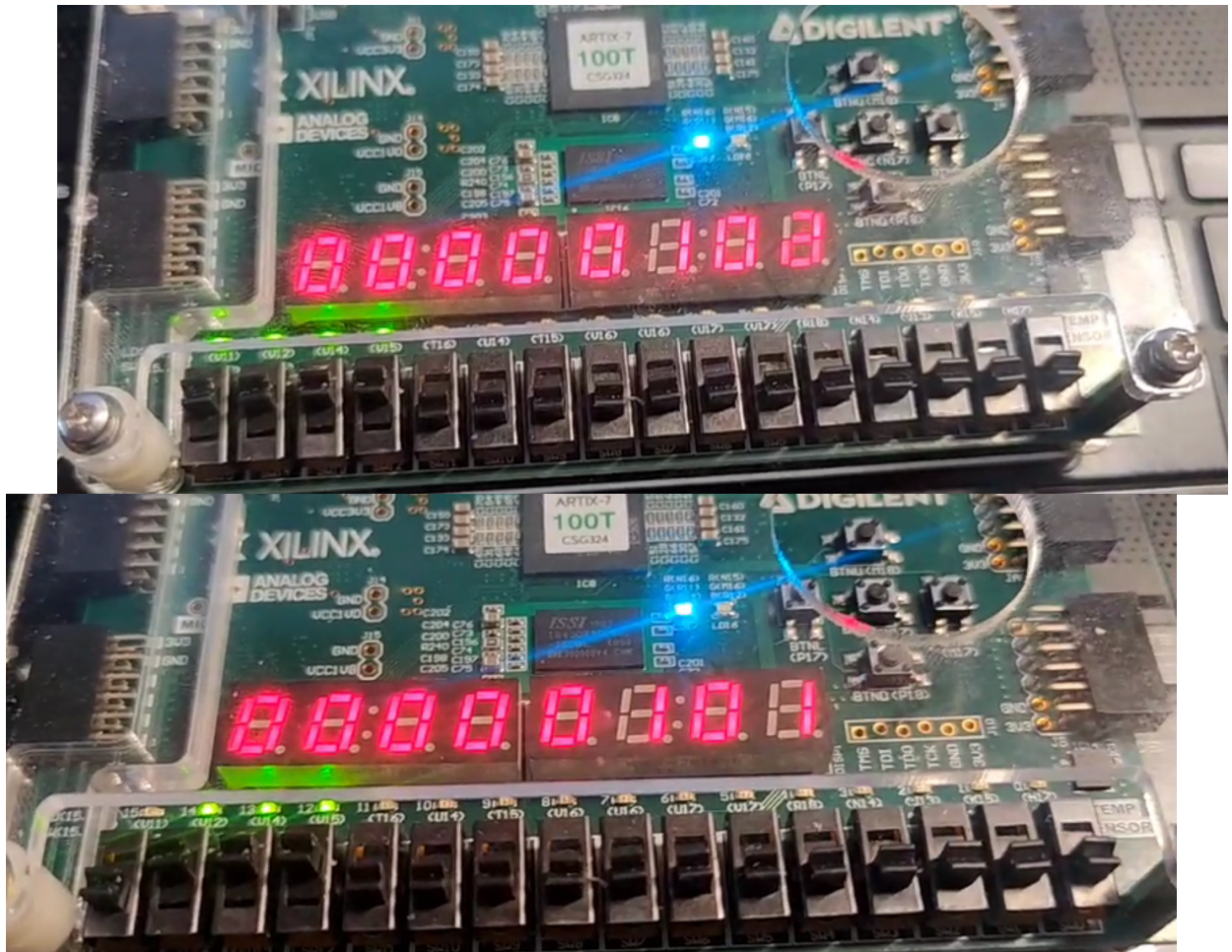
仿真:

- Design Sources (5)
    - Verilog Header (1)
    - Top (Top.sv) (2)
    - Core\_tb (Core\_tb.sv) (1)
      - uut : Core (Core.v) (3)
        - cpu : SCPU (SCPU.v) (2)
          - control : Control (Control.v)
          - datapath : Datapath (Datapath.v) (14)
            - forwarding : Forwarding\_Unit (Forwarding\_Unit.v)
            - csr : CSR (CSR.v)
            - regs : Regs (Regs.v)
            - imm\_gen : ImmGen (ImmGen.v)
            - mux5 : MUX2T1\_32 (MUX2T1\_32.v)
            - mux1 : MUX2T1\_32 (MUX2T1\_32.v)
            - mux6 : MUX2T1\_32 (MUX2T1\_32.v)
            - mux7 : MUX2T1\_32 (MUX2T1\_32.v)
            - Forward1 : Forward\_mux (Forward\_mux.v)
            - Forward2 : Forward\_mux (Forward\_mux.v)
            - Forward3 : Forward\_mux (Forward\_mux.v)
            - alu : ALU (ALU.v)
            - mux3 : MUX3T1\_32 (MUX3T1\_4.v)
            - mux2 : MUX4T1\_32 (MUX4T1\_32.v)
    - rom\_unit : Rom (Rom.xci)
    - ram\_unit : Ram (Ram.xci)

仿真结果:



上板结果：



### 三.实验思考题

从硬件设计的角度看，发生异常和发生中断是否有区别？

- 中断由CPU外的中断信号触发启动，系统无法得知何时会出现中断情况；异常则是由CPU自己检测得出，由控制器产生。
- 中断发生后，需要CPU优先处理，并且需要处理后再继续进行下一任务；异常大都出现在用户态，由CPU进行同步处理。
- 中断通常不会导致异常的出现；但异常可能导致中断发生。