浙江大学

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　　11 月　　23 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：Lab4：Pipelined CPU with Cache	
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：2022 年 11 月 23 日

1 Tasks and requirements

1.1 Tasks

The main tasks of Lab-4 are:

1. Get knowledge of the principles of cache controller.
2. Complete the code of cache controller.

1.2 Requirements

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:

cmu.v, the control unit of a 2-way set associative LRU replaced write-back cache which we finished in lab-3.

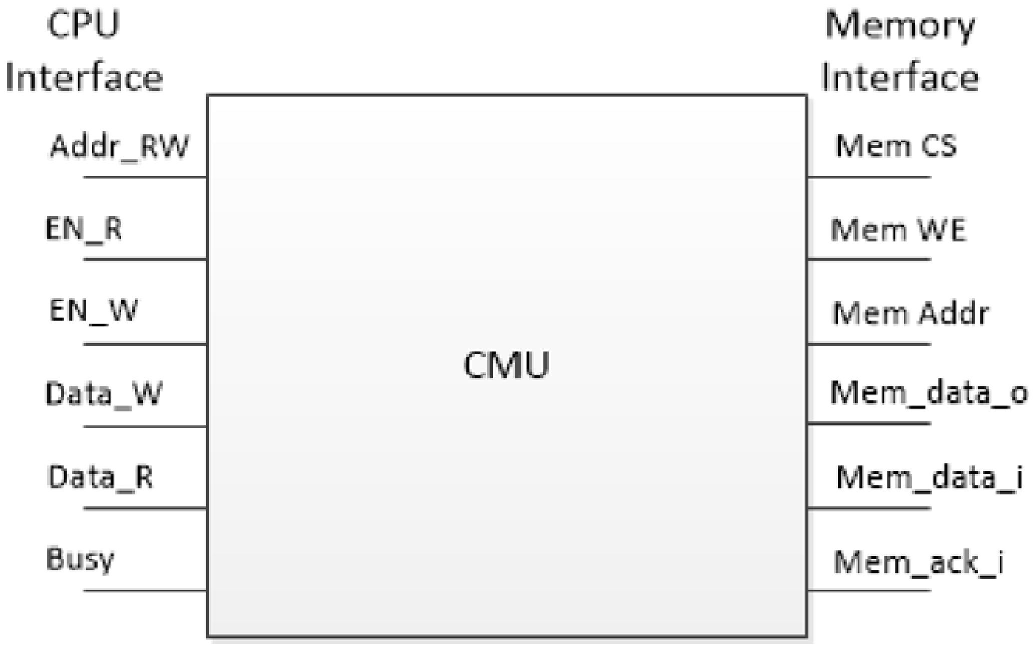
When this work is finished, we shall verify both the simulation results and the on-board results of a preset program.

2 Contents and principles

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

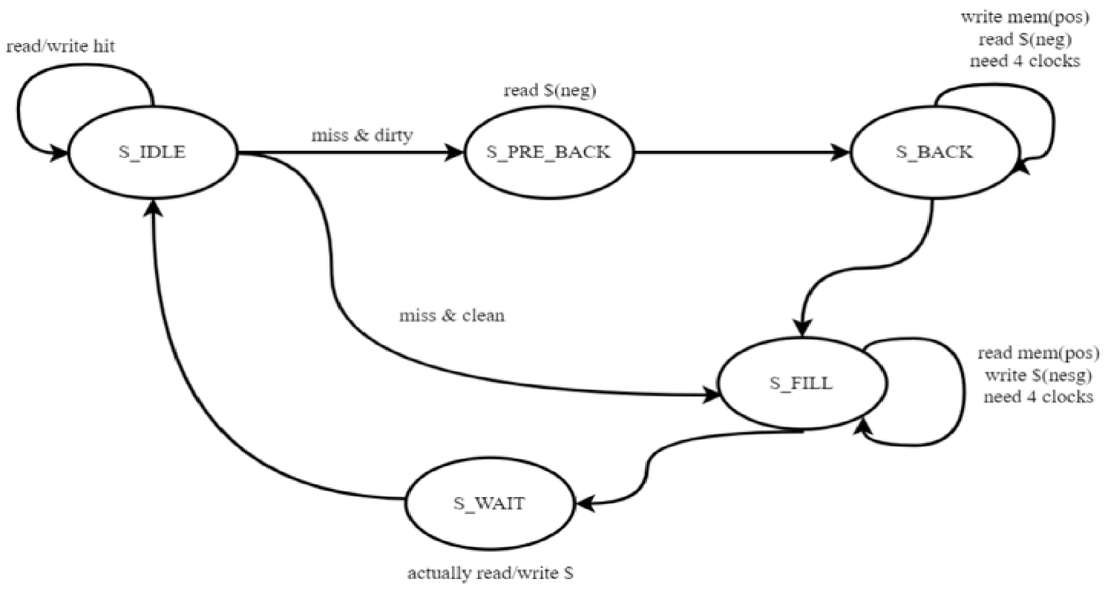
2.1 Cache Controller

Cache Controller is the unit for CPU-Cache communication and control. The cache controller we need to implement has following inputs and outputs, as a special instance of CPU-Memory communication:



In fact, cache controller is used to control the **load**, **edit** or **store** operations of cache. How to arrange these operations will be explained later.

2.2 Cache Automaton



To deal with the **hit**, **dirty miss** or **clean miss** conditions, we introduced the shown automaton. The automaton has 5 determined states: **S_IDLE**, **S_PRE_BACK**, **S_BACK**, **S_FILL**, **S_WAIT**.

S_IDLE is used to indicate that the cache is idle and does not need do anything else. If the CPU does not need to access the cache or the cache is experiencing a **hit**, the state maintains. If encountering a dirty miss, then it transfers to **S_PRE_BACK**; If a clean miss, then it transfers to **S_FILL**.

S_PRE_BACK is for the preparation of **S_BACK**. It sets the word count to 0 and load the first word from the cache for writing back to memory. It must transfer to **S_BACK** to finish following writes.

S_BACK would do the following 4 writes to the lower memory. It increases the word count by 1 at each clock cycle, write the word back to memory, and read another word from cache. When the operations are finished after 4 cycles, it will transfer to **S_FILL**, otherwise it will maintain.

S_FILL loads 4 words from memory. It increases the word count by 1 at each clock cycle, read that word from memory, and write this word to cache. When the operations are finished after 4 cycles, it will transfer to **S_WAIT**, otherwise it will maintain.

S_WAIT deals with the real query about the cache that produced that miss. At **S_WAIT** we can guarantee it produces a hit according to the automaton. Then the whole process after miss is finished, and the state will transfer to **S_IDLE**.

3 Steps and data records

3.1 Completed Verilog source files

3.1.1 cmu.v

For clarity, we will omit some pre-set code segments, and only analysis the code blocks that we need to fill in.

The most important parts we need to finish is the parts that concerns the automaton, in the always@* block.

In fact, the code filled here is quite simple and we need not to explain much. The principles we need can be found in part 2.2. Following is the automaton implement:

```
always @ (*) begin
  if (rst) begin
    next_state = S_IDLE;
    next_word_count = 2'b00;
  end
  else begin
  case (state)
  S_IDLE: begin
    if (en_r || en_w) begin
      if (cache_hit)
        next_state = S_IDLE;
      else if (cache_valid && cache_dirty)
        next_state = S_PRE_BACK;
      else
        next_state = S_FILL;
    end
      next_word_count = 2'b00;
    end

  S_PRE_BACK: begin
    next_state = S_BACK;
    next_word_count = 2'b00;
  end

  S_BACK: begin
    if (mem_ack_i && word_count == {ELEMENT_WORDS_WIDTH{1'b1}})
      next_state = S_FILL;
    else
      next_state = S_BACK;
```

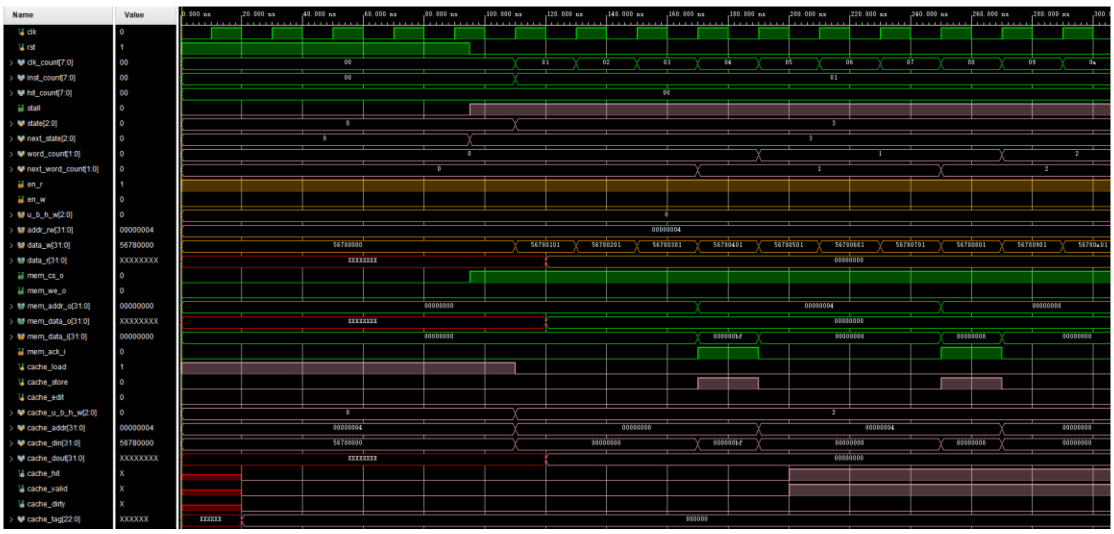
```
        if (mem_ack_i)
            next_word_count = word_count + 1;
        else
            next_word_count = word_count;
    end

    S_FILL: begin
        if (mem_ack_i && word_count == {ELEMENT_WORDS_WIDTH{1'b1}})
            next_state = S_WAIT;
        else
            next_state = S_FILL;
        if (mem_ack_i)
            next_word_count = word_count + 1;
        else
            next_word_count = word_count;
    end

    S_WAIT: begin
        next_state = S_IDLE;
        next_word_count = 2'b00;
    end
    endcase
end
```

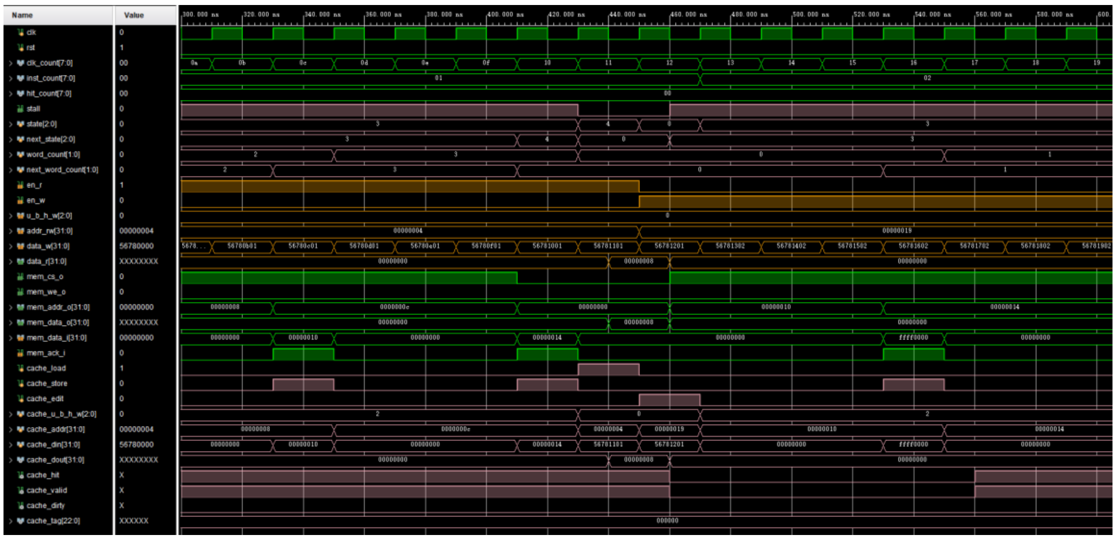
At last, we stall the pipeline when the cache is busy interacting with the memory.

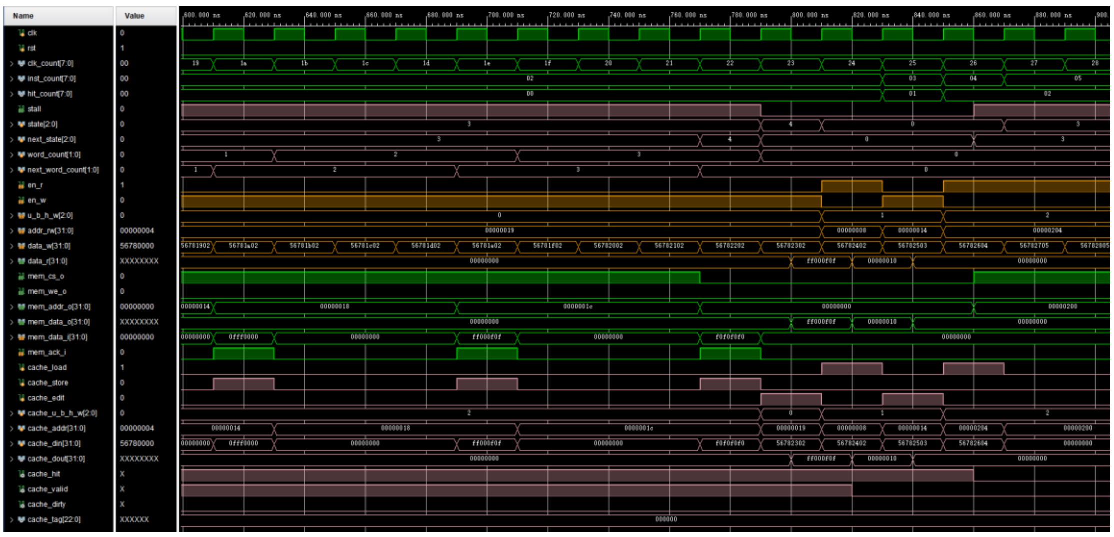
```
assign stall = ~rst & ~(next_state == S_IDLE);
```

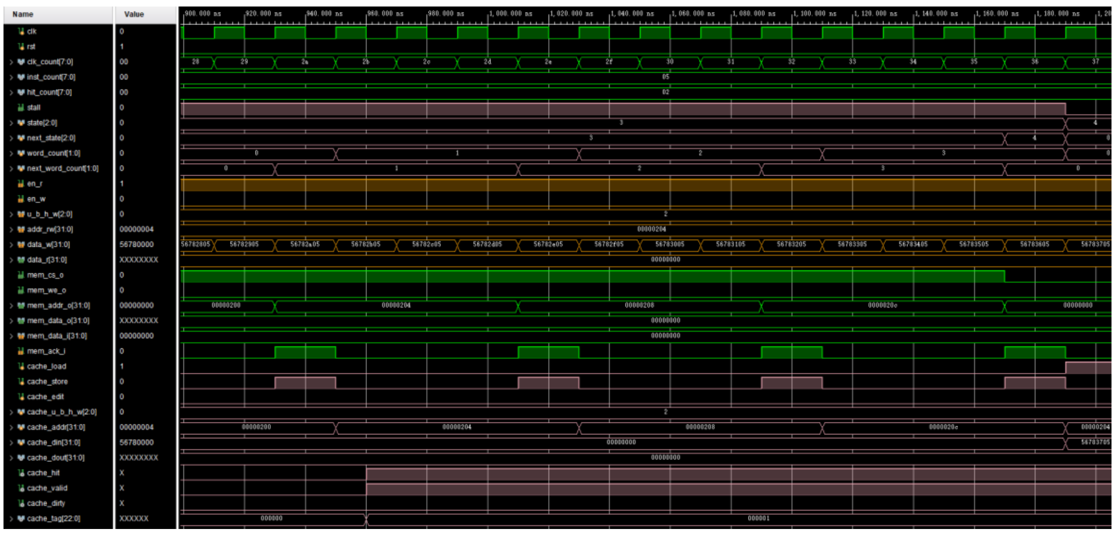
3.2 Implementation results

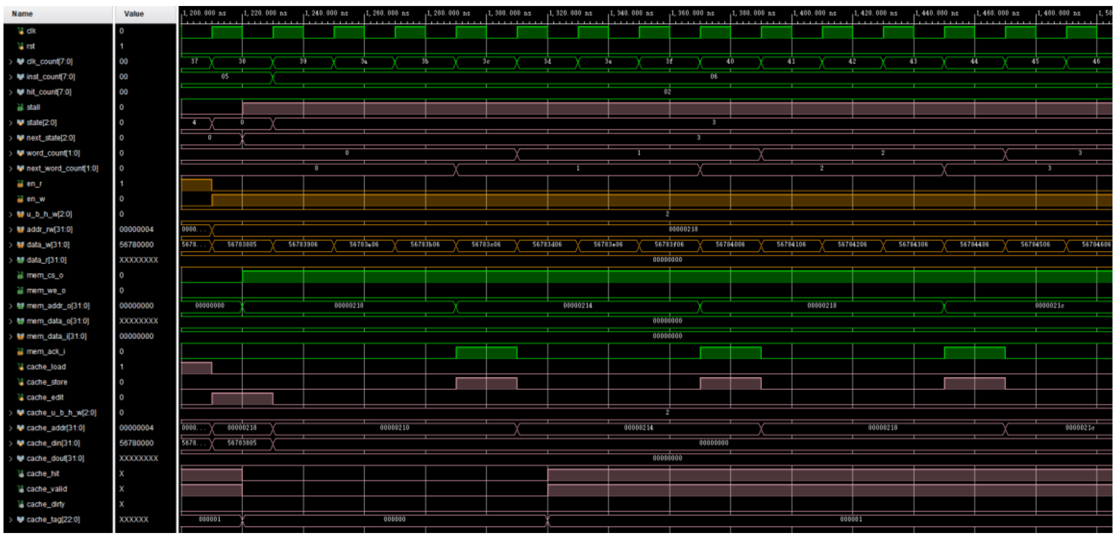
Here we record all our behavioral simulation results. Most will not be used for analysis.

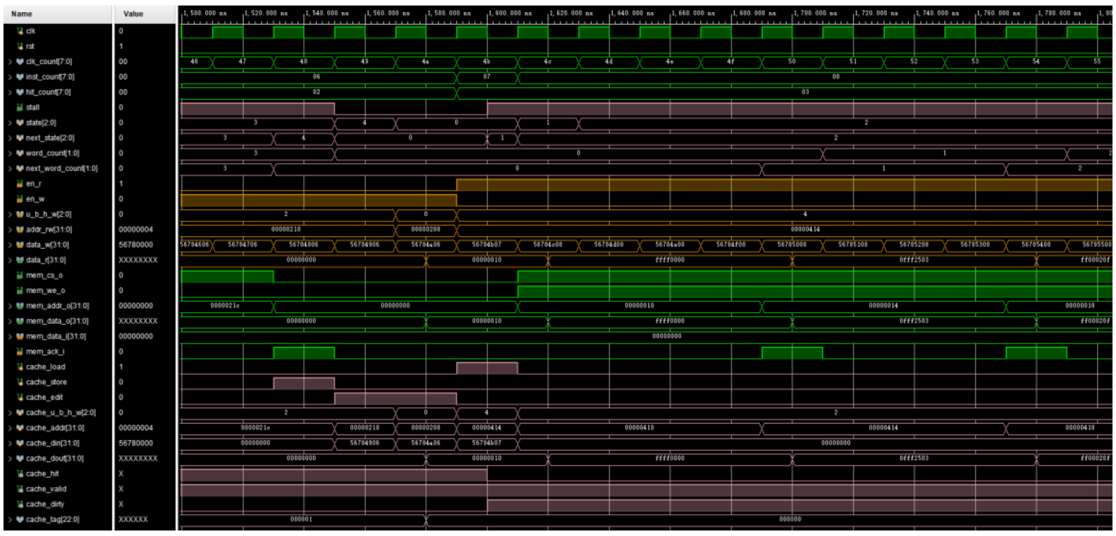

Name
Value
clk
rst
clk_count[7:0]
inst_count[7:0]
hit_count[7:0]
stall
state[2:0]
next_state[2:0]
word_count[1:0]
next_word_count[1:0]
en_r
en_w
u_b_h_w[2:0]
addr_rw[31:0]
data_w[31:0]
data_r[31:0]
mem_cs_o
mem_we_o
mem_addr_o[31:0]
mem_data_o[31:0]
mem_data_i[31:0]
mem_ack_i
cache_load
cache_store
cache_edit
cache_u_b_h_w[2:0]
cache_addr[31:0]
cache_din[31:0]
cache_dout[31:0]
cache_hit
cache_valid
cache_dirty
cache_tag[22:0]

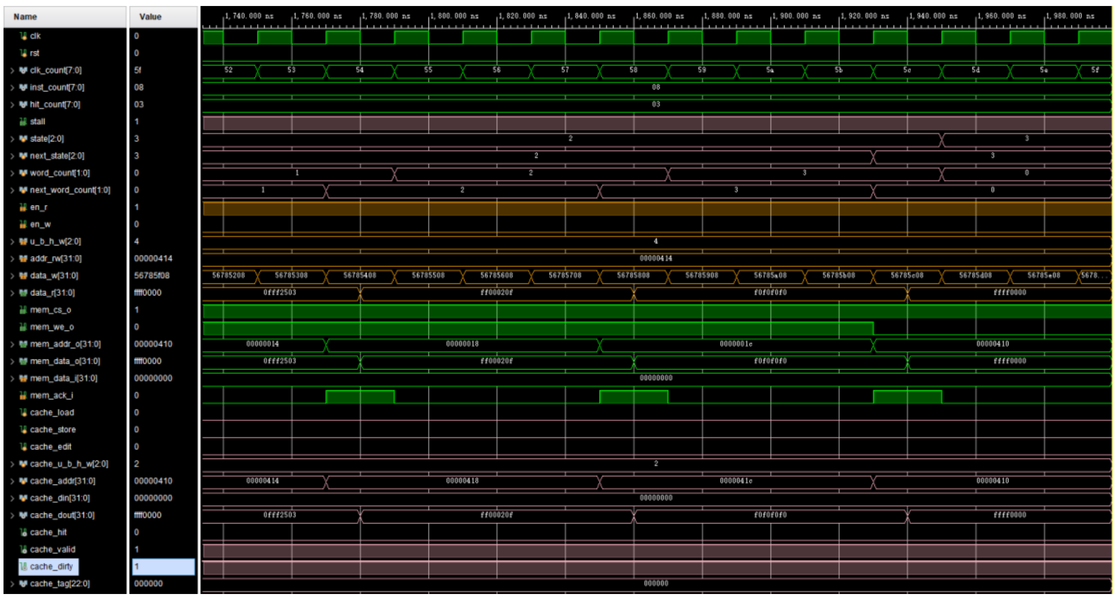

Name
Value
clk
rst
clk_count[7:0]
inst_count[7:0]
hit_count[7:0]
stall
state[2:0]
next_state[2:0]
word_count[1:0]
next_word_count[1:0]
en_r
en_w
u_b_h_w[2:0]
addr_rw[31:0]
data_w[31:0]
data_r[31:0]
mem_cs_o
mem_we_o
mem_addr_o[31:0]
mem_data_o[31:0]
mem_data_i[31:0]
mem_ack_i
cache_load
cache_store
cache_edit
cache_u_b_h_w[2:0]
cache_addr[31:0]
cache_din[31:0]
cache_dout[31:0]
cache_hit
cache_valid
cache_dirty
cache_tag[22:0]


Name
Value
clk
rst
clk_count[7:0]
inst_count[7:0]
hit_count[7:0]
stall
state[2:0]
next_state[2:0]
word_count[1:0]
next_word_count[1:0]
en_r
en_w
u_b_h_w[2:0]
addr_rw[31:0]
data_w[31:0]
data_r[31:0]
mem_cs_o
mem_we_o
mem_addr_o[31:0]
mem_data_o[31:0]
mem_data_i[31:0]
mem_ack_i
cache_load
cache_store
cache_edit
cache_u_b_h_w[2:0]
cache_addr[31:0]
cache_din[31:0]
cache_dout[31:0]
cache_hit
cache_valid
cache_dirty
cache_tag[22:0]


Name
Value
clk
rst
clk_count[7:0]
inst_count[7:0]
hit_count[7:0]
stall
state[2:0]
next_state[2:0]
word_count[1:0]
next_word_count[1:0]
en_r
en_w
u_b_h_w[2:0]
addr_rw[31:0]
data_w[31:0]
data_r[31:0]
mem_cs_o
mem_we_o
mem_addr_o[31:0]
mem_data_o[31:0]
mem_data_i[31:0]
mem_ack_i
cache_load
cache_store
cache_edit
cache_u_b_h_w[2:0]
cache_addr[31:0]
cache_din[31:0]
cache_dout[31:0]
cache_hit
cache_valid
cache_dirty
cache_tag[22:0]


Name
Value
clk
rst
clk_count[7:0]
inst_count[7:0]
hit_count[7:0]
stall
state[2:0]
next_state[2:0]
word_count[1:0]
next_word_count[1:0]
en_r
en_w
u_b_h_w[2:0]
addr_rw[31:0]
data_w[31:0]
data_r[31:0]
mem_cs_o
mem_we_o
mem_addr_o[31:0]
mem_data_o[31:0]
mem_data_i[31:0]
mem_ack_i
cache_load
cache_store
cache_edit
cache_u_b_h_w[2:0]
cache_addr[31:0]
cache_din[31:0]
cache_dout[31:0]
cache_hit
cache_valid
cache_dirty
cache_tag[22:0]

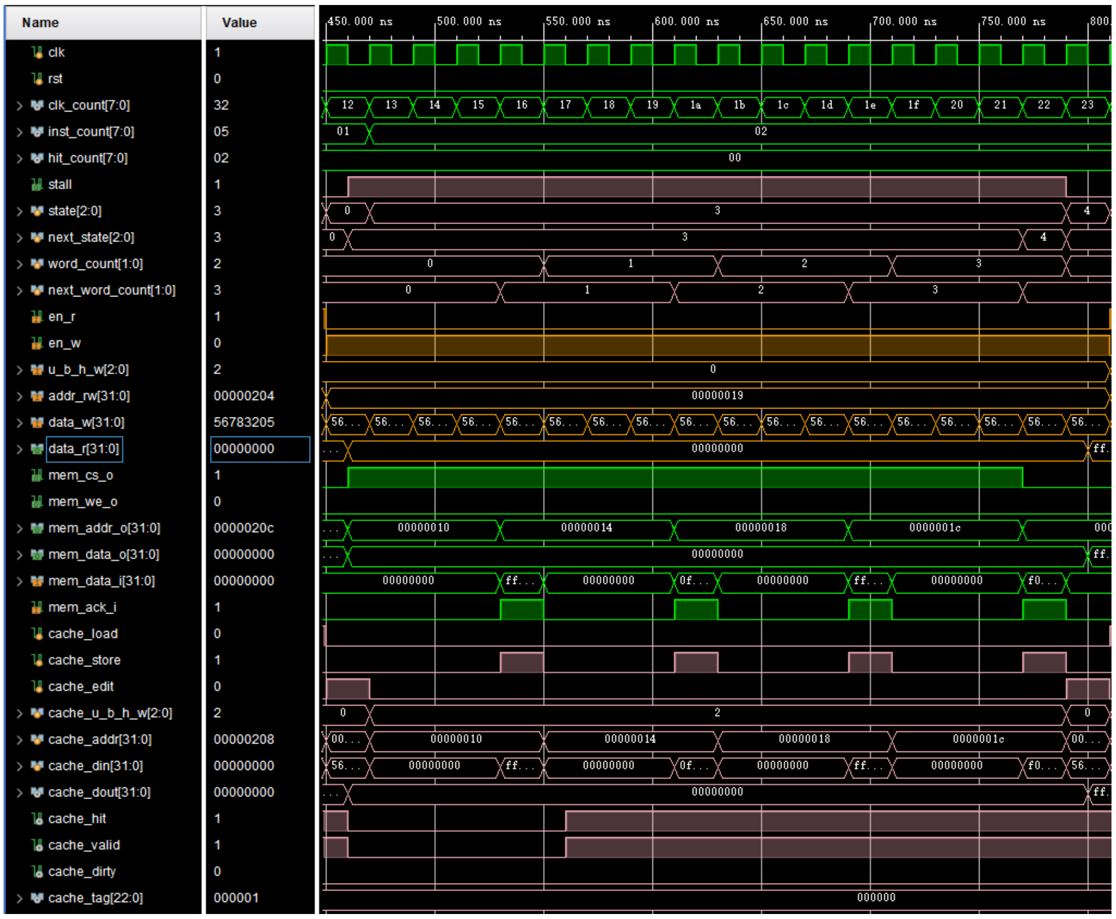

Name
Value
clk
rst
clk_count[7:0]
inst_count[7:0]
hit_count[7:0]
stall
state[2:0]
next_state[2:0]
word_count[1:0]
next_word_count[1:0]
en_r
en_w
u_b_h_w[2:0]
addr_rw[31:0]
data_w[31:0]
data_r[31:0]
mem_cs_o
mem_we_o
mem_addr_o[31:0]
mem_data_o[31:0]
mem_data_i[31:0]
mem_ack_i
cache_load
cache_store
cache_edit
cache_u_b_h_w[2:0]
cache_addr[31:0]
cache_din[31:0]
cache_dout[31:0]
cache_hit
cache_valid
cache_dirty
cache_tag[22:0]



4 Analysis of the results

Here we only discuss some typical results.

4.1 Clean miss



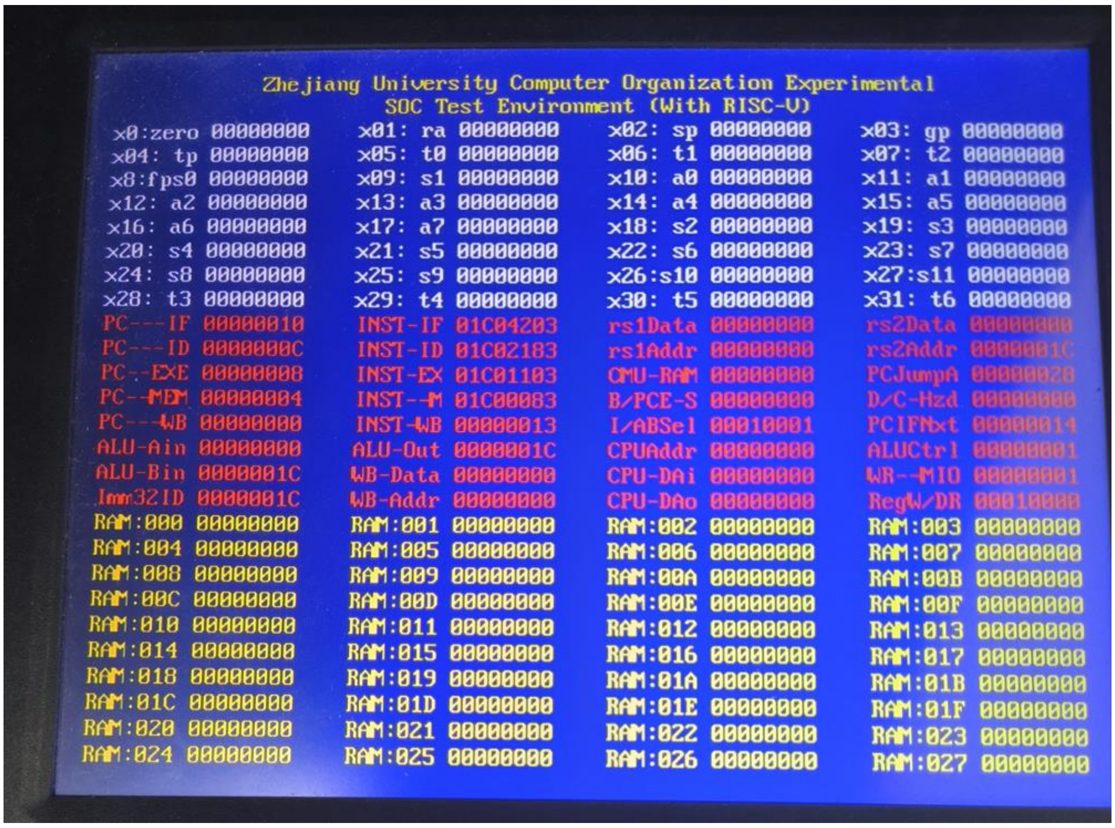
Consider the simulation in 450-810ns.

From 450ns, a **clean miss** happened. The state is transferred to **S_FILL**, and the cache was gradually written (according to the **cache_din** signal). At the end of **S_FILL**, it transfers to **S_WAIT**, and the cache output was given out.

Meanwhile, the whole pipeline is stalled.

After the process the data in memory is put in cache, we could see the process is successful.

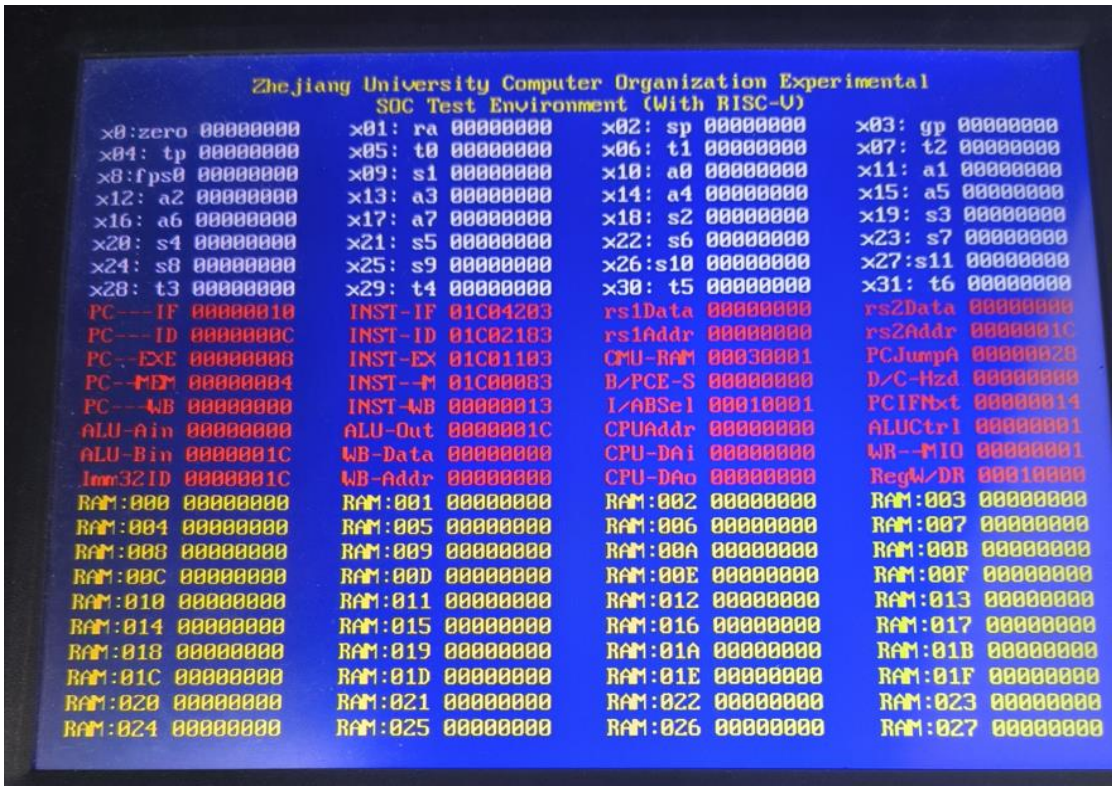
The on-board results of a clean miss are:



```
                Zhejiang University Computer Organization Experimental
                          SOC Test Environment (With RISC-V)
x0:zero 00000000    x01: ra 00000000    x02: sp 00000000    x03: gp 00000000
x04: tp 00000000    x05: t0 00000000    x06: t1 00000000    x07: t2 00000000
x8:fps0 00000000    x09: s1 00000000    x10: a0 00000000    x11: a1 00000000
x12: a2 00000000    x13: a3 00000000    x14: a4 00000000    x15: a5 00000000
x16: a6 00000000    x17: a7 00000000    x18: s2 00000000    x19: s3 00000000
x20: s4 00000000    x21: s5 00000000    x22: s6 00000000    x23: s7 00000000
x24: s8 00000000    x25: s9 00000000    x26:s10 00000000    x27:s11 00000000
x28: t3 00000000    x29: t4 00000000    x30: t5 00000000    x31: t6 00000000
PC---IF 00000010    INST-IF 01C04203    rs1Data 00000000    rs2Data 00000000
PC---ID 0000000C    INST-ID 01C02183    rs1Addr 00000000    rs2Addr 0000001C
PC--EXE 00000008    INST-EX 01C01103    CMU-RAM 00000000    PCJumpA 00000028
PC--MEM 00000004    INST--M 01C00083    B/PCE-S 00000000    D/C-Hzd 00000000
PC---WB 00000000    INST-WB 00000013    I/ABSel 00010001    PCIFNxt 00000014
ALU-Ain 00000000    ALU-Out 0000001C    CPUAddr 00000000    ALUCtrl 00000001
ALU-Bin 0000001C    WB-Data 00000000    CPU-DAi 00000000    WR--MIO 00000001
Imm32ID 0000001C    WB-Addr 00000000    CPU-DAo 00000000    RegW/DR 00010000
RAM:000 00000000    RAM:001 00000000    RAM:002 00000000    RAM:003 00000000
RAM:004 00000000    RAM:005 00000000    RAM:006 00000000    RAM:007 00000000
RAM:008 00000000    RAM:009 00000000    RAM:00A 00000000    RAM:00B 00000000
RAM:00C 00000000    RAM:00D 00000000    RAM:00E 00000000    RAM:00F 00000000
RAM:010 00000000    RAM:011 00000000    RAM:012 00000000    RAM:013 00000000
RAM:014 00000000    RAM:015 00000000    RAM:016 00000000    RAM:017 00000000
RAM:018 00000000    RAM:019 00000000    RAM:01A 00000000    RAM:01B 00000000
RAM:01C 00000000    RAM:01D 00000000    RAM:01E 00000000    RAM:01F 00000000
RAM:020 00000000    RAM:021 00000000    RAM:022 00000000    RAM:023 00000000
RAM:024 00000000    RAM:025 00000000    RAM:026 00000000    RAM:027 00000000
```

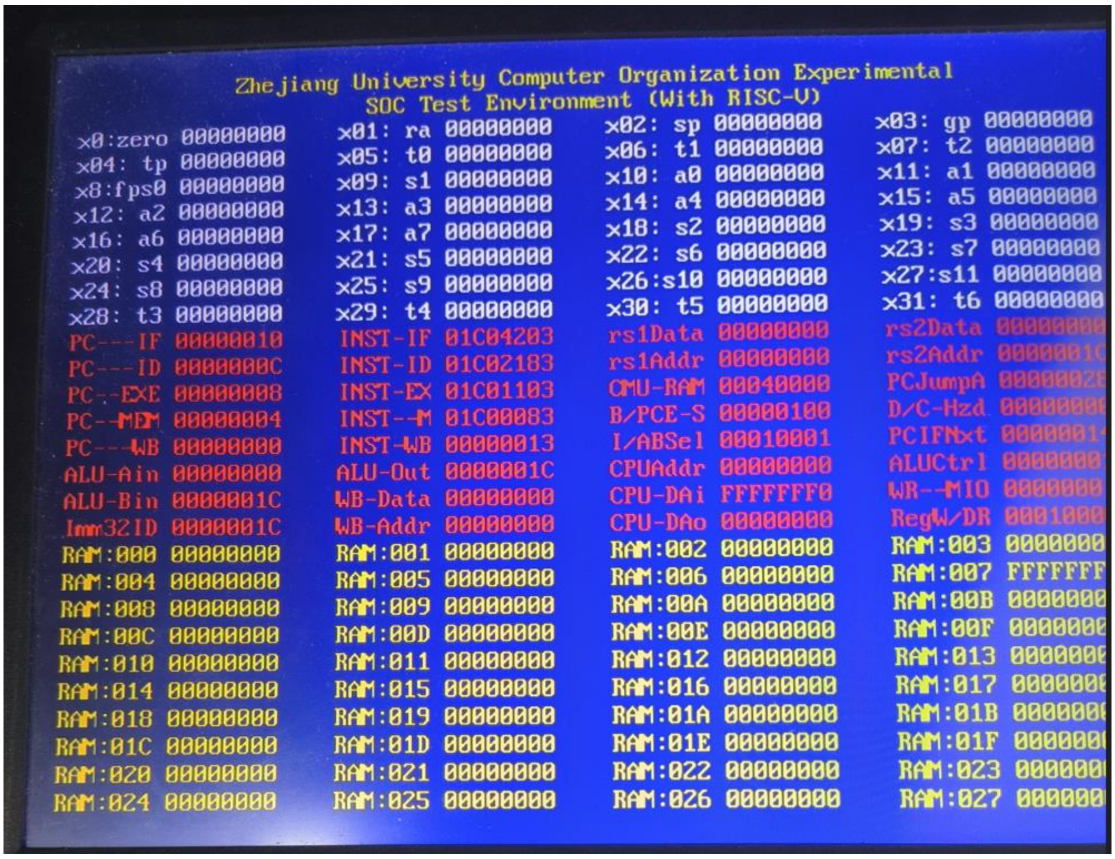
S_IDLE.

=



```
          Zhejiang University Computer Organization Experimental
                  SOC Test Environment (With RISC-V)
x0:zero 00000000   x01: ra 00000000   x02: sp 00000000    x03: gp 00000000
x04: tp 00000000   x05: t0 00000000   x06: t1 00000000    x07: t2 00000000
x8:fps0 00000000   x09: s1 00000000   x10: a0 00000000    x11: a1 00000000
x12: a2 00000000   x13: a3 00000000   x14: a4 00000000    x15: a5 00000000
x16: a6 00000000   x17: a7 00000000   x18: s2 00000000    x19: s3 00000000
x20: s4 00000000   x21: s5 00000000   x22: s6 00000000    x23: s7 00000000
x24: s8 00000000   x25: s9 00000000   x26:s10 00000000    x27:s11 00000000
x28: t3 00000000   x29: t4 00000000   x30: t5 00000000    x31: t6 00000000
PC---IF 00000010   INST-IF 01C04203   rs1Data 00000000    rs2Data 00000000
PC---ID 0000000C   INST-ID 01C02183   rs1Addr 00000000    rs2Addr 0000001C
PC--EXE 00000008   INST-EX 01C01103   CMU-RAM 00030001    PCJumpA 00000028
PC--MEM 00000004   INST--M 01C00083   B/PCE-S 00000000    D/C-Hzd 00000000
PC---WB 00000000   INST-WB 00000013   I/ABSel 00010001    PCIFNxt 00000014
ALU-Ain 00000000   ALU-Out 0000001C   CPUAddr 00000000    ALUCtrl 00000001
ALU-Bin 0000001C   WB-Data 00000000   CPU-DAi 00000000    WR--MIO 00000001
Imm32ID 0000001C   WB-Addr 00000000   CPU-DAo 00000000    RegW/DR 00010000
RAM:000 00000000   RAM:001 00000000   RAM:002 00000000    RAM:003 00000000
RAM:004 00000000   RAM:005 00000000   RAM:006 00000000    RAM:007 00000000
RAM:008 00000000   RAM:009 00000000   RAM:00A 00000000    RAM:00B 00000000
RAM:00C 00000000   RAM:00D 00000000   RAM:00E 00000000    RAM:00F 00000000
RAM:010 00000000   RAM:011 00000000   RAM:012 00000000    RAM:013 00000000
RAM:014 00000000   RAM:015 00000000   RAM:016 00000000    RAM:017 00000000
RAM:018 00000000   RAM:019 00000000   RAM:01A 00000000    RAM:01B 00000000
RAM:01C 00000000   RAM:01D 00000000   RAM:01E 00000000    RAM:01F 00000000
RAM:020 00000000   RAM:021 00000000   RAM:022 00000000    RAM:023 00000000
RAM:024 00000000   RAM:025 00000000   RAM:026 00000000    RAM:027 00000000
```

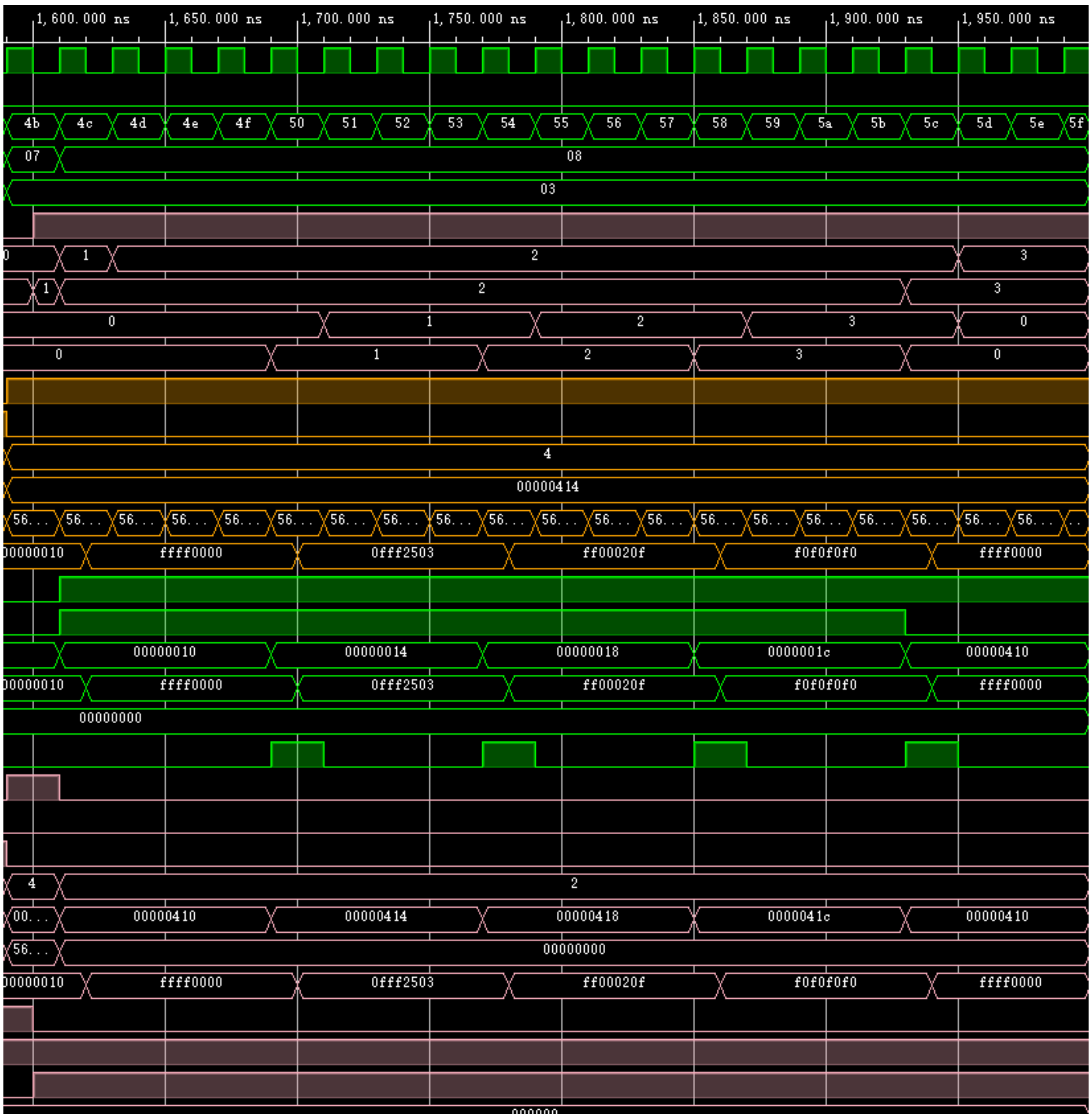
The start of **S_FILL** state. After this moment there are RAM automaton running to get the required data. The process will repeat 4 times so we omitted some graphs.



```
          Zhejiang University Computer Organization Experimental
                  SOC Test Environment (With RISC-V)
x0:zero 00000000   x01: ra 00000000   x02: sp 00000000    x03: gp 00000000
x04: tp 00000000   x05: t0 00000000   x06: t1 00000000    x07: t2 00000000
x8:fps0 00000000   x09: s1 00000000   x10: a0 00000000    x11: a1 00000000
x12: a2 00000000   x13: a3 00000000   x14: a4 00000000    x15: a5 00000000
x16: a6 00000000   x17: a7 00000000   x18: s2 00000000    x19: s3 00000000
x20: s4 00000000   x21: s5 00000000   x22: s6 00000000    x23: s7 00000000
x24: s8 00000000   x25: s9 00000000   x26:s10 00000000    x27:s11 00000000
x28: t3 00000000   x29: t4 00000000   x30: t5 00000000    x31: t6 00000000
PC---IF 00000010   INST-IF 01C04203   rs1Data 00000000    rs2Data 00000000
PC---ID 0000000C   INST-ID 01C02183   rs1Addr 00000000    rs2Addr 0000001C
PC--EXE 00000008   INST-EX 01C01103   CMU-RAM 00040000    PCJumpA 00000028
PC--MEM 00000004   INST--M 01C00083   B/PCE-S 00000100    D/C-Hzd 00000000
PC---WB 00000000   INST-WB 00000013   I/ABSel 00010001    PCIFNxt 0000001
ALU-Ain 00000000   ALU-Out 0000001C   CPUAddr 00000000    ALUCtrl 0000000
ALU-Bin 0000001C   WB-Data 00000000   CPU-DAi FFFFFFF0    WR--MIO 0000000
Imm32ID 0000001C   WB-Addr 00000000   CPU-DAo 00000000    RegW/DR 0001000
RAM:000 00000000   RAM:001 00000000   RAM:002 00000000    RAM:003 0000000
RAM:004 00000000   RAM:005 00000000   RAM:006 00000000    RAM:007 FFFFFFF
RAM:008 00000000   RAM:009 00000000   RAM:00A 00000000    RAM:00B 0000000
RAM:00C 00000000   RAM:00D 00000000   RAM:00E 00000000    RAM:00F 0000000
RAM:010 00000000   RAM:011 00000000   RAM:012 00000000    RAM:013 0000000
RAM:014 00000000   RAM:015 00000000   RAM:016 00000000    RAM:017 0000000
RAM:018 00000000   RAM:019 00000000   RAM:01A 00000000    RAM:01B 0000000
RAM:01C 00000000   RAM:01D 00000000   RAM:01E 00000000    RAM:01F 000000
RAM:020 00000000   RAM:021 00000000   RAM:022 00000000    RAM:023 000000
RAM:024 00000000   RAM:025 00000000   RAM:026 00000000    RAM:027 000000
```

The **S_WAIT** state at end.

4.2 Dirty miss



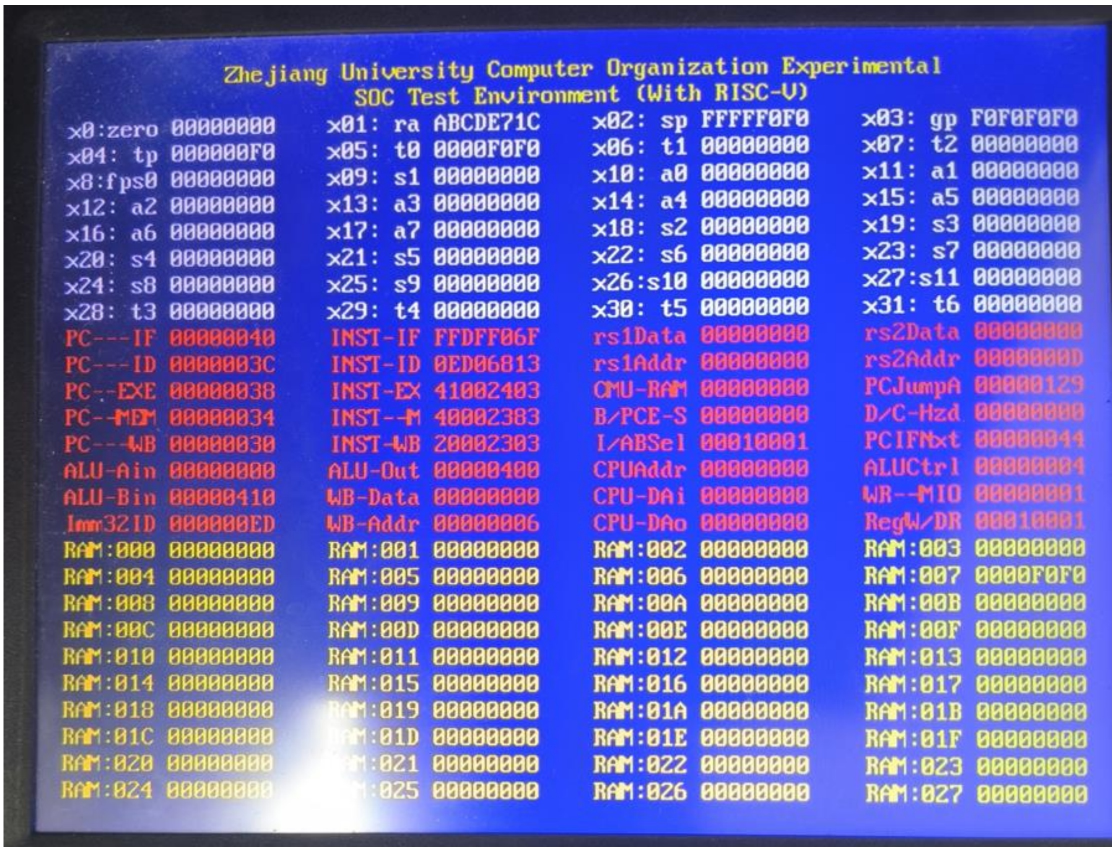
1, 600. 000 ns
1, 650. 000 ns
1, 700. 000 ns
1, 750. 000 ns
1, 800. 000 ns
1, 850. 000 ns
1, 900. 000 ns
1, 950. 000 ns
4b 4c 4d 4e 4f 50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f
07 08
03
0 1 2 3
1 2 3
0 1 2 3 0
0 1 2 3 0
4
00000414
00000010 ffff0000 0fff2503 ff00020f f0f0f0f0 ffff0000
00000010 00000014 00000018 0000001c 00000410
00000010 ffff0000 0fff2503 ff00020f f0f0f0f0 ffff0000
00000000
4 2
00... 00000410 00000414 00000418 0000041c 00000410
56... 00000000
00000010 ffff0000 0fff2503 ff00020f f0f0f0f0 ffff0000

Consider the result in 1590ns-2000ns.

In this period, a **dirty miss** is detected, as we found the **cache_dirty** signal is valid. The state is first transferred to **S_PRE_BACK**, then **S_BACK** for some clock cycles.

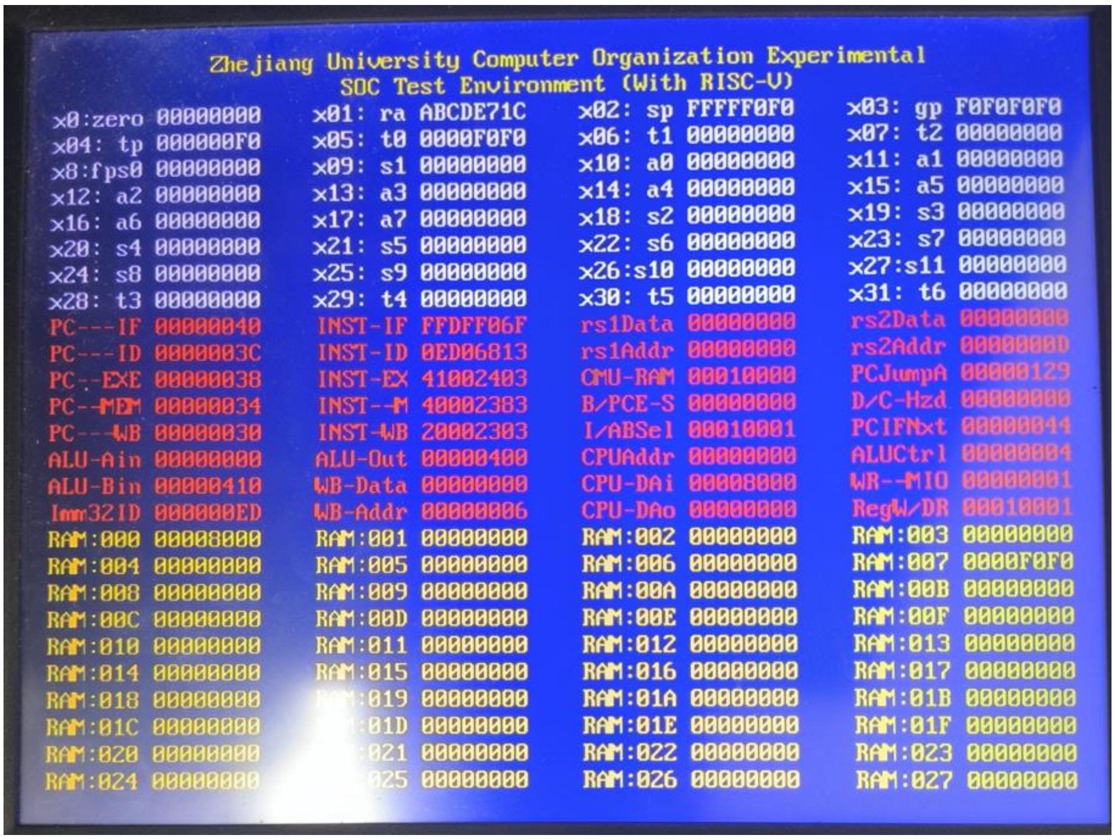
In these clock cycles, the memory in cache was written to the main memory according to the write-back principle. After that the state transferred to **S_FILL**, and from that time on the process is similar to a **clean miss**.

The on-board results of a clean miss are:



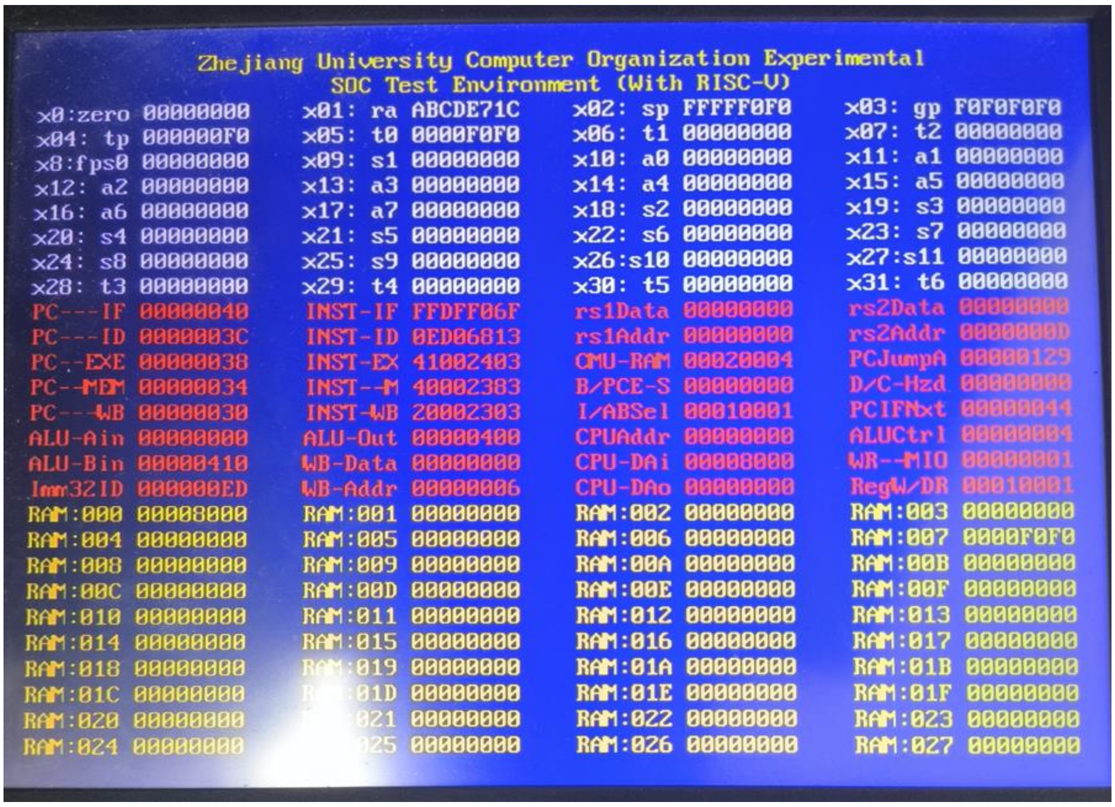
```
        Zhejiang University Computer Organization Experimental
                  SOC Test Environment (With RISC-V)
x0:zero 00000000   x01: ra ABCDE71C   x02: sp FFFFF0F0   x03: gp F0F0F0F0
x04: tp 000000F0   x05: t0 0000F0F0   x06: t1 00000000   x07: t2 00000000
x8:fps0 00000000   x09: s1 00000000   x10: a0 00000000   x11: a1 00000000
x12: a2 00000000   x13: a3 00000000   x14: a4 00000000   x15: a5 00000000
x16: a6 00000000   x17: a7 00000000   x18: s2 00000000   x19: s3 00000000
x20: s4 00000000   x21: s5 00000000   x22: s6 00000000   x23: s7 00000000
x24: s8 00000000   x25: s9 00000000   x26:s10 00000000   x27:s11 00000000
x28: t3 00000000   x29: t4 00000000   x30: t5 00000000   x31: t6 00000000
PC---IF 00000040   INST-IF FFDFF06F   rs1Data 00000000   rs2Data 00000000
PC---ID 0000003C   INST-ID 0ED06813   rs1Addr 00000000   rs2Addr 0000000D
PC--EXE 00000038   INST-EX 41002403   CMU-RAM 00000000   PCJumpA 00000129
PC--MEM 00000034   INST--M 40002383   B/PCE-S 00000000   D/C-Hzd 00000000
PC---WB 00000030   INST-WB 20002303   I/ABSel 00010001   PCIFNxt 00000044
ALU-Ain 00000000   ALU-Out 00000400   CPUAddr 00000000   ALUCtrl 00000004
ALU-Bin 00000410   WB-Data 00000000   CPU-DAi 00000000   WR--MIO 00000001
Imm32ID 000000ED   WB-Addr 00000006   CPU-DAo 00000000   RegW/DR 00010001
RAM:000 00000000   RAM:001 00000000   RAM:002 00000000   RAM:003 00000000
RAM:004 00000000   RAM:005 00000000   RAM:006 00000000   RAM:007 0000F0F0
RAM:008 00000000   RAM:009 00000000   RAM:00A 00000000   RAM:00B 00000000
RAM:00C 00000000   RAM:00D 00000000   RAM:00E 00000000   RAM:00F 00000000
RAM:010 00000000   RAM:011 00000000   RAM:012 00000000   RAM:013 00000000
RAM:014 00000000   RAM:015 00000000   RAM:016 00000000   RAM:017 00000000
RAM:018 00000000   RAM:019 00000000   RAM:01A 00000000   RAM:01B 00000000
RAM:01C 00000000   RAM:01D 00000000   RAM:01E 00000000   RAM:01F 00000000
RAM:020 00000000   RAM:021 00000000   RAM:022 00000000   RAM:023 00000000
RAM:024 00000000   RAM:025 00000000   RAM:026 00000000   RAM:027 00000000
```

S_IDLE.



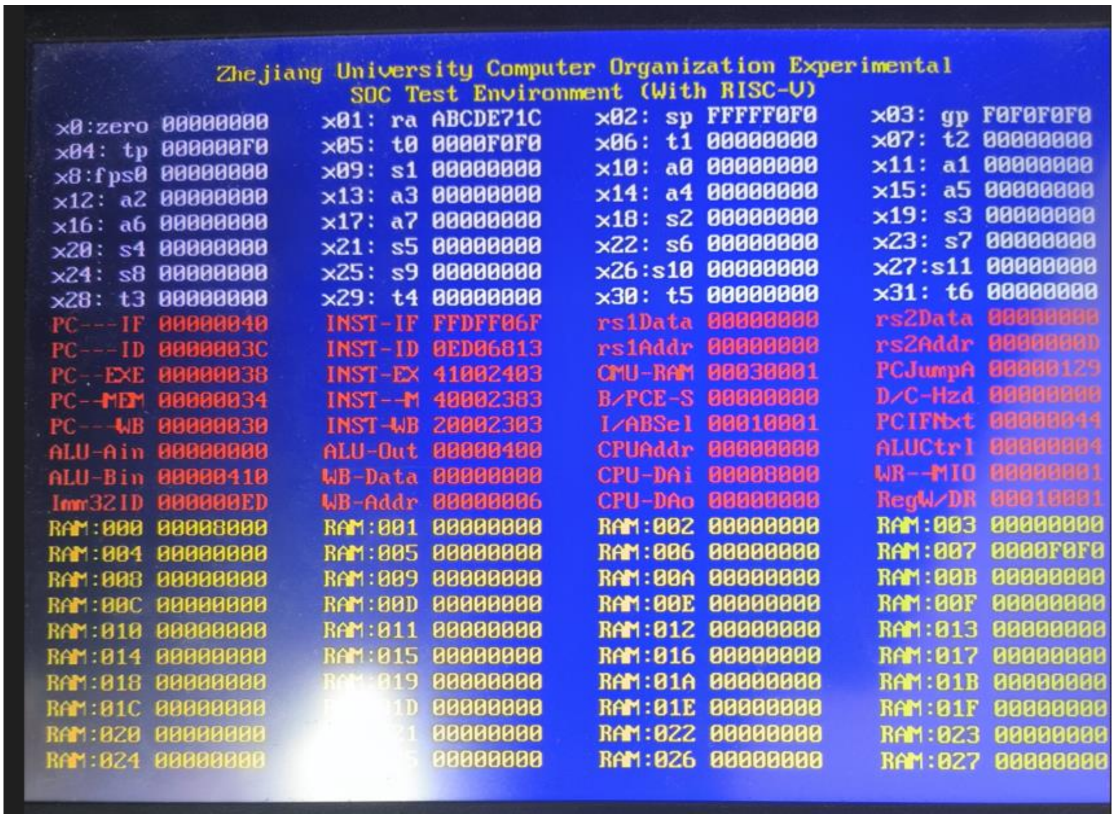
```
        Zhejiang University Computer Organization Experimental
                  SOC Test Environment (With RISC-V)
x0:zero 00000000   x01: ra ABCDE71C   x02: sp FFFFF0F0   x03: gp F0F0F0F0
x04: tp 000000F0   x05: t0 0000F0F0   x06: t1 00000000   x07: t2 00000000
x8:fps0 00000000   x09: s1 00000000   x10: a0 00000000   x11: a1 00000000
x12: a2 00000000   x13: a3 00000000   x14: a4 00000000   x15: a5 00000000
x16: a6 00000000   x17: a7 00000000   x18: s2 00000000   x19: s3 00000000
x20: s4 00000000   x21: s5 00000000   x22: s6 00000000   x23: s7 00000000
x24: s8 00000000   x25: s9 00000000   x26:s10 00000000   x27:s11 00000000
x28: t3 00000000   x29: t4 00000000   x30: t5 00000000   x31: t6 00000000
PC---IF 00000040   INST-IF FFDFF06F   rs1Data 00000000   rs2Data 00000000
PC---ID 0000003C   INST-ID 0ED06813   rs1Addr 00000000   rs2Addr 0000000D
PC--EXE 00000038   INST-EX 41002403   CMU-RAM 00010000   PCJumpA 00000129
PC--MEM 00000034   INST--M 40002383   B/PCE-S 00000000   D/C-Hzd 00000000
PC---WB 00000030   INST-WB 20002303   I/ABSel 00010001   PCIFNxt 00000044
ALU-Ain 00000000   ALU-Out 00000400   CPUAddr 00000000   ALUCtrl 00000004
ALU-Bin 00000410   WB-Data 00000000   CPU-DAi 00008000   WR--MIO 00000001
Imm32ID 000000ED   WB-Addr 00000006   CPU-DAo 00000000   RegW/DR 00010001
RAM:000 00008000   RAM:001 00000000   RAM:002 00000000   RAM:003 00000000
RAM:004 00000000   RAM:005 00000000   RAM:006 00000000   RAM:007 0000F0F0
RAM:008 00000000   RAM:009 00000000   RAM:00A 00000000   RAM:00B 00000000
RAM:00C 00000000   RAM:00D 00000000   RAM:00E 00000000   RAM:00F 00000000
RAM:010 00000000   RAM:011 00000000   RAM:012 00000000   RAM:013 00000000
RAM:014 00000000   RAM:015 00000000   RAM:016 00000000   RAM:017 00000000
RAM:018 00000000   RAM:019 00000000   RAM:01A 00000000   RAM:01B 00000000
RAM:01C 00000000   RAM:01D 00000000   RAM:01E 00000000   RAM:01F 00000000
RAM:020 00000000   RAM:021 00000000   RAM:022 00000000   RAM:023 00000000
RAM:024 00000000   RAM:025 00000000   RAM:026 00000000   RAM:027 00000000
```

S_PRE_BACK.



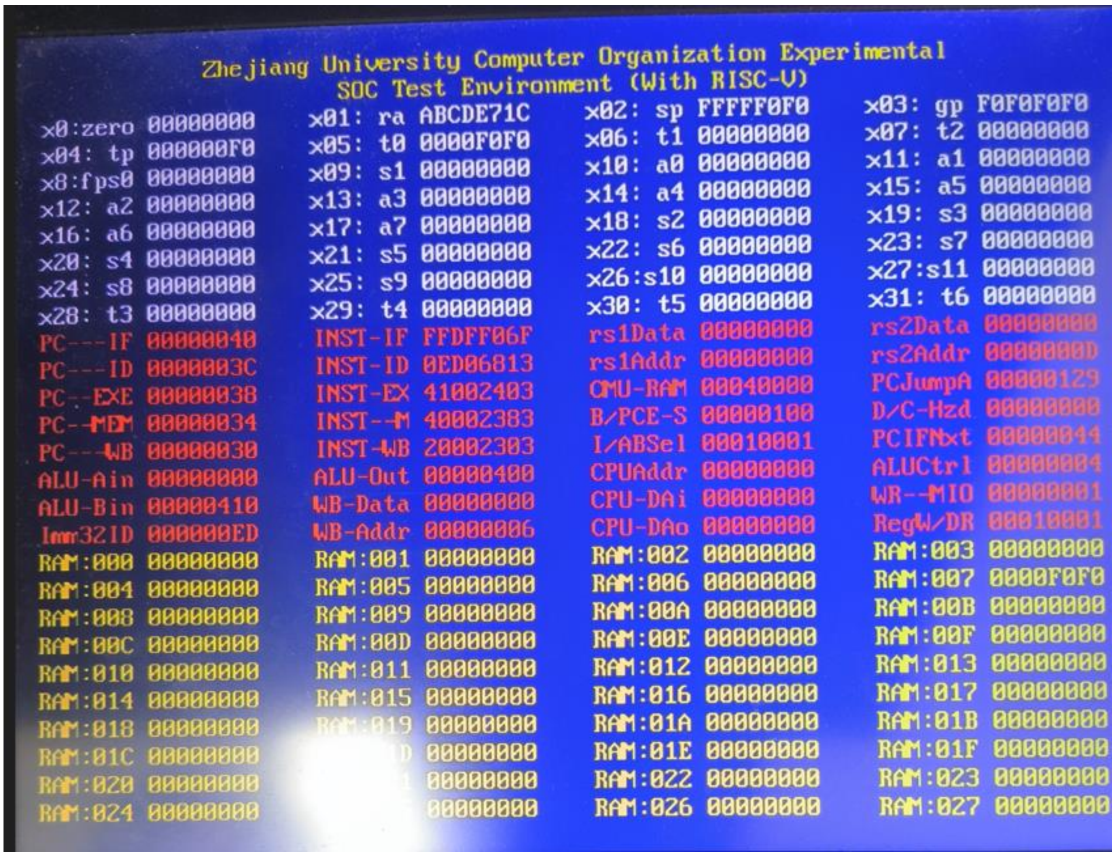
```
        Zhejiang University Computer Organization Experimental
                  SOC Test Environment (With RISC-V)
x0:zero 00000000   x01: ra ABCDE71C   x02: sp FFFFF0F0   x03: gp F0F0F0F0
x04: tp 000000F0   x05: t0 0000F0F0   x06: t1 00000000   x07: t2 00000000
x8:fps0 00000000   x09: s1 00000000   x10: a0 00000000   x11: a1 00000000
x12: a2 00000000   x13: a3 00000000   x14: a4 00000000   x15: a5 00000000
x16: a6 00000000   x17: a7 00000000   x18: s2 00000000   x19: s3 00000000
x20: s4 00000000   x21: s5 00000000   x22: s6 00000000   x23: s7 00000000
x24: s8 00000000   x25: s9 00000000   x26:s10 00000000   x27:s11 00000000
x28: t3 00000000   x29: t4 00000000   x30: t5 00000000   x31: t6 00000000
PC---IF 00000040   INST-IF FFDFF06F   rs1Data 00000000   rs2Data 00000000
PC---ID 0000003C   INST-ID 0ED06813   rs1Addr 00000000   rs2Addr 0000000D
PC--EXE 00000038   INST-EX 41002403   CMU-RAM 00020004   PCJumpA 00000129
PC--MEM 00000034   INST--M 40002383   B/PCE-S 00000000   D/C-Hzd 00000000
PC---WB 00000030   INST-WB 20002303   I/ABSel 00010001   PCIFNxt 00000044
ALU-Ain 00000000   ALU-Out 00000400   CPUAddr 00000000   ALUCtrl 00000004
ALU-Bin 00000410   WB-Data 00000000   CPU-DAi 00008000   WR--MIO 00000001
Imm32ID 000000ED   WB-Addr 00000006   CPU-DAo 00000000   RegW/DR 00010001
RAM:000 00008000   RAM:001 00000000   RAM:002 00000000   RAM:003 00000000
RAM:004 00000000   RAM:005 00000000   RAM:006 00000000   RAM:007 0000F0F0
RAM:008 00000000   RAM:009 00000000   RAM:00A 00000000   RAM:00B 00000000
RAM:00C 00000000   RAM:00D 00000000   RAM:00E 00000000   RAM:00F 00000000
RAM:010 00000000   RAM:011 00000000   RAM:012 00000000   RAM:013 00000000
RAM:014 00000000   RAM:015 00000000   RAM:016 00000000   RAM:017 00000000
RAM:018 00000000   RAM:019 00000000   RAM:01A 00000000   RAM:01B 00000000
RAM:01C 00000000   RAM:01D 00000000   RAM:01E 00000000   RAM:01F 00000000
RAM:020 00000000   RAM:021 00000000   RAM:022 00000000   RAM:023 00000000
RAM:024 00000000   RAM:025 00000000   RAM:026 00000000   RAM:027 00000000
```

The start of **S_BACK** state. RAM automaton runs to get the required data. The process will repeat 3 times so we omitted some graphs.



```
        Zhejiang University Computer Organization Experimental
                  SOC Test Environment (With RISC-V)
x0:zero 00000000   x01: ra ABCDE71C   x02: sp FFFFF0F0   x03: gp F0F0F0F0
x04: tp 000000F0   x05: t0 0000F0F0   x06: t1 00000000   x07: t2 00000000
x8:fps0 00000000   x09: s1 00000000   x10: a0 00000000   x11: a1 00000000
x12: a2 00000000   x13: a3 00000000   x14: a4 00000000   x15: a5 00000000
x16: a6 00000000   x17: a7 00000000   x18: s2 00000000   x19: s3 00000000
x20: s4 00000000   x21: s5 00000000   x22: s6 00000000   x23: s7 00000000
x24: s8 00000000   x25: s9 00000000   x26:s10 00000000   x27:s11 00000000
x28: t3 00000000   x29: t4 00000000   x30: t5 00000000   x31: t6 00000000
PC---IF 00000040   INST-IF FFDFF06F   rs1Data 00000000   rs2Data 00000000
PC---ID 0000003C   INST-ID 0ED06813   rs1Addr 00000000   rs2Addr 0000000D
PC--EXE 00000038   INST-EX 41002403   CMU-RAM 00030001   PCJumpA 00000129
PC--MEM 00000034   INST--M 40002383   B/PCE-S 00000000   D/C-Hzd 00000000
PC---WB 00000030   INST-WB 20002303   I/ABSel 00010001   PCIFNxt 00000044
ALU-Ain 00000000   ALU-Out 00000400   CPUAddr 00000000   ALUCtrl 00000004
ALU-Bin 00000410   WB-Data 00000000   CPU-DAi 00008000   WR--MIO 00000001
Imm32ID 000000ED   WB-Addr 00000006   CPU-DAo 00000000   RegW/DR 00010001
RAM:000 00008000   RAM:001 00000000   RAM:002 00000000   RAM:003 00000000
RAM:004 00000000   RAM:005 00000000   RAM:006 00000000   RAM:007 0000F0F0
RAM:008 00000000   RAM:009 00000000   RAM:00A 00000000   RAM:00B 00000000
RAM:00C 00000000   RAM:00D 00000000   RAM:00E 00000000   RAM:00F 00000000
RAM:010 00000000   RAM:011 00000000   RAM:012 00000000   RAM:013 00000000
RAM:014 00000000   RAM:015 00000000   RAM:016 00000000   RAM:017 00000000
RAM:018 00000000   RAM:019 00000000   RAM:01A 00000000   RAM:01B 00000000
RAM:01C 00000000   RAM:01D 00000000   RAM:01E 00000000   RAM:01F 00000000
RAM:020 00000000   RAM:021 00000000   RAM:022 00000000   RAM:023 00000000
RAM:024 00000000   RAM:025 00000000   RAM:026 00000000   RAM:027 00000000
```

The start of **S_FILL**, similar to above.



```
            Zhejiang University Computer Organization Experimental
                       SOC Test Environment (With RISC-V)
x0:zero 00000000   x01: ra ABCDE71C   x02: sp FFFFF0F0   x03: gp F0F0F0F0
x04: tp 000000F0   x05: t0 0000F0F0   x06: t1 00000000   x07: t2 00000000
x8:fps0 00000000   x09: s1 00000000   x10: a0 00000000   x11: a1 00000000
x12: a2 00000000   x13: a3 00000000   x14: a4 00000000   x15: a5 00000000
x16: a6 00000000   x17: a7 00000000   x18: s2 00000000   x19: s3 00000000
x20: s4 00000000   x21: s5 00000000   x22: s6 00000000   x23: s7 00000000
x24: s8 00000000   x25: s9 00000000   x26:s10 00000000   x27:s11 00000000
x28: t3 00000000   x29: t4 00000000   x30: t5 00000000   x31: t6 00000000
PC---IF 00000040   INST-IF FFDFF06F   rs1Data 00000000   rs2Data 00000000
PC---ID 0000003C   INST-ID 0ED06813   rs1Addr 00000000   rs2Addr 0000000D
PC--EXE 00000038   INST-EX 41002403   CMU-RAM 00040000   PCJumpA 00000129
PC--MEM 00000034   INST--M 40002383   B/PCE-S 00000100   D/C-Hzd 00000000
PC---WB 00000030   INST-WB 20002303   I/ABSel 00010001   PCIFNxt 00000044
ALU-Ain 00000000   ALU-Out 00000400   CPUAddr 00000000   ALUCtrl 00000004
ALU-Bin 00000410   WB-Data 00000000   CPU-DAi 00000000   WR--MIO 00000001
Imm32ID 000000ED   WB-Addr 00000006   CPU-DAo 00000000   RegW/DR 00010001
RAM:000 00000000   RAM:001 00000000   RAM:002 00000000   RAM:003 00000000
RAM:004 00000000   RAM:005 00000000   RAM:006 00000000   RAM:007 0000F0F0
RAM:008 00000000   RAM:009 00000000   RAM:00A 00000000   RAM:00B 00000000
RAM:00C 00000000   RAM:00D 00000000   RAM:00E 00000000   RAM:00F 00000000
RAM:010 00000000   RAM:011 00000000   RAM:012 00000000   RAM:013 00000000
RAM:014 00000000   RAM:015 00000000   RAM:016 00000000   RAM:017 00000000
RAM:018 00000000   RAM:019 00000000   RAM:01A 00000000   RAM:01B 00000000
RAM:01C 00000000   [illegible]D 00000000   RAM:01E 00000000   RAM:01F 00000000
RAM:020 00000000   [illegible]1 00000000   RAM:022 00000000   RAM:023 00000000
RAM:024 00000000   [illegible] 00000000   RAM:026 00000000   RAM:027 00000000
```

S_BACK.

5 Discussion and Conclusion

5.1 Problems

5.1.1 Problems concerning the Experiment Guides

Problem 1. Program Difference

The program used in the experiment guilds are not the similar as given in the project and waveform simulation. That original program can differ **lw**, **lb**, … operations, but the given program cannot.

We have modified the given program to overcome this difference, so some of our data may not be similar to other groups.

5.2 Achievements and conclusion

In this experiment, I implemented a cache controller based on the cache we wrote in lab-3, and learned the knowledge about CPU-Memory communication. Overall the experiment is successful.