浙江水学

本科实验报告

课程名称:	计算机体系结构
姓 名:	
学 院:	计算机科学与技术学院
系:	计算机科学与技术系
专业:	计算机科学与技术
学 号:	
指导教师:	卜凯

2022年 10月 26日

浙江大学实验报告

课程名称:	计算机体系结构	实验类型:	综合	
-------	---------	-------	----	--

实验项目名称: Lab2: Pipelined CPU supporting exception & interrupt

学生姓名: 专业: 计算机科学与技术 学号:

同组学生姓名: 指导老师: 卜凯

实验地点: 曹西 301 实验日期: <u>2022</u>年 <u>10</u>月 <u>26</u>日

1 Tasks and requirements

1.1 Tasks

The only main task of Lab-2 is:

 Master the design methods of pipelined CPU supporting exception & interrupt.

1.2Requirements

This experiment would be based on SWORD development board, with xc7k325tffg676-2L FPGA.

We are given a Verilog project complementing a CPU core and other components for debugging on board. Most of the parts have been finished. There is only one file we need to complete, that is:

ExceptionUnit.v, in order to deal with **CSR operations** and **traps** (interrupts or exceptions).

When this work is finished, we shall verify both the simulation results and the on-board results of a preset program.

2 Contents and principles

All of the following principles are based on RISC-V 5-stage pipelined CPU.

2.1 RISC-V privilege levels

RISC-V has 3 privilege levels until now:

- 1. Machine, usually for bootloader and other hardware demands,
- 2. **Supervisor**, usually for operating systems,
- 3. **User**, usually for applications.

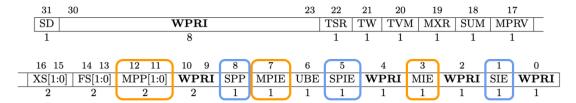
However, only Machine level is necessary; That means, Supervisor and User level may be not implemented in a system.

Switches between privilege levels are realized by **trap**. An **interrupt** or **exception** at a lower privilege level will switch the processor to a higher level to deal with that interrupt or exception. After some dealing process, the privilege level should be recovered to its original.

2.2CSR (Control and Status Registers)

Control and status registers are one kind of registers built in CPU, storing information about the running conditions of the system. Different privilege level may access different CSRs. In this experiment we mainly intervene following CSRs:

1. **mstatus** (at 0x300). As its name implies, it stores information about the status of the machine. We mainly consider xIE and xPIE bits, whose function is highly relevant to interrupts.



mstatus.xIE: Interrupt Enable in x mode mstatus.xPIE: Previous Interrupt Enable in x mode mstatus.xPP: Previous Priviledge mode up to x mode

- 2. **mtvec** (at 0x305). This register contains information about how to deal with different traps, and may contain addresses for trap handling programs.
- 3. **mcause** (at 0x342). This register contains information about the cause of the trap. It should be automatically written when trap occurs. Following is a table that defines how **mcause** should be written when entering a trap.

Interrupt	Exception Code	Description
1	0	Reserved
1	1	Supervisor software interrupt
1	2	Reserved
1	3	Machine software interrupt
1	4	Reserved
1	5	Supervisor timer interrupt
1	6	Reserved
1	7	Machine timer interrupt
1	8	Reserved
1	9	Supervisor external interrupt
1	10	Reserved
1	11	Machine external interrupt
1	12–15	Reserved
1	≥16	Designated for platform use
0	0	Instruction address misaligned
0	1	Instruction access fault
0	2	Illegal instruction
0	3	Breakpoint
0	4	Load address misaligned
0	5	Load access fault
0	6	Store/AMO address misaligned
0	7	Store/AMO access fault
0	8	Environment call from U-mode
0	9	Environment call from S-mode
0	10	Reserved
0	11	Environment call from M-mode
0	12	Instruction page fault
0	13	Load page fault
0	14	Reserved
0	15	Store/AMO page fault
0	16–23	Reserved
0	24–31	Designated for custom use
0	32–47	Reserved
0	48-63	Designated for custom use
0	≥64	Reserved

4. **mepc** (at 0x341). This register should contain the address that the trap handling programs should return to.

Obviously, when entering a trap, some CSRs should be automatically modified by the hardware. Somehow, CSRs can also be written or read by **CSR instructions** explicitly.

2.3 Privileged Instructions

There are some instructions that can only be used for a certain privilege level. In this experiment we will deal with **ecall** and **sret** instructions.

ecall instruction is used to generate a software interrupt. The user should first write some information about the interrupt in some registers to let the system run the corresponded handling program.

mret instruction is used to return to Supervisor level from Machine level. Of

course, it should change some data in the CSRs.

Likewise, there is a **sret** instruction, but we don't need to implement it in this experiment.

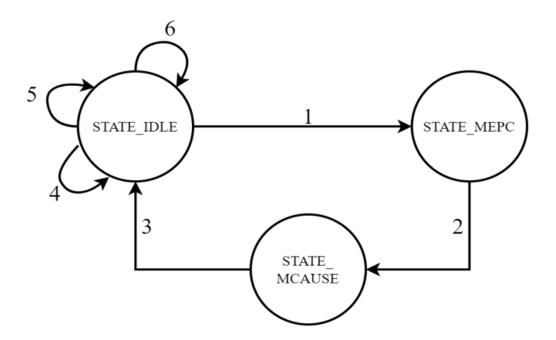
2.4 Trap Handling Process & Implement on Pipelined CPU

When a trap occurs, we need to assure the following principles or steps to keep correct running status:

- 1. Finish all instructions before the trap correctly, and flush the instruction which caused that trap and its following instructions.
- 2. Store the current running status to CSRs, and jump to the address that **mtvec** defined.
- 3. After a sret instruction, we remodify the CSRs, and jump to the address that **mepc** defined to recover running status.

In this experiment, we are to find the interrupt instructions at **WB** stage, illegal instructions at **ID** stage, and memory fault at **MEM** stage. That means we need to flush at most 4 instructions. As guidebook suggested, we enter trap at **WB** stage. However, this may cause some problems, and we will discuss it later.

Each trap handling will cost 3 cycles to finish all writings to CSRs, so we introduce an automaton:



The transactions of the automaton will be discussed in more detail afterwards.

3 Steps and data records

3.1 Completed Verilog source files

3.1.1 ExceptionUnit.v

In the predefined parts there contains CSR registers and some inputs and outputs of this unit. As we are to finish other parts, we omit corresponding code.

We first define some constants for use. These constants stand for states of the automaton and addresses of CSRs.

```
parameter STATE_IDLE = 2'boo;

parameter STATE_MEPC = 2'bo1;

parameter STATE_MCAUSE = 2'b10;

parameter MSTATUS = 12'h300;

parameter MIE = 12'h304;

parameter MTVEC = 12'h305;

parameter MEPC = 12'h341;

parameter MCAUSE = 12'h342;

parameter MTVAL = 12'h343;

parameter MIP = 12'h344;
```

And then we define one wire to represent trapping in signal, and some registers for storing some useful variables during cycles. After them there is a state register to identify current state of the automaton.

```
wire trap_in = interrupt | illegal_inst | l_access_fault | s_access_fault | ecall_m;
```

```
reg[31:0] epc_cur_IDLE;
reg[31:0] epc_next_IDLE;
reg interrupt_IDLE;
reg illegal inst IDLE;
```

```
reg l_access_fault_IDLE;

reg s_access_fault_IDLE;

reg ecall_m_IDLE;

reg trap_in_IDLE;

reg interrupt_MEPC;

reg illegal_inst_MEPC;

reg l_access_fault_MEPC;

reg s_access_fault_MEPC;

reg ecall_m_MEPC;
```

reg[1:0] *state*;

We would store values to use across cycles; And then we also consider state transfers of the automaton, because they should both be implemented under sequential logic.

If reset signal is invalid, then we use the following codes to proceed; or we reset all these values to 0 (also represent **STATE_IDLE**). The codes are all in an **always** block triggered by **posedge clk**.

The automaton works as follows:

If the state is **STATE_IDLE**, then it keeps, until **trap_in** or **mret** signal is set. If set, the state transfers to **STATE_MEPC**.

If the state is **STATE_MEPC**, then it must transfer to **STATE_MCAUSE**.

And, if the state is **STATE_MCAUSE**, then it must transfer to **STATE_IDLE**.

```
always@(posedge clk) begin

if(rst) begin

epc_cur_IDLE <= 0;

epc_next_IDLE <= 0;

interrupt_IDLE <= 0;

illegal_inst_IDLE <= 0;

l access fault IDLE <= 0;</pre>
```

```
s_access_fault_IDLE <= 0;
  ecall_m_IDLE <= o;
  trap_in_IDLE <= o;
  interrupt_MEPC <= 0;</pre>
  illegal_inst_MEPC <= 0;</pre>
  l_access_fault_MEPC <= 0;
  s_access_fault_MEPC <= 0;
  ecall_mMEPC \le o;
  state <= 0;
  end
else begin
  epc_cur_IDLE <= epc_cur;
  epc_next_IDLE <= epc_next;
  interrupt_IDLE <= interrupt;</pre>
  illegal_inst_IDLE <= illegal_inst;</pre>
  l_access_fault_IDLE <= l_access_fault;
  s_access_fault_IDLE <= s_access_fault;
  ecall_m_IDLE <= ecall_m;
  trap_in_IDLE <= trap_in;
  interrupt_MEPC <= interrupt_IDLE;</pre>
  illegal_inst_MEPC <= illegal_inst_IDLE;
  l_access_fault_MEPC <= l_access_fault_IDLE;
  s_access_fault_MEPC <= s_access_fault_IDLE;</pre>
  ecall_m_MEPC <= ecall_m_IDLE;
  case(state)
  STATE IDLE: begin
```

```
if(trap_in | mret) state <= STATE_MEPC;
else state <= STATE_IDLE;
end

STATE_MEPC: begin
    state <= STATE_MCAUSE;
end

STATE_MCAUSE: begin
    state <= STATE_IDLE;
end
endcase
end</pre>
```

Next are the operations we need to do at each state of the automaton.

If reset signal is valid, we set CSR operations invalid, and do nothing else. If invalid, we operate as the following.

If the state is **STATE_IDLE**, we will consider several circumstances:

If **trap_in** signal is valid, we store MIE to MPIE and set MIE to 0 in **mstatus** register to disable Machine level interrupt.

Or if **mret** is valid, we restore the **mstatus** modified by **trap_in**, and read **mepc** to recover running status.

Or if **csr_rw_in** is valid, we do the CSR operations informed.

Or if all are invalid, we keep CSR operations invalid, and do nothing else.

If the state is **STATE_MEPC**, we set **mepc** to next instruction address (which we stored before) to be run in the original program, and read **mtvec** to jump to the trap handling program.

If the state is **STATE_MCAUSE**, we set **meause** according to the trap reason we stored before.

```
always@* begin

if(rst) begin

csr \ w \le 0;
```

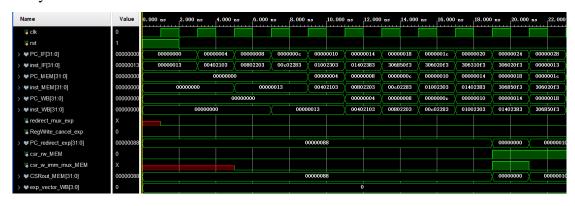
```
csr\_raddr <= o;
end
else begin
  case(state)
  STATE_IDLE: begin
    if(trap_in) begin
      csr\_w = 1'b1;
      csr wsc = 2'bo1;
      csr\_waddr = MSTATUS;
      csr\_wdata = \{mstatus[31:8], mstatus[3], mstatus[6:4], 1'b0, mstatus[2:0]\};
    end
    else if(mret) begin
      csr\_w = 1'b1;
      csr\_wsc = 2'bo1;
      csr\_waddr = MSTATUS;
      csr\ raddr = MEPC;
      csr_wdata = {mstatus[31:8], 1'b1, mstatus[6:4], mstatus[7], mstatus[2:0]};
    end
    else if(csr_rw_in) begin
      csr\_w = 1'b1;
      csr\_wsc = csr\_wsc\_mode\_in;
      csr\ raddr = csr\ rw\ addr\ in;
      csr_waddr = csr_rw_addr_in;
      csr_wdata = csr_w_imm_mux ? {{27{0}}}, csr_w_data_imm} :
      csr_w_data_reg;
    end
    else begin
      csr_w = 1'bo;
    end
  end
```

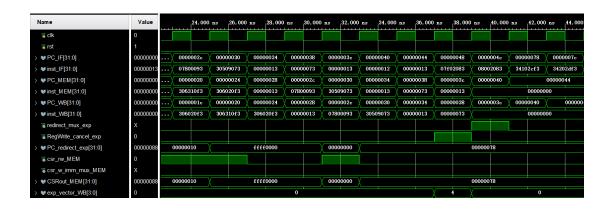
```
STATE MEPC: begin
     csr\_w = 1'b1;
     csr\_wsc = 2'bo1;
     csr\_waddr = MEPC;
     csr\_raddr = MTVEC;
     csr_wdata = interrupt_IDLE ? epc_next_IDLE : epc_cur_IDLE;
    end
   STATE_MCAUSE: begin
     csr_w = 1'b1;
     csr\_wsc = 2'bo1;
     csr\_waddr = MCAUSE;
     csr_wdata = {32{interrupt_MEPC}} & 32'h8000 |
     {32{l_access_fault_MEPC}} & 32'd5 |
     {32{s_access_fault_MEPC}} & 32'd7 |
     {32{ecall_m_MEPC}} & 32'd11;
   end
   endcase
 end
end
   Finally, we set some control signals according to the current state. Mainly we
need to flush all wrong instructions and redirect PC.
assign PC_redirect = csr_r_data_out;
assign redirect_mux = trap_in_IDLE | mret;
assign reg_MW_flush = trap_in | mret;
assign reg_EM_flush = trap_in | mret;
assign reg_DE_flush = trap_in | mret;
assign reg_FD_flush = trap_in | trap_in_IDLE | mret;
```

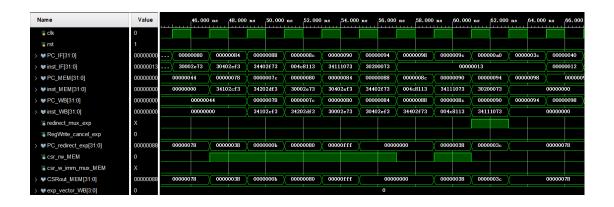
assign RegWrite_cancel = trap_in & ~interrupt;

3.2Implementation results

Here we record all our behavioral simulation results. Most will not be used for analysis.







Name	Value		68.000	ns 70	0.000 ns	72.0	00 ns	74.000	ns	76.000	ns 7	8.000	ns 80.0	00 ns	82.000	ns	84.000	ns 8	B6.000	ns 88.	00
¼ clk	0																				
₩ rst	1																				
> W PC_IF[31:0]	00000000) 00	000044	000000	48 00	00004c	0000	0050	0000	0054	00000	078	0000007c	000	08000	0000	0084	00000	0088	8000000	c
> w inst_IF[31:0]	00000013	00	000013	07£020	83 08	002083	0000	0013	0810	2023	34102	cf3	34202df3	300	02e73	3040	2ef3	34402	2£73	004c811	13
> ₩ PC_MEM[31:0]	00000000	0000	009с	000000	3c / 00	000040	0000	0044	0000	0048			0000004c			0000	0078	00000	007c	8000000	80
> W inst_MEM[31:0]	00000000	0000	0000	000000	13 00	000012	0000	0013				00000	1000			3410	2cf3	34202	2df3	30002e7	73
> W PC_WB[31:0]	00000000	X	0000	009с) 00i	00003с	0000	0040	0000	0044	00000	048		000	0004c			00000	0078	0000007	C
> W inst_WB[31:0]	00000000		000000	00) 00i	000013	0000	0012					00000000					34102	2cf3	34202df	E3
le redirect_mux_exp	X																				
18 RegWrite_cancel_exp	0																				
> W PC_redirect_exp[31:0]	00000088		00000078									0000	0040	00000	0002	8000000	80				
18 csr_rw_MEM	0																				
1& csr_w_imm_mux_MEM	X																				
> W CSRout_MEM[31:0]	00000088							000000	78							0000	0040	00000	0002	8000000	30
> we exp_vector_WB[3:0]	0			0			χ :	В							0						

Name	Value	90.000	ns 92.000	ns 94.000	ns 96.000	ns 98.000	ns 100.00	00 вз 102.00	00 ns 104.00	00 ns 106.0	00 ns 108.0	00 ns 110.0
1∉ clk	0											
₩ rst	1											
> W PC_IF[31:0]	00000000	00000090	00000094	00000098	0000009с	000000a0	00000044	00000048	0000004c	00000050	00000054	00000058
> W inst_IF[31:0]	00000013	34111073	30200073	(0000	0013		07f02083	08002083	00000013	08102023	000
> ₩ PC_MEM[31:0]	00000000	00000084	00000088	0000008c	00000090	00000094	00000098	0000	0009с	00000044	00000048	0000004c
> W inst_MEM[31:0]	00000000	30402ef3	34402f73	004c8113	34111073	30200073	X	00000000		00000013	07f02083	08002083
> W PC_WB[31:0]	00000000	00000080	00000084	00000088	0000008c	00000090	00000094	00000098	0000	0009c	00000044	00000048
> W inst_WB[31:0]	00000000	30002e73	30402ef3	34402f73	004c8113	34111073	χ	0000	00000		00000013	07f02083
16 redirect_mux_exp	X											
RegWrite_cancel_exp	0											
> W PC_redirect_exp[31:0]	00000088	00000fff	0000	0000	00000040	00000044	Χ			00000078		
16 csr_rw_MEM	0											
¼ csr_w_imm_mux_MEM	Х											
> W CSRout_MEM[31:0]	00000088	00000fff	0000	0000	00000040	00000044	χ			00000078		
> W exp_vector_WB[3:0]	0						0					

Name	Value	112.	000 ns 114.	000 ns 116.00	00 ns 118.00	00 ns 120.00	00 ns 122.0	00 ns 124.00	00 ns 126.00	00 ns 128.00	00 ns 130.00	0 ns 132.0				
₩ clk	0															
₩ rst	1															
₩ PC_IF[31:0]	00000000	0000005	00000060	00000078	0000007с	00000080	00000084	00000088	0000008c	00000090	00000094	00000098				
	00000013	00	000013	34102ef3	34202df3	30002e73	30402ef3	34402f73	004c8113	34111073	30200073	0000				
▶ PC_MEM[31:0]	00000000	00000050	00000054	χ '	00000058		00000078	0000007с	00000080	00000084	00000088	0000008c				
■ inst_MEM[31:0]	00000000	00000013	· X	0000	00000		34102cf3	34202df3	30002e73	30402ef3	34402£73	004c8113				
▶ ₩ PC_WB[31:0]	00000000	0000004c	00000050	00000054	χ	00000058		00000078	0000007с	00000080	00000084	00000088				
₩ inst_WB[31:0]	00000000	08002083	· X		00000000		30402ef3 34402f73 004c8 00000078 0000007c 00000 34102ef3 34202df3 30002 00000078 00000 34102ef3 34202			30002e73	30402ef3	34402£73				
18 redirect_mux_exp	x															
18 RegWrite_cancel_exp	0															
■ PC_redirect_exp[31:0]	00000088			00000078			0000004c	00000005	00000080	00000fff	0000	0000				
18 csr_rw_MEM	0															
16 csr_w_imm_mux_MEM	X															
■ CSRout_MEM[31:0]	00000088			00000078			0000004c	00000005	00000080	00000fff	0000	2f73 004c8113 0084 00000088				
→ exp_vector_WB[3:0]	0	2						0								

Name	Value	134.00	0 ns 136.00	0 ns 138.00	00 ns 140.00	00 ns 142.0	00 ns 144.0	00 ns 146.00	00 ns 148.	000 ns 150.0	00 ns 152.00	00 ns 154.0
¼ clk	0											
¼ rst	1											
> W PC_IF[31:0]	00000000	0000009с	000000a0	00000050	00000054	00000058	0000005c	00000060	00000064	00000068	00000078	0000007с
> W inst_IF[31:0]	00000013		00000013		08102023			00000013			34102ef3	34202df3
> ₩ PC_MEM[31:0]	00000000	00000090	00000094	00000098	0000	0009c	00000050	00000054	00000058	0000005c	χ	00000060
> • inst_MEM[31:0]	00000000	34111073	30200073	<u> </u>	00000000		00000013	08102023	00000013		0000	0000
> W PC_WB[31:0]	00000000	0000008c	00000090	00000094	00000098	000	0009c	00000050	00000054	00000058	0000005с	000
> Winst_WB[31:0]	00000000	004c8113	34111073		0000	00000		00000013	08102023	χ	0000	0000
18 redirect_mux_exp	x											
1 RegWrite_cancel_exp	0											
> W PC_redirect_exp[31:0]	00000088	0000004c	00000050	(<u> </u>				0000	0078			
18 csr_rw_MEM	0											
16 csr_w_imm_mux_MEM	х											
> ₩ CSRout_MEM[31:0]	00000088	0000004c	00000050	(0000	0078			
> ₩ exp vector WB[3:0]	0				0				y i	Υ		, '

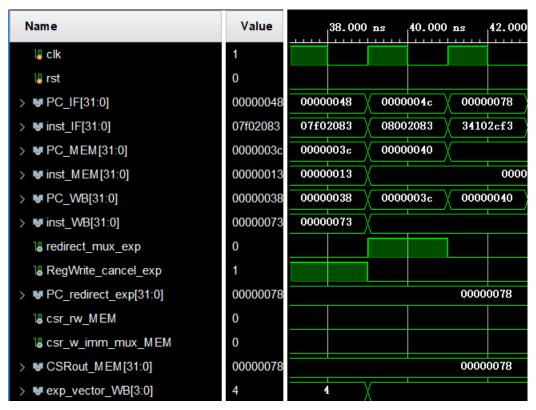
Name	Value	156.00	00 ns 158.00	00 ns 160.00	00 ns 162.00	0 ns 164.00	0 ns 166.00	00 ns 168.00	00 ns 170.00	00 ns 172.00	00 ns 174.0	00 ns 176.0
¼ clk	0											
₩ rst	1											
> W PC_IF[31:0]	00000000	00000080	00000084	00000088	0000008c	00000090	00000094	00000098	0000009с	000000a0	00000058	0000005c
> w inst_IF[31:0]	00000013	30002e73	30402ef3	34402f73	004c8113	34111073	30200073	χ		000	00013	
> ₩ PC_MEM[31:0]	00000000	00000060	00000078	0000007с	00000080	00000084	00000088	0000008c	00000090	00000094	00000098	000
> ₩ inst_MEM[31:0]	00000000	00000000	34102cf3	34202df3	30002e73	30402ef3	34402f73	004c8113	34111073	30200073	χ	00000000
> ₩ PC_WB[31:0]	00000000	0000	00060	00000078	0000007с	00000080	00000084	00000088	0000008c	00000090	00000094	00000098
> ₩ inst_WB[31:0]	00000000	0000	00000	34102cf3	34202df3	30002e73	30402ef3	34402£73	004c8113	34111073	χ	00000000
18 redirect_mux_exp	X											
18 RegWrite_cancel_exp	0											
> ₩ PC_redirect_exp[31:0]	00000088	00000078	00000054	00000007	00000080	00000fff	0000	00000	00000054	00000058	χ	00000078
16 csr_rw_MEM	0											
16 csr_w_imm_mux_MEM	X											
> W CSRout_MEM[31:0]	00000088	00000078	00000054	00000007	00000080	00000fff	0000	00000	00000054	00000058		00000078
> W exp_vector_WB[3:0]	0							0				

Name	Value	1	78.000 ns	180.00	00 ns	182.00	0 ns	184.00	0 ns	186.00	0 ns	188.00	0 ns	190.00	0 ns 1	92.00	0 ns 1	94.000	ns 1	196.00
₩ clk	0																			
₩ rst	1																			
> W PC_IF[31:0]	00000000	. 00000	060 00	00064	0000	0068	00000	006c	0000	0070	0000	0074	0000	0078	00000	000	00000	004	00000	800
> • inst_IF[31:0]	00000013				000000	013					0000	0067	3410	2cf3	00000	013	00402	103	00802	203
> W PC_MEM[31:0]	00000000	0000009	9c / 00	000058	0000	005с	00000	0060	0000	00064	0000	0068	0000	006c	00000	070		00000	0074	
> • inst_MEM[31:0]	00000000	0000000)0)						0000	0013							00000	067	00000	000
> W PC_WB[31:0]	00000000	- χ	0000009с		0000	0058	00000	005c	0000	00060	0000	0064	0000	8900	00000	06c	00000	070		0000
> W inst_WB[31:0]	00000000		00000000		X						0000	00064 \ 00000068 \ 0000006 \ 0000007 \ \ 00013 \ \ 0000								067
18 redirect_mux_exp	X																			
1 RegWrite_cancel_exp	0																			
> W PC_redirect_exp[31:0]	00000088											000000	78							
16 csr_rw_MEM	0																			
18 csr_w_imm_mux_MEM	X																			
> W CSRout_MEM[31:0]	00000088		00000078																	
> W exp_vector_WB[3:0]	0											0								

4 Analysis of the results

Here we only consider some typical results.

4.1 Trap by ecall



We will see the simulation in 37-43ns. During 37-39ms, the instructions in the pipeline is:

IF 0x07f02083, **lw** x1, 127(x0)

MEM 0x00000013, addi x0, x0, 0

WB 0x00000073, **ecall**

As We detected trap at WB stage, **RegWrite_cancel** was immediately set to invalid, and flush all other instructions (**STATE_IDLE**). Next cycle we see **redirect_mux_exp** set to valid, in order to begin the trap handling program, and flush all other instructions as well (**STATE_MEPC**). Finally at 41-43ns, we read out the first instruction of the trap handling program (**STATE_MCAUSE**). That first instruction is:

IF 0x34102cf3, **csrr x25**, **0x341**

This process run correctly as we could see.

```
Zhejiang University Computer Organization Experimental
                             SOC Test Environment (With RISC-U)
                                                                             ×03: gp 00000000
   x0:zero 00000000
                           x01: ra 00000078
                                                   x02: sp 00000008
  x04: tp 00000010
                          x05: t0 00000014
                                                                             x07: t2 0FFF0000
                                                   x06: t1 FFFF0000
                                                                             x11: a1 00000000
  x8:fps0 00000000
                          x09: s1 00000000
                                                   ×10: a0 00000000
  ×12: a2 00000000
×16: a6 00000000
                          x13: a3 00000000
x17: a7 00000000
                                                                             x15: a5 000000000
                                                   ×14: a4 00000000
                                                                             x19: s3 00000000
                          x17: a7
                                                   x18: s2 00000000
  x20: s4 00000000
                          x21: s5 000000000
                                                   x22: s6 000000000
                                                                             x23: s7 00000000
  x24: s8 00000000
                          x25: s9 00000054
                                                   x26:s10 000000000
                                                                             x27:s11 00000007
  x28: t3 00000080
                          x29: t4 0000057E
                                                   x30: t5 000000000
                                                                             x31: t6 000000000
                                   00000013
00000073
                                                   rs1Addr 00000000
Exp-Sig 80000000
                                                   I/ABSel 00000000
CPUAddr 00000000
                         ALU-Out 00000000
WB-Data 00000078
                           B-Addr 00000001
                                                   CPU-DAO 00000014
CODE-00 00000000
                         mop JStall:addi0
                                                                             CODE-03 00000000
CODE-04 00000000
                              call
                                                                             CODE-07 00000000
CODE-B8 808080808
CODE-BC 808880808
CODE-10 808880808
CODE-14 808080808
CODE-18 808080808
                                     nop JStall:addi0
                                                                             CODE-0B 00000000
                                           csrrw x01,x05,0300
                                                                             CODE-0F 00000000
                                                  addi x01,x00,078
                                                                             CODE-13 00000000
                        CODE-15 888888888
CODE-19 88888888
CODE-1D 88888888
CODE-21 88888888
                                                   CODE-16 00000000
                                                                             CODE-17 00000000
                                                                             CODE-1B 00000000
CODE-1F 00000000
CODE-23 00000000
                                                   CODE-1A 00000000
CODE-1C 00000000
                                                  CODE-1E 00000000
CODE-29 999999999
CODE-24 99999999
                                                  CODE-22 00000000
                        CODE-25 00000000
                                                  CODE-26 00000000
                                                                             CODE-27 00000000
```

The ecall at MEM stage.

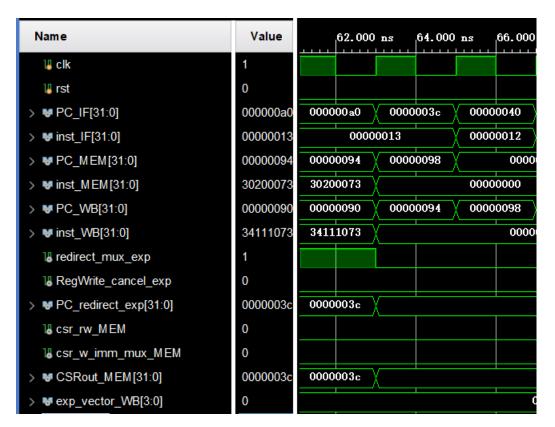
```
Zhejiang University Computer Organization Experimental SOC Test Environment (With RISC-U)
                                                                                    x03: gp 00000000
  ×0:zero 00000000
                             ×01: ra FFFF0000
                                                        x02: sp 00000008
                                                                                    x07: t2 0FFF0000
                             x85: t0 00000014
                                                        ×06: t1 FFFF0000
  x04: tp 00000010
                                                                                               00000000
                                                        ×10: a0 000000000
                                                                                    x11: a1
  ×8:fps0 00000000
×12: a2 00000000
                            ×89: s1 00000000
                            ×13: a3 00000000
                                                                                    ×15: a5 00000000
                                                                  00000000
                                                        ×14: a4
  ×16: a6 000000000
                            x17: a7 00000000
                                                        x18: s2 00000000
                                                                                    x19: s3 00000000
                                                                                    x23: s7 000000000
  ×20: s4 00000000
                            x21: s5 000000000
                                                        x22: s6 00000000
       s8 00000000
t3 00000080
                            x25: s9 00000054
                                                        x26:s10 000000000
                                                                                    x27:s11 000000007
                            x29: t4 0000057E
                                                        x30: t5 00000000
                                                                                    x31: t6 000000000
                            INST-D 0000001
INST-D 30509073
INST-W 07800093
INST-W 00000013
ALU-Out 00000000
WB-Data 00000000
                                                        Exp-Sig 00000000
B/PCE-S 00000100
I/ABSel 00010001
 CODE-00 00000000
                                                                                    CODE-03 00000000
 CODE-04 00000000
                                 nop JStall:addi0
CODE-94 8888888
CODE-96 88888888
CODE-10 88888888
CODE-14 88888888
CODE-18 88888888
CODE-16 88888888
CODE-1C 88888888
CODE-28 88888888
CODE-24 88888888
                                                                                    CODE-07 00000000
                                        csrrw x01,x05,0300
                                                                                    CODE-0B 00000000
                                               addi x01,x00,078
                                                                                     CODE-OF
                                                                                                00000000
                                                                                    CODE-13 888888888
CODE-17 88888888
CODE-18 88888888
CODE-1F 88888888
                                                       nop JStall:addin
                           CODE-15 00000000
                                                        CODE-16 00000000
                           CODE-19 00000000
                                                       CODE-1A 00000000
                           CODE-1D 00000000
                                                       CODE-1E 00000000
                           CODE-21 00000000
                                                       CODE-22 00000000
                                                                                     CODE-23 00000000
                           CODE-25 00000000
                                                       CODE-26 00000000
                                                                                     CODE-27 00000000
```

The ecall at WB stage (**STATE_IDLE**).

```
Zhejiang University Computer Organization Experimental
                          SOC Test Environment (With RISC-V)
 ×0:zero 00000000
                        x01: ra 00000078
                                              x02: sp 00000044
                                                                      ×03: gp 00000000
 x04: tp 00000010
                       x05: t0 00000014
                                              x06: t1 FFFF0000
                                                                     x07: t2 OFFF0000
 x8:fps0 00000000
                       x09: s1 00000000
                                              ×10: a0 00000000
                                                                     ×11: a1 00000000
×12: a2 00000000
                       ×13: a3 00000000
                                              ×14: a4 00000000
                                                                     ×15: a5 00000000
×16: a6 000000000
                       ×17: a7 00000000
                                              ×18: s2 00000000
                                                                     x19: s3 000000000
x20: s4 00000000
                       x21: s5 00000000
                                              x22: s6 000000000
                                                                     x23: s7 000000000
x24: s8 00000000
                       x25: s9 00000040
                                              x26:s10 000000000
                                                                     x27:s11 00000002
x28: t3 00000080
                       x29: t4 0000057E
                                              x30: t5 000000000
                                                                     x31: t6 00000000
                       INST-IF 302310F3
INST-ID 302CADF3
                                              rs1Data 000000040
                                              rs1Addr 00000019
                                             Exp-Sig 999991F9
B/PCE-S 99999199
1/ABSel 99999999
CPU-DAi FFFFFBF
CPU-DAO 9999993C
                       INST-# 30200073
INST-WB 34111073
ALU-Out 00000000
WB-Data 00000040
CODE-00 00000000
                       csrrw x06,x02,0B
                                                                     CODE-03 00000000
CODE-04 00000000
                           csrrs x19,x02,0Bia
                                                                     CODE-07 00000000
CODE-08 00000000
                                 srrwi x10,x03,085A
                                                                     CODE-0B 00000000
CODE-0C 00000000
                                       illegal instruction
                                                                     CODE-0F 00000000
CODE-10 00000000
                                            csrrw x02,x01,03400 CODE-13 00000000
CODE-14 00000000
                      CODE-15 00000000
                                             CODE-16 00000000
                                                                    CODE-17 00000000
CODE-1B 00000000
CODE-18 00000000
                      CODE-19 00000000
                                             CODE-1A 00000000
CODE-1E 00000000
CODE-1C 00000000
                      CODE-1D 00000000
CODE-20 00000000
                                                                    CODE-1F 00000000
                      CODE-21 00000000
                                             CODE-22 00000000
                                                                    CODE-23 00000000
CODE-24 00000000
                      CODE-25 00000000
                                             CODE-26 00000000
                                                                    CODE-27 00000000
```

Trap handling at running, using CSR operations.

4.2Returning from Trap



We will see the simulation in 61-67ns. That trap handling was caused by ecall at address 0x38. During 61-63ms, the instructions in the pipeline is:

IF 0x00000013, addi x0, x0, 0

MEM 0x30200073, mret

WB 0x34111073, **csrw** 0x341, x2

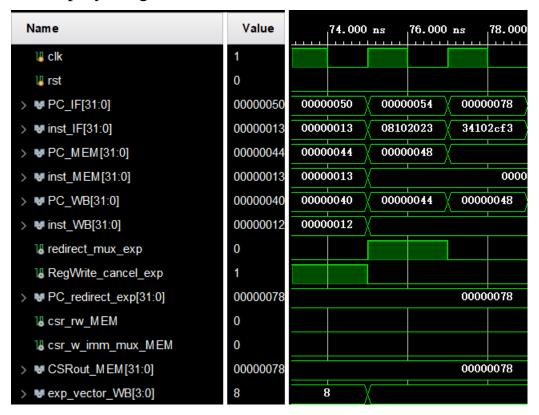
As We detected a **mret** instruction at MEM stage, we immediately set **redirect_mux_exp** to valid, and flush other instructions (**STATE_IDLE**). We read out mepc and jumped back to the original program (**STATE_MEPC**). After that we do nothing significantly intervening the running conditions of our program (**STATE_MCAUSE**).

As we could see, the address we return to is 0x3c, which was not so correct in theory because we add 4 to **mepc** in our trap handling program. And this will be discussed later.

```
Zhejiang University Computer Organization Experimental
                         SOC Test Environment (With RISC-U)
  x0:zero 00000000
                                                                   x03: gp 00000000
                       x01: ra 00000078
                                            x02: sp 00000008
  x04: tp 00000010
                       x05: t0 00000014
                                                                   x07: t2 0FFF0000
                                            x06: t1
                                                     FFFFAAAA
  x8:fps0 000000000
                       x09: s1 00000000
                                            ×10: a0 00000000
                                                                            00000000
                                                                  ×11: a1
  ×12: a2 00000000
×16: a6 00000000
                       x13: a3 00000000
                                            ×14: a4 000000000
                                                                   ×15: a5 00000000
                       x17: a7 00000000
                                            x18: s2 00000000
                                                                  x19: s3 000000000
  x20: s4 00000000
                       x21: s5 000000000
                                            x22: s6 00000000
                                                                   x23: s7 00000000
  x24: s8 00000000
                       x25: s9 000000054
                                            x26:s10 000000000
                                                                  x27:s11 000000007
  ×28: t3 00000080
                      x29: t4 0000057E
                                            ×30: t5 00000000
                                                                  x31: t6 00000000
                      ALU-Out 00000000
WB-Data 00000078
CODE-00 00000000
                      addi x00,x00,000
                                                                   CODE-03 00000000
CODE-04
                          nop JStall:addi0
                                                                   CODE-07 00000000
CODE-08 00000000
CODE-0C 00000000
CODE-10 00000000
                                                                   CODE-0B 00000000
                                     nop JStall:addi0
                                                                   CODE-0F 00000000
                                           CODE-16 00000000 CODE-17 000000000
CODE-14 00000000
                     CODE-15 00000000
                     CODE-19 00000000
CODE-1D 00000000
                                                                   CODE-17 00000000
CODE-18 00000000
                                            CODE-1A 00000000
                                                                   CODE-1B 00000000
CODE-1C 00000000
                                            CODE-1E 00000000
CODE-28 00000000
CODE-24 00000000
                                                                   CODE-1F 00000000
                     CODE-21 00000000
                                            CODE-22 00000000
                                                                   CODE-23 00000000
                     CODE-25 00000000
                                            CODE-26 00000000
                                                                   CODE-27 00000000
```

The program control returned.

4.3 Trap by Illegal Instruction



We see the simulation in 73-79ns. However, as the process is very similar to trap by ecall, we won't explain it so much.

We can see the following instruction caused the trap:

WB 0x00000012, illegal instruction

We start automaton transfer at this time, and just as same as how we deal with trap by ecall, we do the same operations in following cycles.

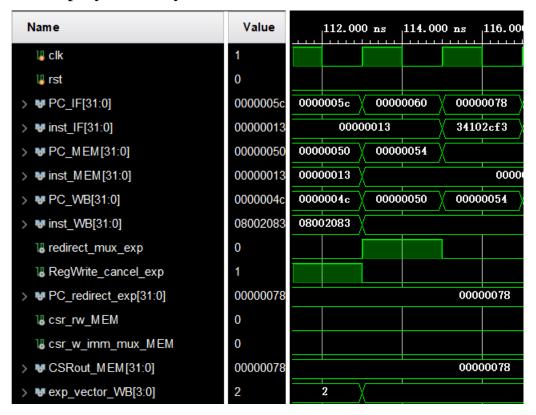
When returned, the return address is 0x44, which was a correct result in theory.

Here are some FPGA running results:

```
Zhejiang University Computer Organization Experimental
                             SOC Test Environment (With RISC-U)
x01: ra 00000078 x02: sp 00000008
                                                         x02: sp 00000008
                                                                                      x03: gp 00000000
   x0:zero 00000000
                             x05: t0 00000014
   ×04: tp 00000010
                                                         x06: t1 FFFF0000
                                                                                      x07: t2 0FFF0000
  x8:fps0 00000000
x12: a2 00000000
                             x09: s1 00000000
                                                         ×10: a0 00000000
                                                                                      x11: a1 00000000
                                                                                     x15: a5 000000000
x19: s3 00000000
                             ×13: a3 00000000
                                                         ×14: a4 00000000
                                                         x18: s2 000000000
  ×16: a6 00000000
                             x17: a7 00000000
  x20: s4 00000000
x24: s8 00000000
                             x21: s5 000000000
                                                         x22: s6 00000000
                                                                                     x23: s7 00000000
                             x25: s9 00000054
x29: t4 0000057E
                                                         x26:s10 00000000
                                                                                      x27:s11 000000007
  x28: t3 00000080
                                                         x30: t5 00000000
                                                                                     x31: t6 00000000
                                                         rs1Data 00000000
rs1Addr 00000000
                                                                                     rs2Data 00000000
rs2Addr 00000000
PCJumpA 00000048
D/C-Hzd 00000000
                            INST-ID 00000012
INST-EX 00000013
INST-# 00000073
INST-WB 00000013
                                                        Exp-Sig 99999999
B-PCE-S 99999999
I-ABSel 99999999
CPUAddr 99999999
                            WB-Data 00000000
WB-Addr 00000000
CODE-00 00000000
                            nop JStall:addi0
CODE-94 999999999
CODE-96 99999999
CODE-9C 99999999
CODE-19 99999999
                                                                                      CODE-03 00000000
                                  addi x00,x00,000
                                                                                      CODE-07 00000000
                                         nop JStall:addi0
                                                                                      CODE-0B 00000000
                                                call
                                                                                      CODE-0F 00000000
                                                        nop JStall:addi0
CODE-14 B0000000
CODE-18 B0000000
CODE-1C B0000000
CODE-20 B00000000
                                                                                      CODE-13 00000000
                            CODE-15 00000000
                                                        CODE-16 00000000
                                                                                     CODE-17 00000000
                            CODE-19 00000000
                                                        CODE-1A 00000000
                                                                                     CODE-1B 00000000
CODE-1F 00000000
CODE-23 00000000
                            CODE-1D 00000000
                                                        CODE-1E 00000000
                           CODE-21 00000000
                                                        CODE-22 00000000
CODE-24 00000000
                           CODE-25 00000000
                                                        CODE-26 00000000
                                                                                      CODE-27 00000000
```

The illegal instruction at MEM stage.

4.4Trap by Memory Fault



We see the simulation in 111-117ns. However, as the process is very similar to traps above, we won't explain it so much too.

We can see the following instruction caused the trap:

WB 0x08002083, **lw** x1, 128(x0)

That is a memory fault which will cause a exception. We start automaton transfer at this time, and just as same as how we deal with traps above, we do the same operations in following cycles.

When returned, the return address is 0x60, which was a correct result in theory.

Here are some FPGA running results:

```
Zhejiang University Computer Organization Experimental
                          SOC Test Environment (With RISC-V)
                                                                    x03: gp 00000000
                                             x02: sp 0000003C
  x0:zero 00000000
                        x01: ra 00000078
                                             x06: t1 FFFF0000
                                                                    x07: t2 0FFF0000
          00000010
                       x05: t0 00000014
  ×94: tp
                                             ×10: a0 00000000
                                                                    ×11: a1 00000000
                       x09: s1 000000000
  x8:fps0 000000000
          00000000
                                             ×14: a4 00000000
                                                                    ×15: a5 00000000
                       x13: a3 000000000
  x12: a2
 x16: a6 00000000
                                             x18: s2
                                                      00000000
                                                                    x19: s3 000000000
                       x17: a7 000000000
                                                                    x23: s7 000000000
                       x21: s5 00000000
                                              x22: s6 000000000
 x20: s4 00000000
      s8
          00000000
                       x25: s9 00000038
                                              x26:s10 000000000
                                                                     x27:s11 0000000B
 x28: t3 00000080
                       x29: t4 0000057E
                                             x30: t5 00000000
                                                                     x31: t6 00000000
                                             rs1Addr 90000908
Exp-Sig 900800F1
B/PCE-S 00000100
                                              I/ABSel 00010001
                       INST-WB 00000012
ALU-Out 90000000
                                             CPUAddr 00000000
CPU-Dai FFFFFFBF
                       WB-Addr 00000000
                      mop JStall:addi0
                                                                     CODE-03 00000000
CODE-84 80808080
CODE-88 80808080
                           lw x01,x00,080H
                                                                     CODE-07 00000000
                                 lw x01,x00,07FH
                                                                     CODE-0B 00000000
CODE-0C 00000000
                                      mop JStall:addi0
                                                                     CODE-0F 00000000
CODE-10 00000000
CODE-14 00000000
                                             addi x00,x00,000
                                                                      CODE-13 00000000
                      CODE-15 00000000
                                             CODE-16 00000000
                                                                     CODE-17 00000000
CODE-1B 00000000
CODE-18 00000000
                      CODE-19 00000000
CODE-1D 00000000
                                             CODE-1A 00000000
CODE-1C 00000000
                                             CODE-1E 00000000
                                                                      CODE-1F 00000000
CODE-20 00000000
                      CODE-21 00000000
                                             CODE-22 00000000
                                                                      CODE-23 00000000
CODE-24 00000000
                      CODE-25 00000000
                                              CODE-26 00000000
                                                                      CODE-27 00000000
```

The fault instruction is at MEM.

```
Zhejiang University Computer Organization Experimental
                         SOC Test Environment (With RISC-U)
  x0:zero 00000000
                       ×01: ra 00000078
                                                                    ×03: gp 00000000
                                             x02: sp 00000008
  x04: tp 00000010
                       x05: t0 00000014
                                             x06: t1 FFFF0000
                                                                    x07: t2 0FFF0000
  x8:fps0 000000000
                       x09: s1 00000000
                                             ×10: a0 00000000
                                                                    ×11: a1 000000000
 x12: a2 00000000
                       ×13: a3 00000000
                                             ×14: a4 00000000
                                                                    x15: a5 000000000
 x16: a6 00000000
                       ×17: a7 00000000
                                             x18: s2 00000000
                                                                    ×19: s3 000000000
 x20: s4 00000000
                       x21: s5 00000000
                                             x22: s6 00000000
                                                                    x23: s7 00000000
 x24: s8 00000000
                       x25: s9 000000054
                                             x26:s10 000000000
                                                                    x27:s11 00000007
 x28: t3 00000080
                       x29: t4 0000057E
                                             x30: t5 00000000
                                                                    x31: t6 00000000
                                                                   rs2Data 88888888
rs2Addr 88888888
PCJumph 88888844
D/C-Hzd 88888888
                                             rs1Data 00000000
rs1Addr 00000000
                                            CPU-DAI FFFFFFBF
CODE-00 00000000
                      lw x01,x00,07FH
                                                                    CODE-03 00000000
CODE-04 00000000
                           nop JStall:addi0
                                                                    CODE-07 00000000
CODE-0B 00000000
CODE-08 00000000
                                addi x00,x00,000
CODE-8C 88888888
                                      nop JStall:addi0
CODE-10 00000000
CODE-14 00000000
                                                                    CODE-0F 00000000
                                                                    CODE-13 00000000
                      CODE-15 00000000
                                            CODE-16 00000000
                                                                    CODE-17 00000000
CODE-18 00000000
                      CODE-19 00000000
                                            CODE-1A 00000000
CODE-1C 00000000
                                                                    CODE-1B 00000000
                     CODE-1D 00000000
CODE-21 00000000
                                            CODE-1E 00000000
CODE-20 00000000
                                                                    CODE-1F 00000000
                                            CODE-22 00000000
CODE-24 00000000
                                                                    CODE-23 00000000
                     CODE-25 00000000
                                            CODE-26 00000000
                                                                    CODE-27 00000000
```

Here lw x1, 128(x0) is at WB stage (STATE IDLE).

```
Zhejiang University Computer Organization Experimental
                                  SOC Test Environment (With RISC-V)
                               x01: ra 0080BF00
   x0:zero 00000000
                                                           x02: sp 0000005C
x06: t1 FFFF0000
                                                                                         x03: gp 00000000
  x04: tp 00000010
x8:fps0 00000000
                               x05: t0 00000014
                                                                                         x07: t2
                                                                                                     ØFFFØØØØ
                               x09: s1 00000000
                                                           ×10: a0 00000000
                                                                                        ×11: a1
                                                                                                    00000000
  x12: a2 00000000
                              x13: a3 00000000 
x17: a7 00000000
                                                           ×14: a4 00000000
                                                                                        ×15: a5 00000000
  ×16: a6 00000000
                                                           ×18: s2 00000000
                                                                                        x19: s3 00000000
  ×20: s4 00000000
×24: s8 00000000
                               x21: s5 00000000
                                                           x22: s6 00000000
                                                                                        x23: s7 000000000
                              x25: s9 00000058
                                                           x26:s10 000000000
                                                                                        x27:s11 00008000
  x28: t3 00000080
                              x29: t4 0000057E
                                                           x30: t5 00000000
                                                                                        x31: t6 00000000
                              INST-IF 00000013
INST-ID 00000013
INST-EX 00000013
INST-M 00000013
                             INST-WB 000000000
ALU-Out 00000000
WB-Data 00000058
CODE - 99 999999999
CODE - 94 999999999
CODE - 98 999999999
                              nop JStall:addi0
                                                                                        CODE-03 00000000
                                   nop JStall:addi0
                                                                                        CODE-07 00000000
CODE-9C 909090999
CODE-19 909090999
CODE-14 90909099
CODE-18 909090999
CODE-1C 909090999
CODE-24 909090999
                                          nop JStall:addi0
                                                                                        CODE-0B 00000000
                                                 nop JStall:addi0
                                                                                        CODE-0F 00000000
                                                         nop DStall: Iw 00
CODE-16 00000000
                                                                                       CODE-17 99999999
CODE-17 99999999
CODE-18 99999999
CODE-1F 99999999
CODE-23 99999999
                             CODE-15 00000000
                             CODE-19 00000000
                                                          CODE-1A 00000000
CODE-1E 00000000
                             CODE-1D 00000000
                            CODE-21 00000000
CODE-25 00000000
                                                          CODE-22 00000000
CODE-24 00000000
                                                          CODE-26 00000000
                                                                                        CODE-27 00000000
```

The program control is returned to User.

```
Zhejiang University Computer Organization Experimental
                            SOC Test Environment (With RISC-V) x01: ra 00000078 x02: sp 00000030
    x0:zero 00000000
                                                    x02: sp 0000003C
                                                                             ×03: gp 00000000
                            x05: t0 00000014
    x04: tp 00000010
                                                    x06: t1 FFFF0000
                                                                             x07: t2 0FFF0000
   x8:fps0 00000000
                            x09: s1 00000000
                                                    ×10: a0 000000000
                                                                             x11: a1 00000000
                                                    ×14: a4 00000000
   x12: a2 00000000
                           ×13: a3 00000000
                                                                             ×15: a5 00000000
   ×16: a6 00000000
                           ×17: a7 00000000
                                                    x18: s2 000000000
                                                                             ×19: s3 00000000
   x20: s4 000000000
                           x21: s5 00000000
                                                    x22: s6 00000000
                                                                             x23: s7 00000000
   ×24: s8 00000000
                           x25: s9 00000038
                                                    x26:s10 000000000
                                                                             x27:s11 0000000B
  x28: t3 00000080
                           x29: t4 0000057E
                                                    x30: t5 00000000
                                                                              x31: t6 00000000
 PC -- IF 88888854
PC -- ID 8888884C
PC -- EXE 8888884C
PC -- HDP 88888848
PC -- WB 88888844
                           INST-ID 00000013
INST-EX 00000000
INST-M 00000000
INST-WB 00000000
                                                    Exp-Sig 9F0001F0
B/PCE-S 80000100
1/ABSel 90010001
CPUAddr 90000000
                          WB-Data 00000000
WB-Addr 00000000
 CODE-00 00000000
                          sw x00,x01,080H
                                                                              CODE-03 00000000
 CODE-04 00000000
                               nop JStall:addi0
                                                                              CODE-07 00000000
 CODE-08 00000000
                                      nop DStall: lw 00
                                                                              CODE-0B 00000000
CODE-0C 00000000
                                            mop DStall: lw 00
                                                                              CODE-0F 00000000
CODE-10 00000000
                                                   nop DStall: lw 00
                                                                               CODE-13 00000000
CODE-14 00000000
CODE-18 00000000
CODE-1C 00000000
                          CODE-15 00000000
                                                    CODE-16 00000000
                                                                               CODE-17 000000000
                         CODE-19 00000000
                                                    CODE-1A 00000000
                                                                               CODE-1B 00000000
                         CODE-1D 00000000
                                                                              CODE-1F 00000000
CODE-23 00000000
CODE-27 00000000
                                                    CODE-1E 00000000
CODE-20 00000000
                         CODE-21 00000000
                                                    CODE-22 00000000
CODE-24 00000000
                         CODE-25 00000000
                                                    CODE-26 00000000
```

At 0x54, the program triggers a save memory fault (**STATE_IDLE**).

```
Zhejiang University Computer Organization Experimental
                         SOC Test Environment (With RISC-U)
                       ×01: ra 0080BF00
                                            x02: sp 0000005C
                                                                  ×03: gp 00000000
 x0:zero 00000000
                                                                 x07: t2
                                                                          ØFFFØØØØ
 x04: tp 00000010
                      x05: t0 00000014
                                            x06: t1 FFFF0000
                                                                          00000000
 x8:fps0 00000000
                                            ×10: a0
                                                    00000000
                      x09: s1 00000000
                                                                  ×15: a5 00000000
                      ×13: a3 00000000
                                                    00000000
 x12: a2 00000000
                                            ×14: a4
 x16: a6 00000000
                               00000000
                                            x18: s2 00000000
                                                                  ×19: s3 00000000
                      ×17: a7
                                                                  x23: s7
                      x21: s5 000000000
                                            x22: s6 00000000
 x20: s4 000000000
                                                                           00008000
     s8 000000000
                      x25: s9
                                            x26:s10
                                                    00000000
                                                                  x27:s11
                      x29: t4 0000057E
                                            x30: t5 00000000
                                                                  x31: t6 000000000
      t3 000000080
                      INST-ID 00000013
INST-EX 00000000
                      ALU-Out 00000000
                      WB-Data 00000058
WB-Addr 00000000
                      nop JStall:addi0
                                                                  CODE-03 00000000
CODE-00 00000000
CODE-04 00000000
                          nop JStall:addi0
                                                                  CODE-07
CODE-08 00000000
                                nop DStall: lw 00
                                                                  CODE-0B 00000000
CODE-0C 00000000
CODE-10 00000000
                                     mop DStall:lw 00
                                                                  CODE-OF
                                                                           00000000
                                           nop DStall: lw 00
                                                                  CODE-13
                                                                           00000000
                                            CODE-16 000000000
CODE-14 00000000
                      CODE-15 00000000
                                                                  CODE-17
                                                                  CODE-17 00000000
CODE-1B 00000000
CODE-18 00000000
                      CODE-19 00000000
                                            CODE-1A 00000000
CODE-1C 00000000
                      CODE-1D 00000000
                                            CODE-1E 00000000
                                                                   CODE-1F 00000000
CODE-20 00000000
                      CODE-21 00000000
                                            CODE-22 00000000
                                                                   CODE-23 00000000
CODE-24 00000000
                     CODE-25 00000000
                                            CODE-26 00000000
                                                                   CODE-27 00000000
```

The exception returned to 0x5c(STATE_MCAUSE).

4.5 Hardware Interrupt

As hardware interrupts have not much relation with software, there are no results we can show in behavioral simulation.

Hardware interrupts in our experiment are made by a switch on board. The dealing process are also similar to the traps we mentioned above. So, we will just show the FPGA results.

```
Zhejiang University Computer Organization Experimental
                             SOC Test Environment (With RISC-U)
                                                                          x03: gp 00000000
                                                  x02: sp 0000003C
                           x01: ra 00000078
x05: t0 00000014
   x0:zero 00000000
                                                                          x07: t2
                                                  x06: t1 FFFF0000
                                                                          ×11: a1 00000000
   x04: tp 00000010
                                                  ×10: a0 00000000
                          x09: s1 00000000
            00000000
   x8:fps0
                                                  ×14: a4 00000000
                                                                          ×15: a5 000000000
                          ×13: a3 00000000
×17: a7 00000000
   ×12: a2 00000000
                                                                          ×19: s3 00000000
                                                  ×18: s2 00000000
  ×16: a6 00000000
                                                                           x23: s7 00000000
                                                  x22: s6 00000000
                          x21: s5 000000000
  x20: s4 00000000
                                                   x26:s10 00000000
                                                                           x27:s11 0000000B
                          x25: s9 00000038
  x24: s8 00000000
                                                                           x31: t6 00000000
                                                   x30: t5 00000000
                          x29: t4 0000057E
  x28: t3 00000080
                                                  rs1Data 00000000
rs1Addr 00000000
Exp-Sig 0F000000
B/PCE-S 00000100
                                    34102CF3
                         INST-ID 88889888
INST-EX 88889888
INST-EX 88889888
INST-EX 88889888
ALU-Out 88889888
                                                   CPU-DAO 00000000
                          WB-Data 00000000
                                                                             CODE-03 00000000
                          csrrs x00,x01,0B<mark>58</mark>
CODE-00 00000000
                                                                             CODE-07 00000000
                               nop JStall:addi0
CODE-04 00000000
                                                                             CODE-0B 00000000
CODE-0F 00000000
CODE-13 00000000
CODE-17 00000000
                                     nop DStall:lw 00
CODE-08 00000000
CODE-0C 00000000
CODE-10 00000000
                                           nop DStall: lw 00
                                                   nop DStall: lw 00
                                                   CODE-16 00000000
CODE-14 00000000
                         CODE-15 00000000
                                                                              CODE-1B 00000000
                                                   CODE-1A 00000000
CODE-18 00000000
                         CODE-19 00000000
                                                                              CODE-1F 00000000
                                                   CODE-1E 00000000
CODE-1C 00000000
                         CODE-1D 00000000
                                                                              CODE-23 00000000
                                                   CODE-22 00000000
CODE-20 000000000
                         CODE-21 00000000
                                                                              CODE-27 00000000
                                                   CODE-26 00000000
CODE-24 00000000
                         CODE-25 00000000
```

Entering hardware interrupt at 0x40, several instructions flushed.

```
Zhejiang University Computer Organization Experimental
                          SOC Test Environment (With RISC-U)
                                               x02: sp 00000044
x06: t1 FFFF0000
                                                                        x03: gp 00000000
                        ×01: ra 00000078
x0:zero 00000000
                                                                        x07: t2 0FFF0000
x04: tp 00000010
x8:fps0 00000000
                       x05: t0 00000014
                                                                        ×11: a1 00000000
                       x09: s1 00000000
                                               ×10: a0 00000000
                                                                        ×15: a5 00000000
                                                ×14: a4 00000000
         00000000
                       ×13: a3 000000000
x12: a2
                                                                        x19: s3 00000000
                        ×17: a7 00000000
                                                ×18: s2 00000000
×16: a6 000000000
                                                x22: s6 00000000
                                                                        x23: s7 00000000
                        x21: s5 000000000
×20: s4 00000000
                                                                        x27:s11 00000002
                                                x26:s10 00000000
                        x25: s9 00000040
x24: s8 00000000
                                                                        x31: t6 000000000
                                                x30: t5 00000000
x28: t3 00000080
                        x29: t4 0000057E
                        INST-IF 00000013
INST-ID 00000013
INST-EX 00000000
                                                Exp-Sig 0F000000
B/PCE-S 00000100
                        INST-M 00000000
INST-WB 00000000
ALU-Out 00000000
                        WB-Addr 00000000
                                                                         CODE-03 00000000
                        nop JStall:addi0
CODE-00 00000000
                                                                        CODE-07 00000000
CODE-0B 00000000
CODE-0F 00000000
CODE-13 00000000
                             nop JStall:addi0
CODE-04 00000000
CODE-08 00000000
                                   nop DStall: lw 00
                                         nop DStall: lw 00
CODE-0C 00000000
                                                nop DStall: lw 00
CODE-10 00000000
                                                 CODE-16 00000000
                        CODE-15 00000000
                                                                         CODE-17 00000000
CODE-14 00000000
CODE-18 00000000
                                                                         CODE-1B 00000000
                        CODE-19 00000000
                                                 CODE-1A 00000000
                                                                         CODE-1F 00000000
                                                 CODE-1E 00000000
CODE-1C 00000000
                        CODE-1D 00000000
                                                 CODE-22 00000000
                                                                         CODE-23 00000000
                        CODE-21 00000000
CODE-20 00000000
                                                 CODE-26 00000000
                        CODE-25 00000000
                                                                         CODE-27 00000000
 CODE-24 00000000
```

Returing to 0x44.

5 Discussion and Conclusion

5.1 Problems

5.1.1 Problems Concerning Verilog Language

Problem 1. always@ Block

As what was generally discussed in the DingTalk group, always@(posedge clk), always@(negedge clk) and always@* would possibly operate very differently when running on FPGA board.

That's generally because at the positive edge of clock signal the values of many signals are not stable. To solve this problem, logic in **always**@* blocks should be as simple as possible, and the coder should avoid using <= and = operators in the same block.

Also, we could consider using negative edges as trigger to obtain stable signals. I put the <= operators in always@(posedge clk) block and = operators in always@* block and then solved the problem.

5.1.2 Problems Concerning the Experiment Guides

Problem 1. Detection at WB Stage

Trap detection at WB stage will surely cause some problems. Consider the following codes:

MEM sd x1, 0(x0)

WB ecall

When **ecall** is at WB stage, the MEM stage instruction has been dealt, so the memory could be wrongly set before entering the trap. The experiment guide didn't solve the problem, and I also can hardly think of a method that could solve this on 5-stage pipelined CPU. Maybe we could try to detect the trap at EXE and insert some **bubble** instructions?

Problem 2. Returning Address

In our experiment, the trap handling process will add 4 to **mepc**. That is to avoid we rerun the instruction which will cause an exception. However, software interrupts

are also affected, so if we enter the trap handling process with **mepc** set to next instruction of an **ecall**, the program will return to a wrong address.

We have to overcome the problem by omitting the difference of a software interrupt from an exception.

5.2 Achievements and conclusion

In this experiment, I implemented the trap handling process of a CPU, and learned the basic knowledge about RISC-V CSRs and control levels. The experiment is successful as a result.