Organization of Digital Computer LAB

EECS 112L

Lab2: Mips Single Cycle Processor

Student Name:TeHsun Wang

Student ID: 24944958

Date: Due 10/27

1 Objective

The objective of this lab report is to design, implement, and analyze the operation of a MIPS single-cycle processor. In this report, we have to provide a comprehensive understanding of the processor's architecture, and finish control unit, ALU, data path, and the execution of various MIPS instructions in a single clock cycle. Additionally, we are going to use the testbench to run the instructions and observe its performance.

2 Procedure

1. We have to complete the ALU.v by implementing new instructions based on AluControl.
2. We have to complete Control.v based on the instruction[31:26]. In each opcode block, we have to add jump, branch bit into it.
3. We have to complete ALUControl.v based on the instruction[5:0] (function) and ALUOp

Name	format	Instruction[31:26]	Instruction[5:0]	ALUop	alu_control
Add	R	000000	100000	10	0010
Addi	I	001000	-	00	0010
and	R	000000	100100	10	0000
andi	I	001100	-	11	0000
Beq	I	000100	-	01	0110
Lw	I	100011	-	00	0010
Nor	R	000000	100111	10	1100
Or	R	000000	100101	10	0001
Slt	R	000000	101010	10	0111
Sll	R	110000	000000	10	1000
Srl	R	110000	000010	10	1001
sra	R	110000	000011	10	1010
Sw	I	101011	-	00	0010
Sub	R	000000	100010	10	0110
xor	R	000000	100110	10	0100
Mult	R	000000	011000	10	0101
div	R	000000	011010	10	1011
jump	J	000010	-	-	-

table 1

4. Then we need to connect additional two muxes in our datapath. To do that we have to add some wires and use assign to define.

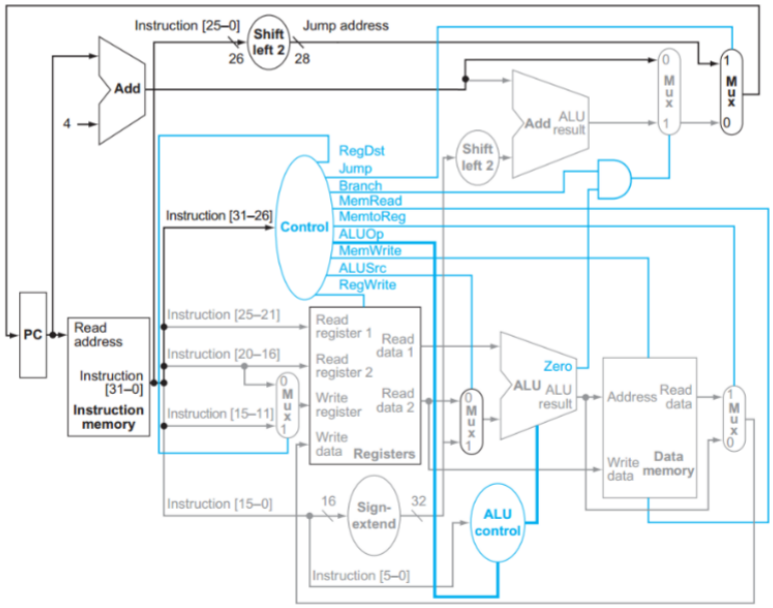


Figure 1

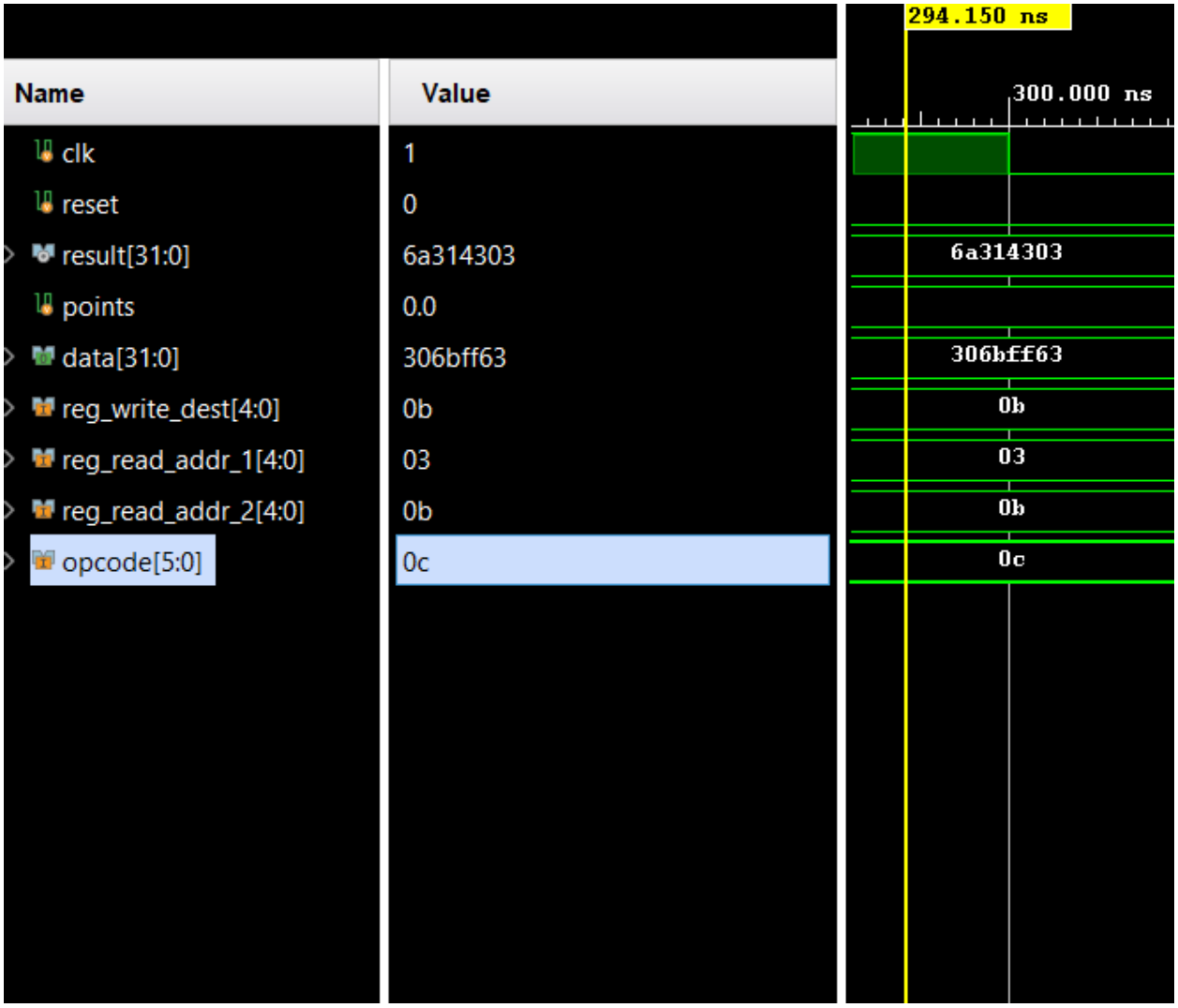
3 Simulation Results

To test the result, we have to first find the binary code of that instruction and add it to the instruction memory. Even though we have all the information in data. I still added opcode, reg_write, reg_read_addr1 and 2 to find which instruction we are doing easier and also we can read rd,rs, and rt at the same time. Next, I went to find the instruction in the instruction memory and confirmed each field depending on the instruction format. Finally, we can observe the result easily in the waveform. The comments are the instructions we want, the result in hex.

In the followings are the 11 new instructions added after implementation:

1. andi

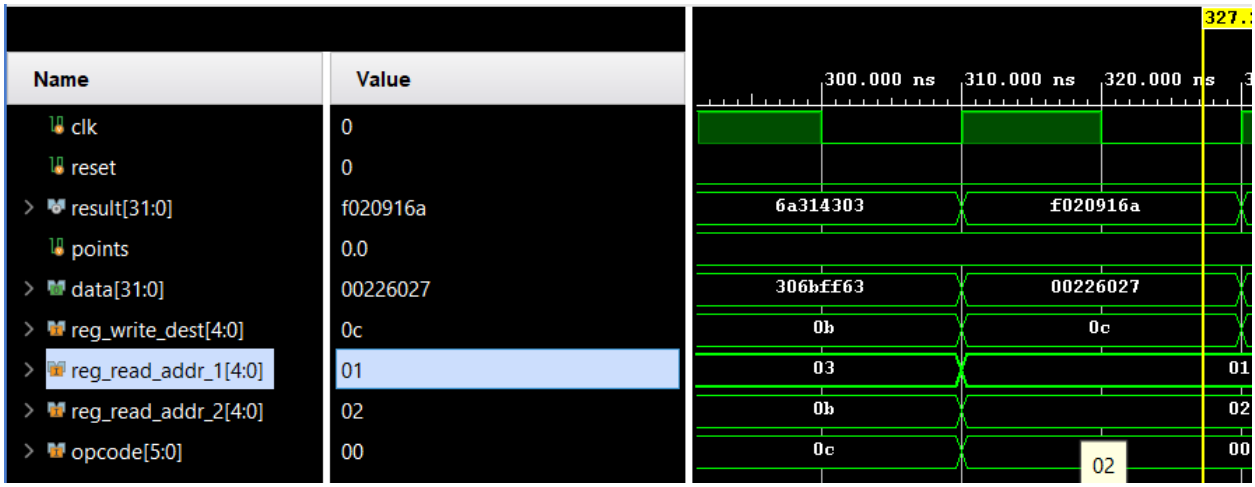
```
rom[10] = 32'b00110000011010111111111101100011; // andi r11,r3,#ff63        6a314303          r11= 6a314303
```



r3 + imm(#ff63) = r11, which equals the result.

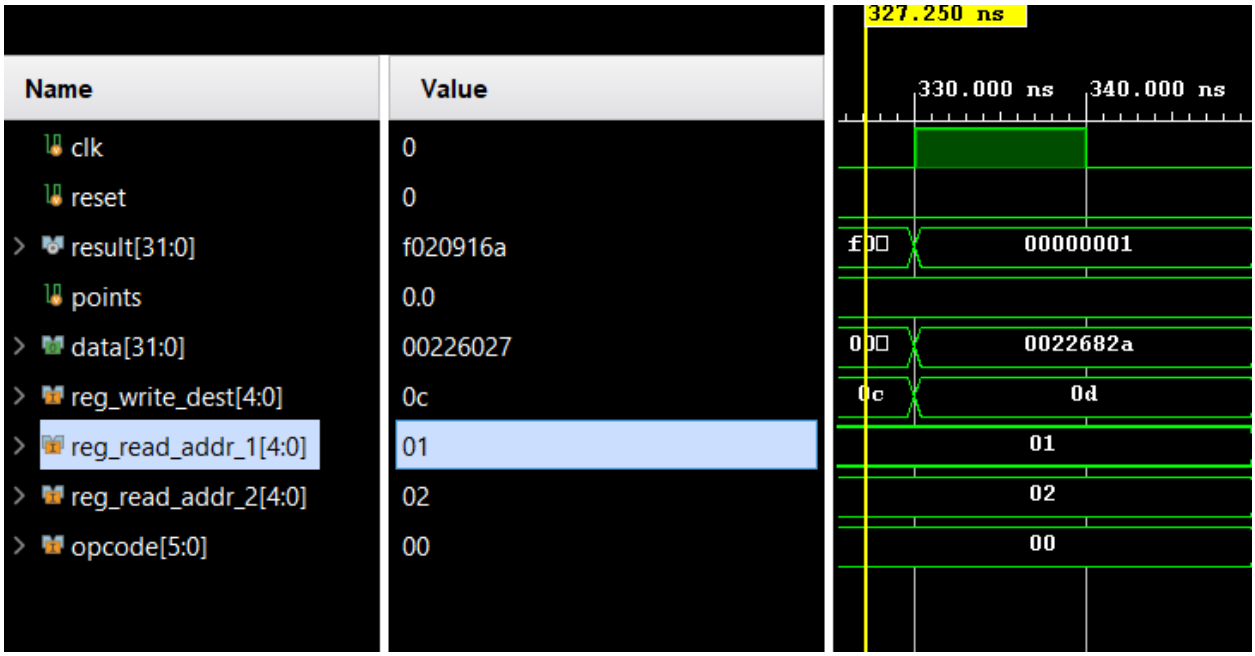
2. nor

```
rom[11] = 32'b00000000001000100110000000100111; // nor  r12,r1,r2          f020916a          r12= f020916a
```



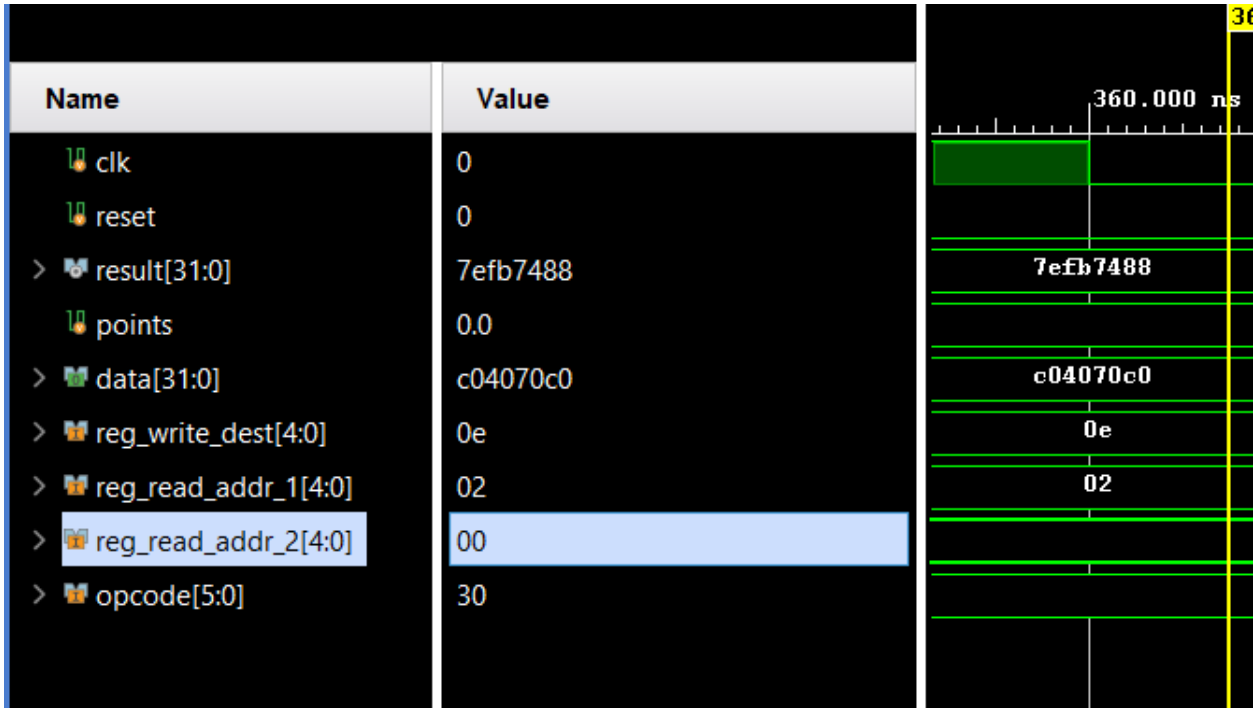
3. slt

```
rom[12] = 32'b00000000001000100110100000101010; // slt  r13,r1,r2          00000001          r13= 00000001
```



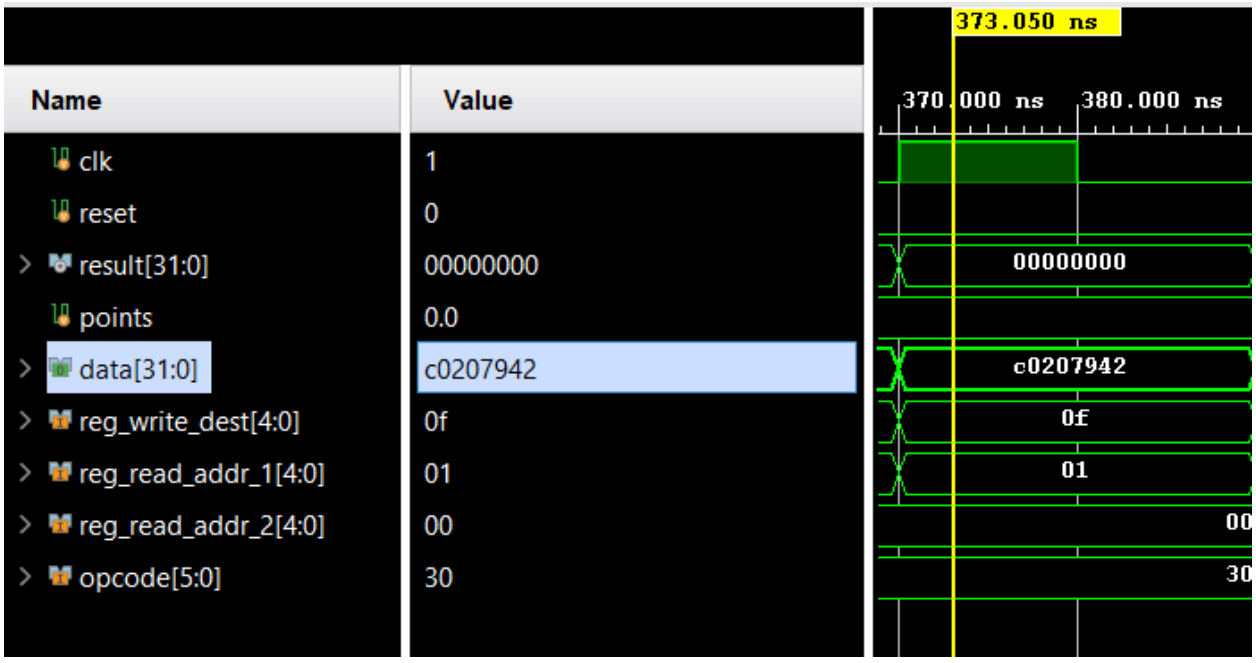
4. sll

```
rom[13] = 32'b11000000010000000111000011000000; // sll  r14,r2,#3          7efb7488          r14= 7efb7488
```



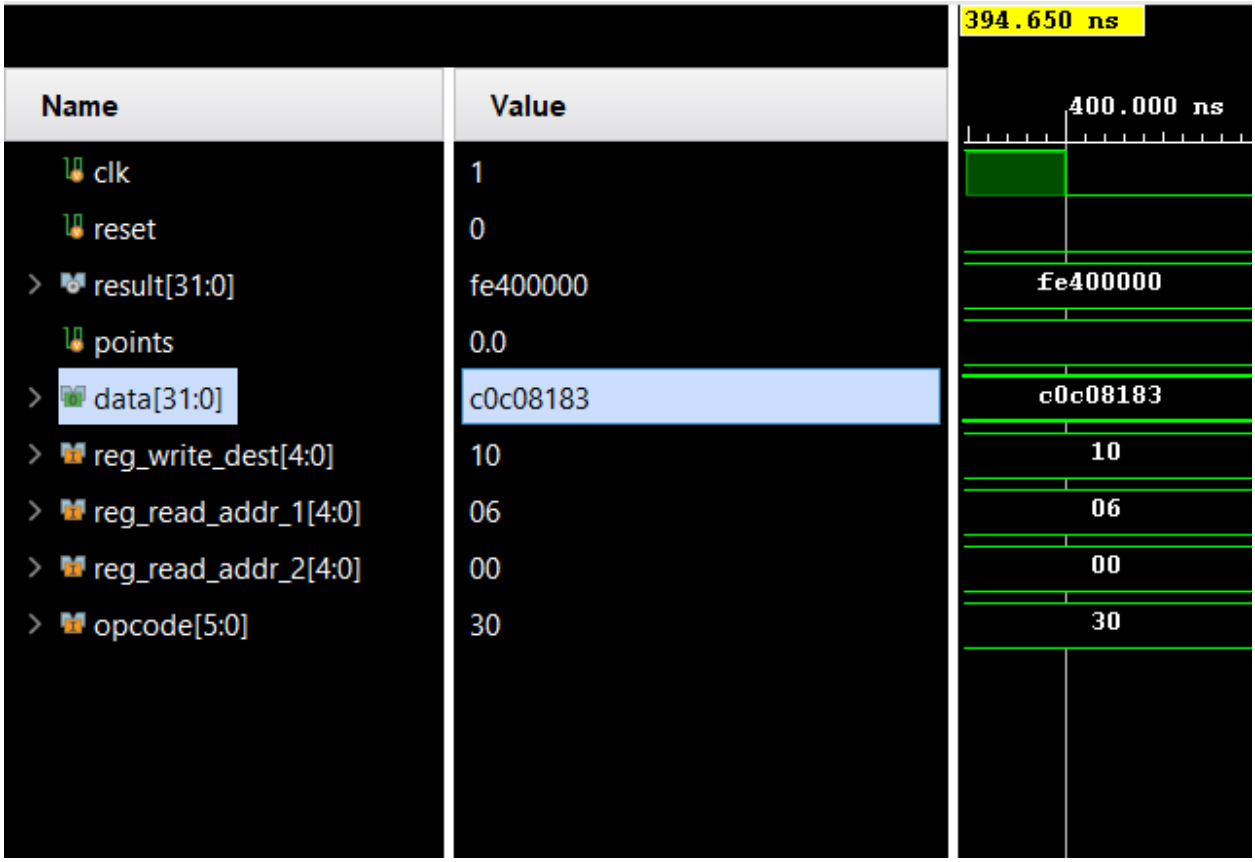
5. slr

```
rom[14] = 32'b11000000001000000111100101000010; // srl  r15,r1,#5          00000000          r15= 00000000
```



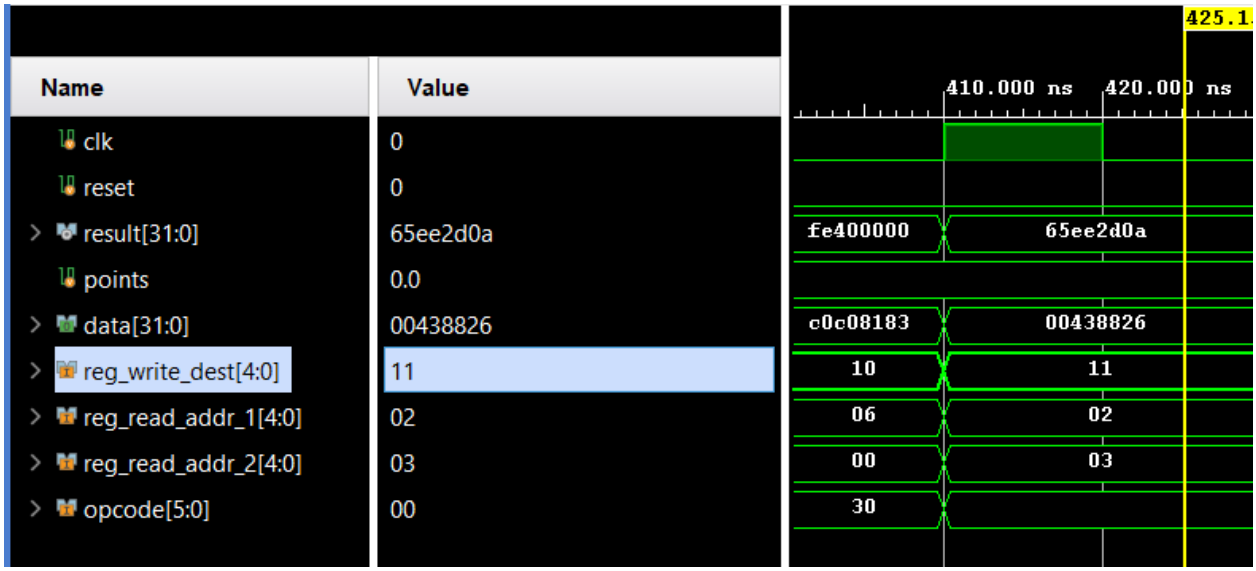
6. sra

```
rom[15] = 32'b11000000110000001000000110000011; // sra  r16,r6,#6          fe400000          r16= fe400000
```



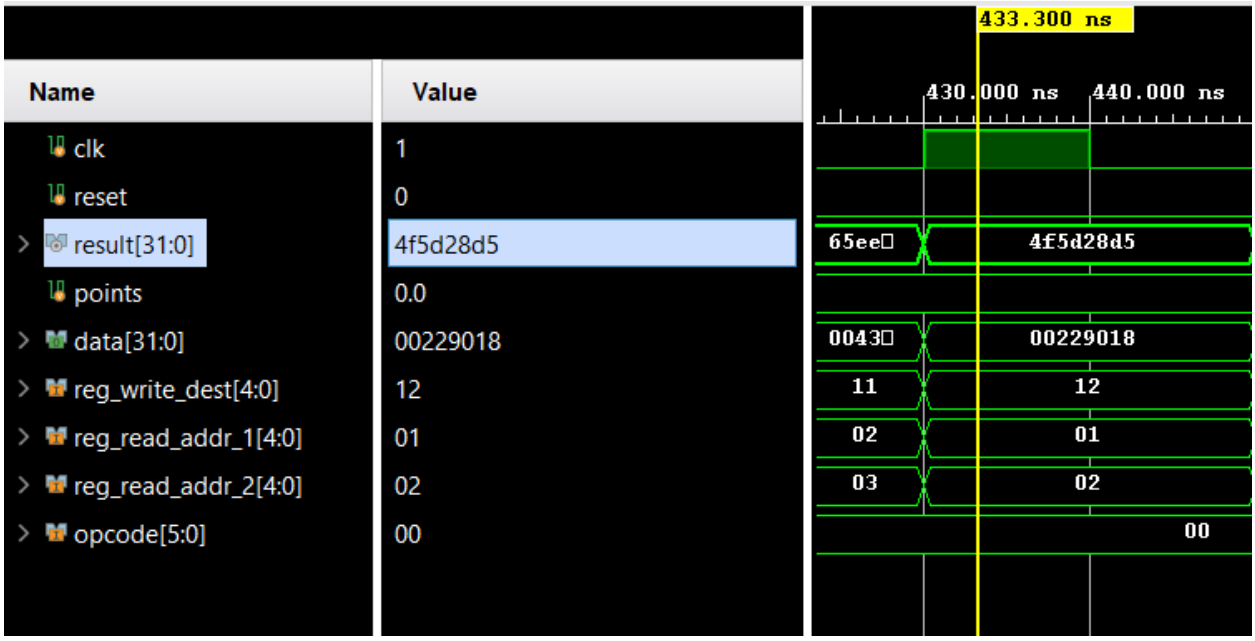
7. xor

```
rom[16] = 32'b00000000010000111000100000100110; // xor  r17,r2,r3          65ee2d0a          r17= 65ee2d0a
```



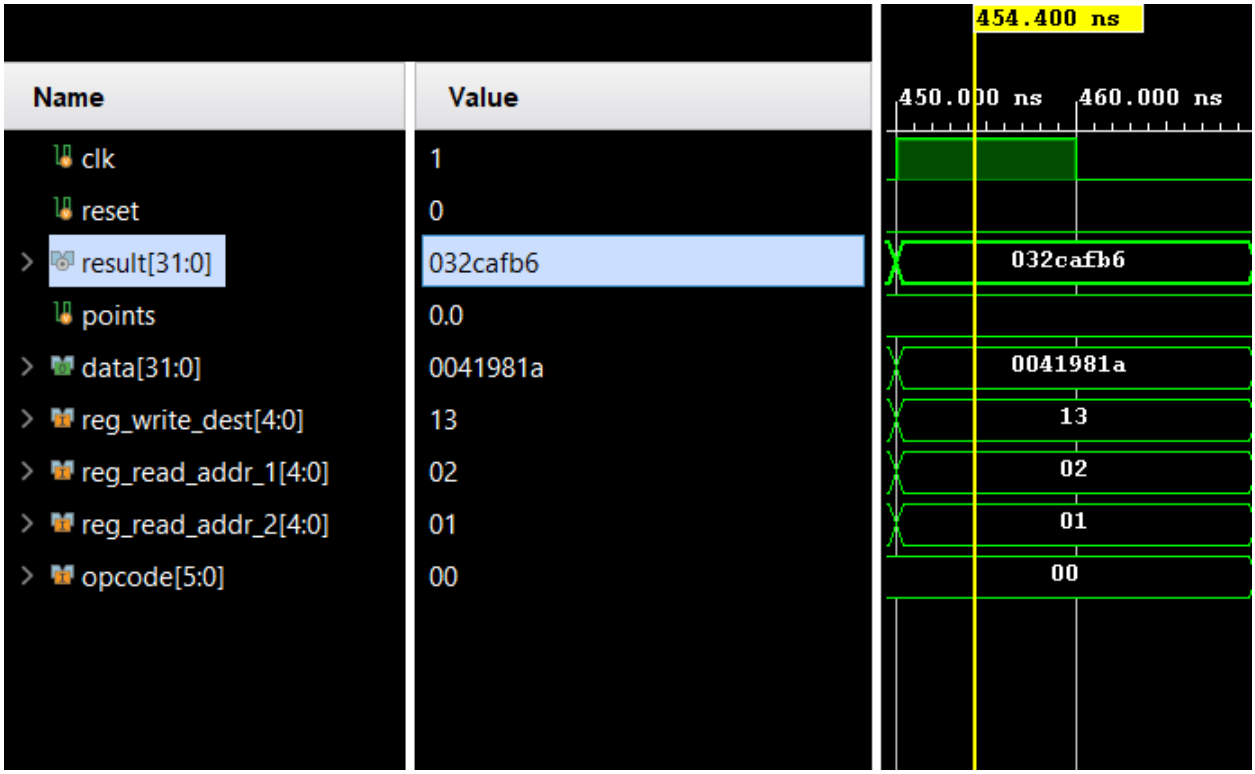
8. mult

```
rom[17] = 32'b00000000001000101001000000011000; // mult r17,r1,r2          4f5d28d5          r18= 4f5d28d5
```



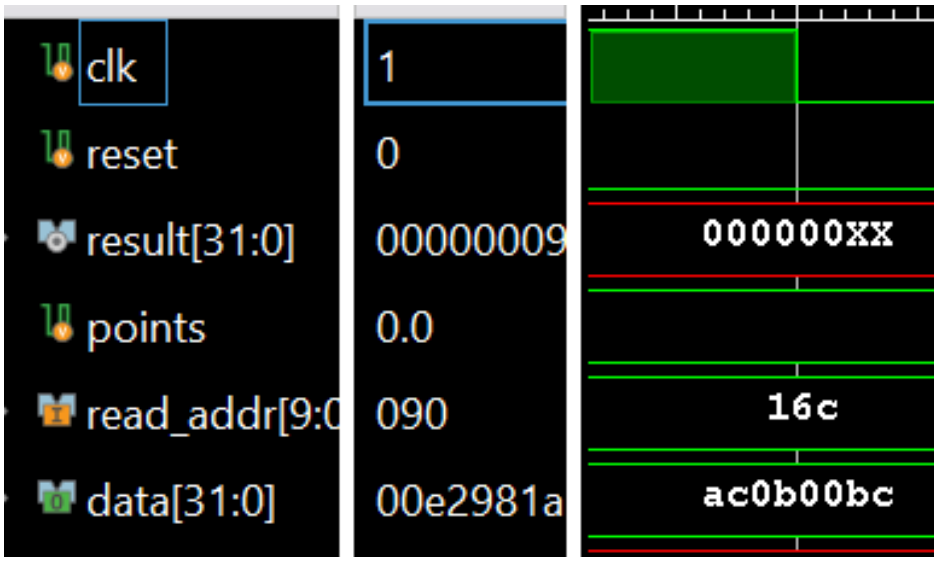
9. div

```
rom[18] = 32'b00000000010000011001100000011010; // div  r19,r2,r1          032cafb6          r19= 032cafb6
```



10. beq

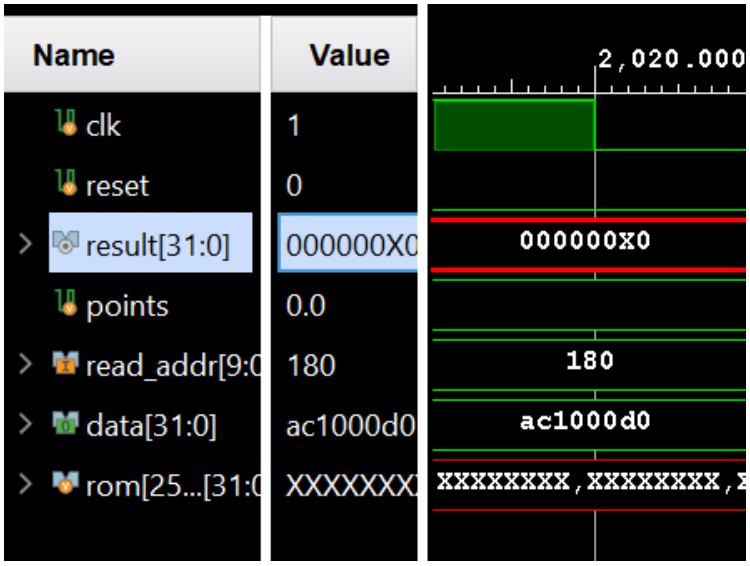
```
rom[55] = 32'b10101100000010110000000001110100; // sw mem[r0+29] <= r11      74(add 29)              -              mem[29]= d18fa000
```



The value of the register is empty, which is also mentioned in the instruction memory. This means that the operation branched correctly.

11. jump

```
rom[55] = 32'b10101100000010110000000001110100; // sw mem[r0+29] <= r11      74(add 29)              -           mem[29]= d18fa000
```



The value in the register is empty, which is also mentioned in the instruction memory. This means that the operation jumped correctly.

This is the result after I run 4000 in the Tcl console