CA2 -SINGLE CYCLE PROCESSOR

Zahra Hojati 810199403 Fatemeh Mohammadi 810199489

Controller: signals:

OPC		Reg_dst	Write_dst	ALUsrc	ALUOP	Mem_read	Mem_write	Mem_to_reg	branch	J	Reg_write
000000	RT	01	0	1	10	0	0	0	0	00	1
100011	lw	00	0	0	00	1	0	1	0	00	1
101011	sw	x	x	1	00	0	1	x	0	00	0
000100	beq	x	x	0	01	0	0	x	1	00	0
001001	addi	00	0	1	00	0	0	0	0	00	1
000010	J	x	x	x	x	0	0	x	x	01	0
000110	Jr	x	x	x	x	0	0	x	x	10	0
000011	Jal	10	1	x	x	0	0	x	x	01	1
001010	slti	00	0	1	11	0	0	0	0	00	1

31 26	25 21	20 16	15 11	10 6	5 0
000000(RT)	**Rs**	**Rt**	**Rd**	**shift**	**func**

31 26	25 21	20 16	15 0
001001(addi)	**Rs**	**Rt**	**data**

31 26	25 21	20 16	15 0
001010(slti)	**Rs**	**Rt**	**data**

31 26	25 21	20 16	15 0
100011(lw)	**Rs**	**Rt**	**address**

31 26	25 21	20 16	15 0
101011(sw)	**Rs**	**Rt**	**address**

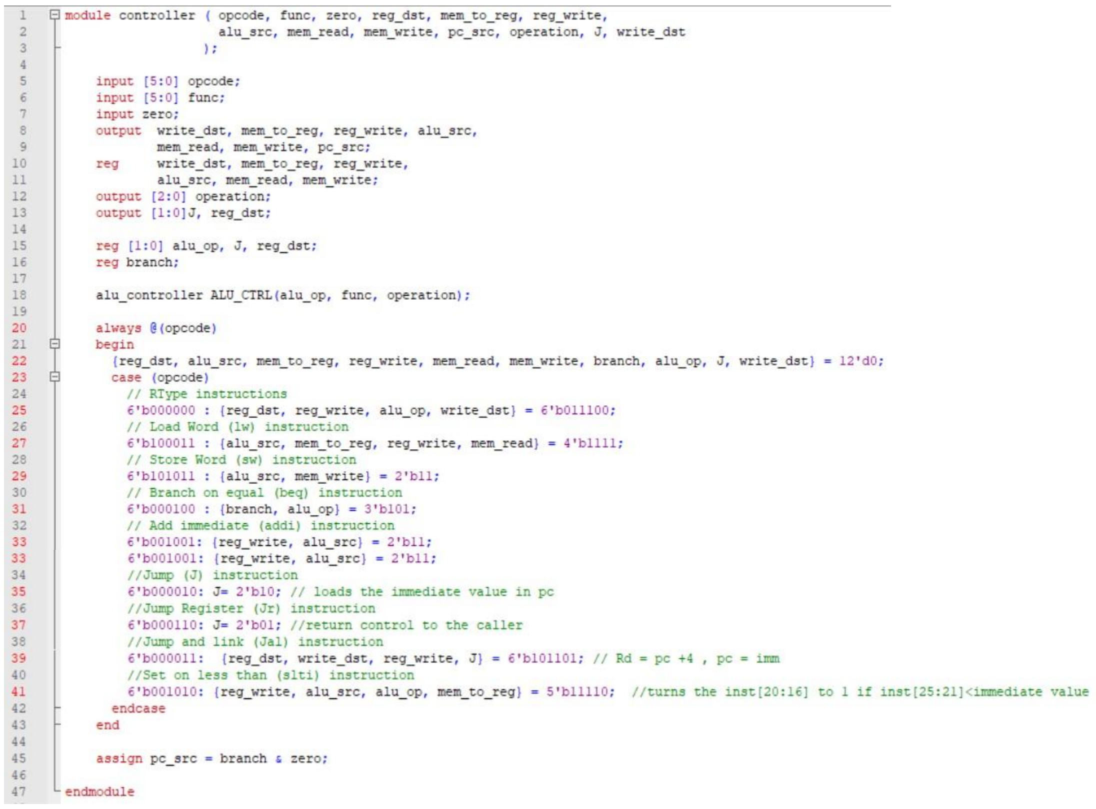
31 26	25 21	20 16	15 0
000100(beq)	**Rs**	**Rt**	**address**

31 26	25 21 20 16 15 11 10 6 5 0
000010(J)	**address**

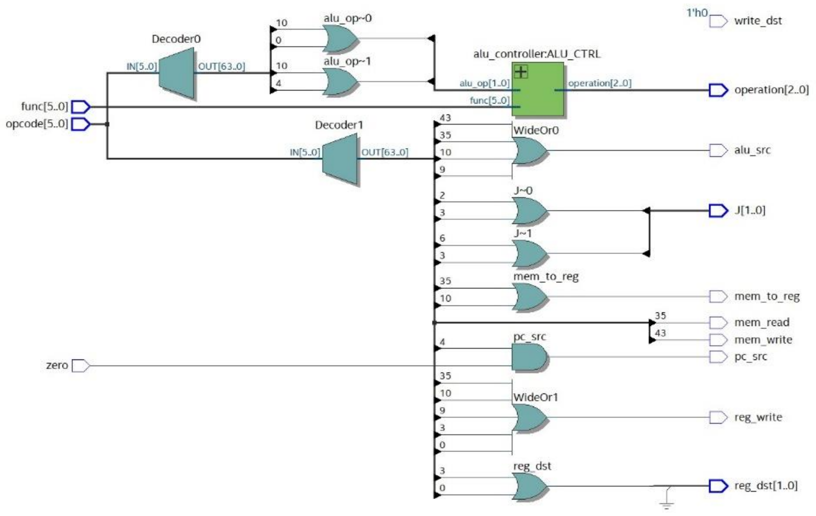
31 26	25 21	20 16 15 11 10 6 5 0
000110(Jr)	**Rs**	**Unimportant**

31 26	25 21 20 16 15 11 10 6 5 0
000011(Jal)	**address**

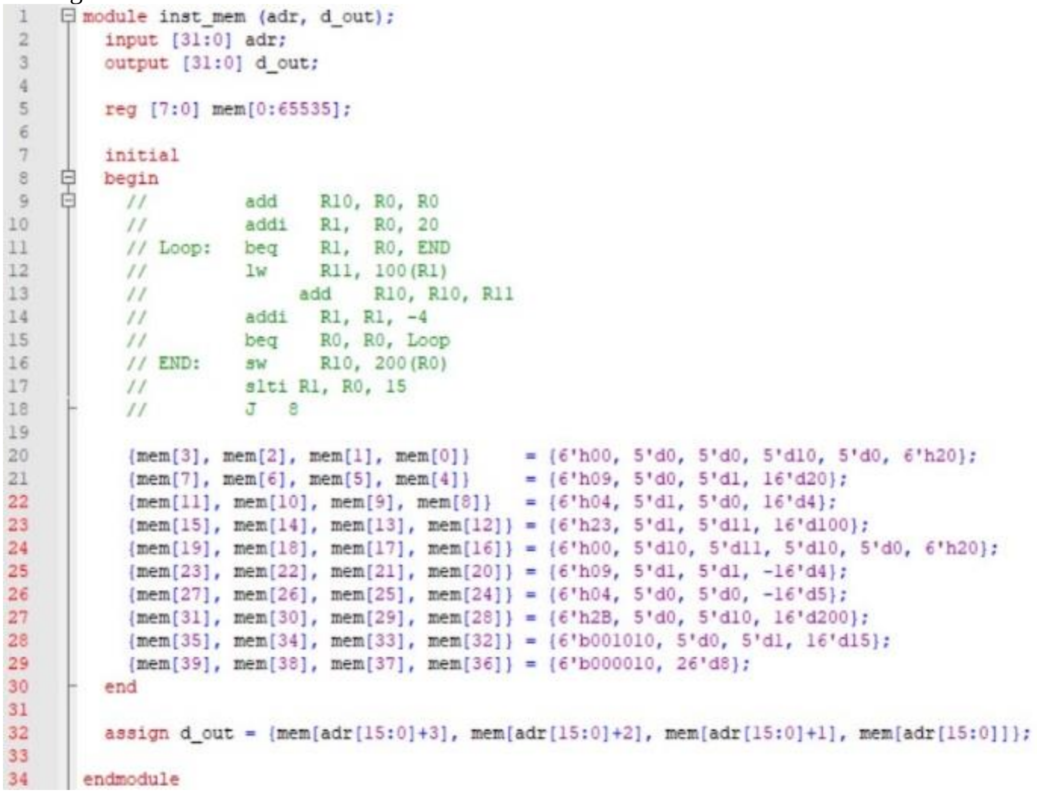
Controller: Code and RTL overview:



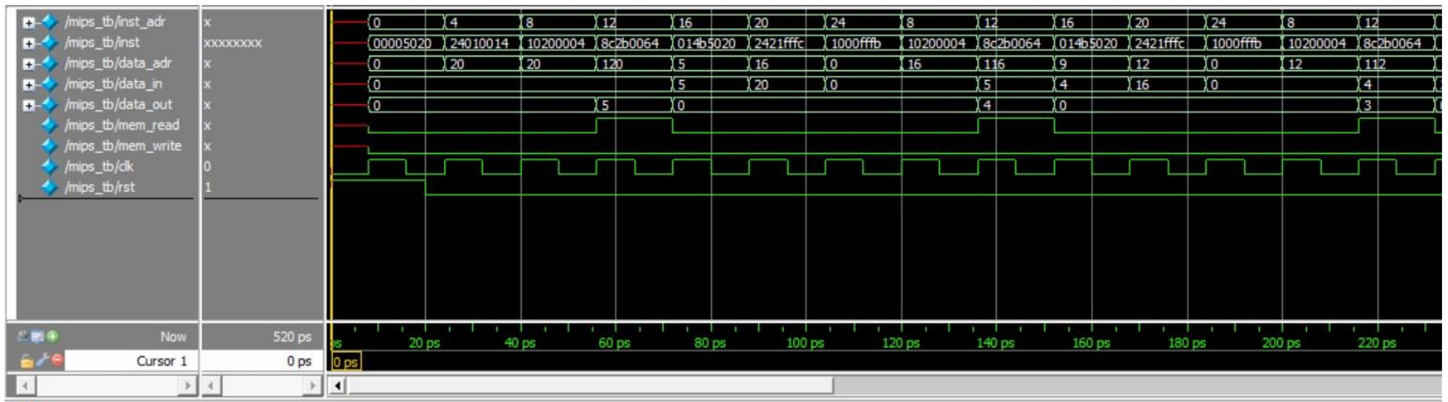
```verilog
module controller ( opcode, func, zero, reg_dst, mem_to_reg, reg_write,
                    alu_src, mem_read, mem_write, pc_src, operation, J, write_dst
                  );

    input [5:0] opcode;
    input [5:0] func;
    input zero;
    output  write_dst, mem_to_reg, reg_write, alu_src,
            mem_read, mem_write, pc_src;
    reg     write_dst, mem_to_reg, reg_write,
            alu_src, mem_read, mem_write;
    output [2:0] operation;
    output [1:0]J, reg_dst;

    reg [1:0] alu_op, J, reg_dst;
    reg branch;

    alu_controller ALU_CTRL(alu_op, func, operation);

    always @(opcode)
    begin
      {reg_dst, alu_src, mem_to_reg, reg_write, mem_read, mem_write, branch, alu_op, J, write_dst} = 12'd0;
      case (opcode)
        // RType instructions
        6'b000000 : {reg_dst, reg_write, alu_op, write_dst} = 6'b011100;
        // Load Word (lw) instruction
        6'b100011 : {alu_src, mem_to_reg, reg_write, mem_read} = 4'b1111;
        // Store Word (sw) instruction
        6'b101011 : {alu_src, mem_write} = 2'b11;
        // Branch on equal (beq) instruction
        6'b000100 : {branch, alu_op} = 3'b101;
        // Add immediate (addi) instruction
        6'b001001: {reg_write, alu_src} = 2'b11;
        6'b001001: {reg_write, alu_src} = 2'b11;
        //Jump (J) instruction
        6'b000010: J= 2'b10; // loads the immediate value in pc
        //Jump Register (Jr) instruction
        6'b000110: J= 2'b01; //return control to the caller
        //Jump and link (Jal) instruction
        6'b000011:  {reg_dst, write_dst, reg_write, J} = 6'b101101; // Rd = pc +4 , pc = imm
        //Set on less than (slti) instruction
        6'b001010: {reg_write, alu_src, alu_op, mem_to_reg} = 5'b11110;  //turns the inst[20:16] to 1 if inst[25:21]<immediate value
      endcase
    end

    assign pc_src = branch & zero;

endmodule
```

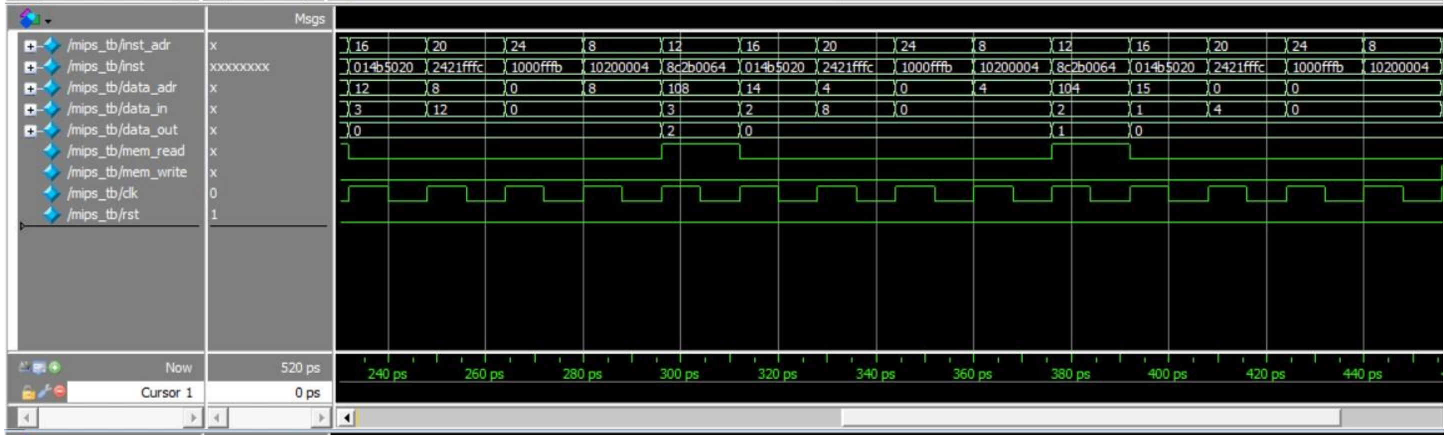


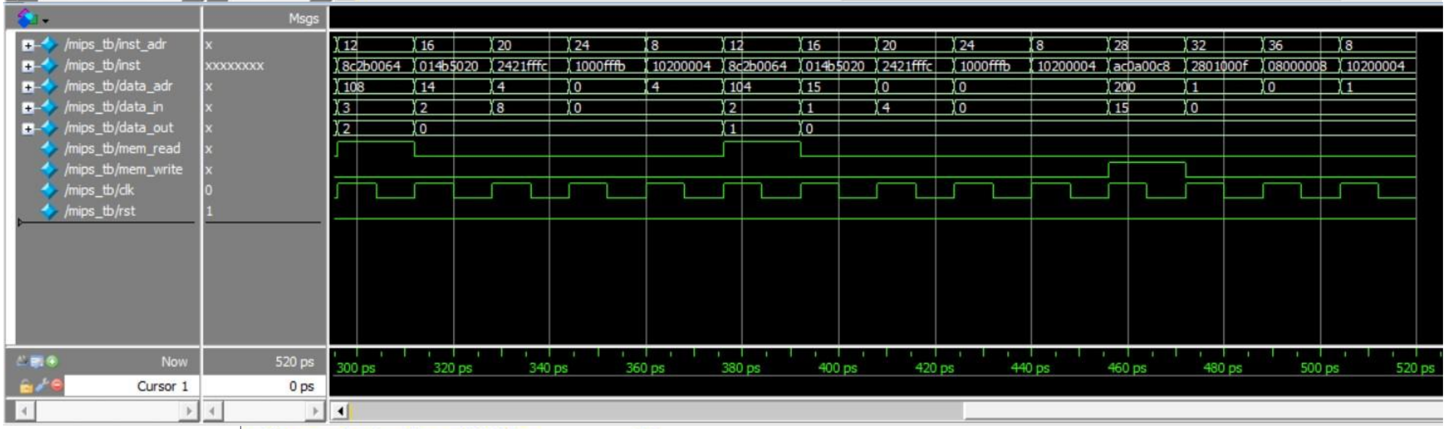
Testing the circuit via the instruction memory module:
Testing slti and J:

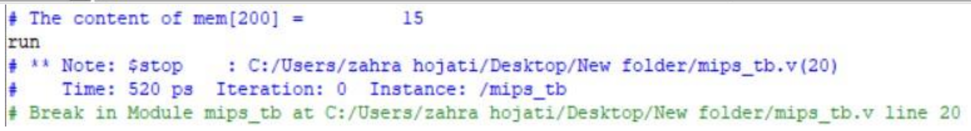

```verilog
module inst_mem (adr, d_out);
  input [31:0] adr;
  output [31:0] d_out;

  reg [7:0] mem[0:65535];

  initial
  begin
    //          add    R10, R0, R0
    //          addi   R1,  R0, 20
    // Loop:    beq    R1,  R0, END
    //          lw     R11, 100(R1)
    //                 add    R10, R10, R11
    //          addi   R1, R1, -4
    //          beq    R0, R0, Loop
    // END:     sw     R10, 200(R0)
    //          slti R1, R0, 15
    //          J   8

    {mem[3], mem[2], mem[1], mem[0]}     = {6'h00, 5'd0, 5'd0, 5'd10, 5'd0, 6'h20};
    {mem[7], mem[6], mem[5], mem[4]}     = {6'h09, 5'd0, 5'd1, 16'd20};
    {mem[11], mem[10], mem[9], mem[8]}   = {6'h04, 5'd1, 5'd0, 16'd4};
    {mem[15], mem[14], mem[13], mem[12]} = {6'h23, 5'd1, 5'd11, 16'd100};
    {mem[19], mem[18], mem[17], mem[16]} = {6'h00, 5'd10, 5'd11, 5'd10, 5'd0, 6'h20};
    {mem[23], mem[22], mem[21], mem[20]} = {6'h09, 5'd1, 5'd1, -16'd4};
    {mem[27], mem[26], mem[25], mem[24]} = {6'h04, 5'd0, 5'd0, -16'd5};
    {mem[31], mem[30], mem[29], mem[28]} = {6'h2B, 5'd0, 5'd10, 16'd200};
    {mem[35], mem[34], mem[33], mem[32]} = {6'b001010, 5'd0, 5'd1, 16'd15};
    {mem[39], mem[38], mem[37], mem[36]} = {6'b000010, 26'd8};
  end

  assign d_out = {mem[adr[15:0]+3], mem[adr[15:0]+2], mem[adr[15:0]+1], mem[adr[15:0]]};

endmodule
```

Waveform: as seen in the image below instruction 32, compares the 2 register and data_in which is the value of alu_out that turns to 1 after confirming the accuracy of the condition. Then right after that, it’ll jump to referred instruction.



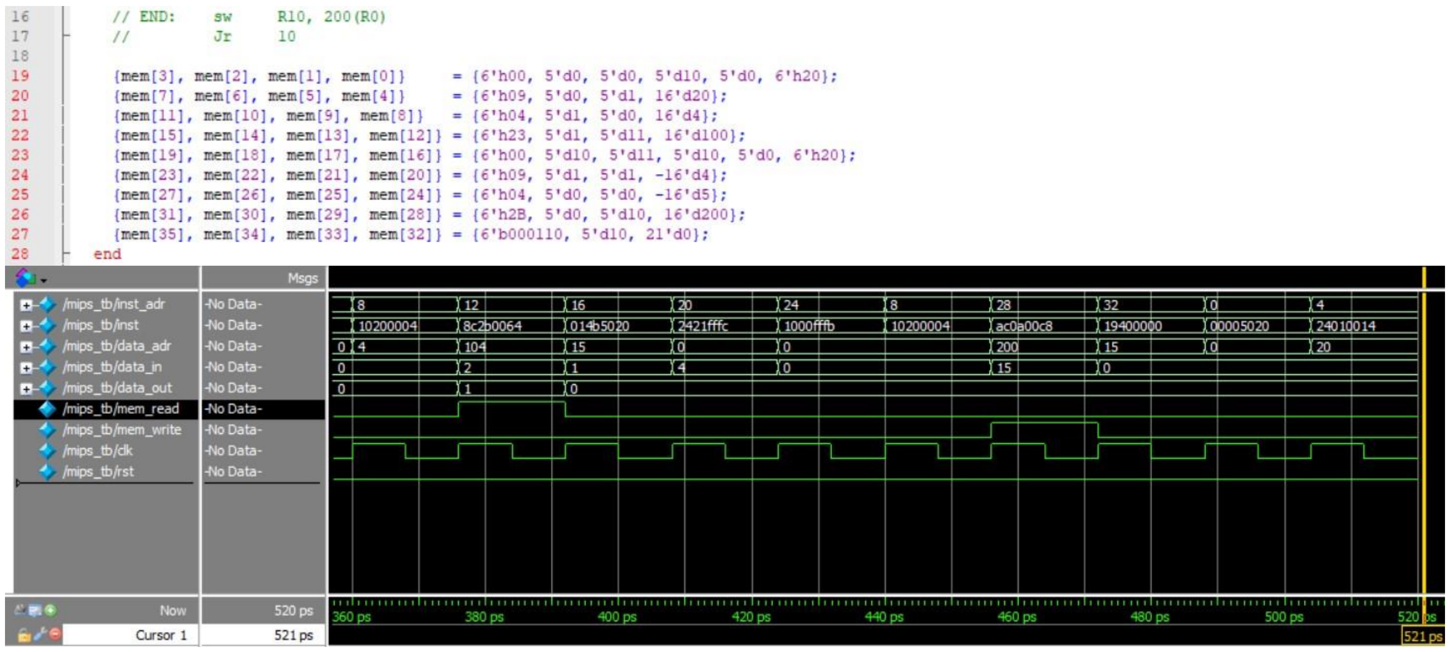





```
# The content of mem[200] =          15
run
# ** Note: $stop    : C:/Users/zahra hojati/Desktop/New folder/mips_tb.v(20)
#    Time: 520 ps  Iteration: 0  Instance: /mips_tb
# Break in Module mips_tb at C:/Users/zahra hojati/Desktop/New folder/mips_tb.v line 20
```

Testing Jr:

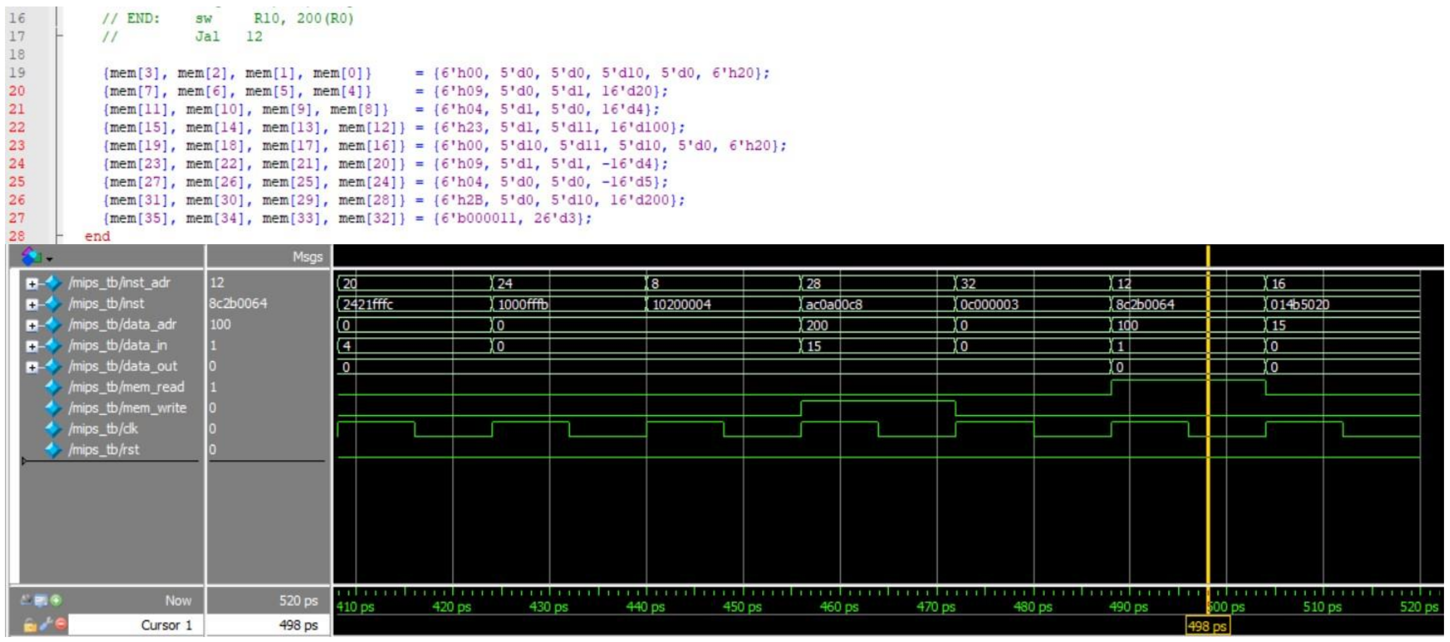
Data upload in R10 was 0, therefore by referring to the 10th register, it'll go to the instruction with the register's value.



```
    // END:    sw     R10, 200(R0)
    //         Jr     10

    {mem[3], mem[2], mem[1], mem[0]}     = {6'h00, 5'd0, 5'd0, 5'd10, 5'd0, 6'h20};
    {mem[7], mem[6], mem[5], mem[4]}     = {6'h09, 5'd0, 5'd1, 16'd20};
    {mem[11], mem[10], mem[9], mem[8]}   = {6'h04, 5'd1, 5'd0, 16'd4};
    {mem[15], mem[14], mem[13], mem[12]} = {6'h23, 5'd1, 5'd11, 16'd100};
    {mem[19], mem[18], mem[17], mem[16]} = {6'h00, 5'd10, 5'd11, 5'd10, 5'd0, 6'h20};
    {mem[23], mem[22], mem[21], mem[20]} = {6'h09, 5'd1, 5'd1, -16'd4};
    {mem[27], mem[26], mem[25], mem[24]} = {6'h04, 5'd0, 5'd0, -16'd5};
    {mem[31], mem[30], mem[29], mem[28]} = {6'h2B, 5'd0, 5'd10, 16'd200};
    {mem[35], mem[34], mem[33], mem[32]} = {6'b000110, 5'd10, 21'd0};
  end
```

Msgs
/mips_tb/inst_adr -No Data- 8 12 16 20 24 8 28 32 0 4
/mips_tb/inst -No Data- 10200004 8c2b0064 014b5020 2421fffc 1000fffb 10200004 ac0a00c8 19400000 00005020 24010014
/mips_tb/data_adr -No Data- 0 4 104 15 0 0 200 15 0 20
/mips_tb/data_in -No Data- 0 2 1 4 0 15 0
/mips_tb/data_out -No Data- 0 1 0
/mips_tb/mem_read -No Data-
/mips_tb/mem_write -No Data-
/mips_tb/clk -No Data-
/mips_tb/rst -No Data-
Now 520 ps
Cursor 1 521 ps
360 ps 380 ps 400 ps 420 ps 440 ps 460 ps 480 ps 500 ps 520 ps
521 ps

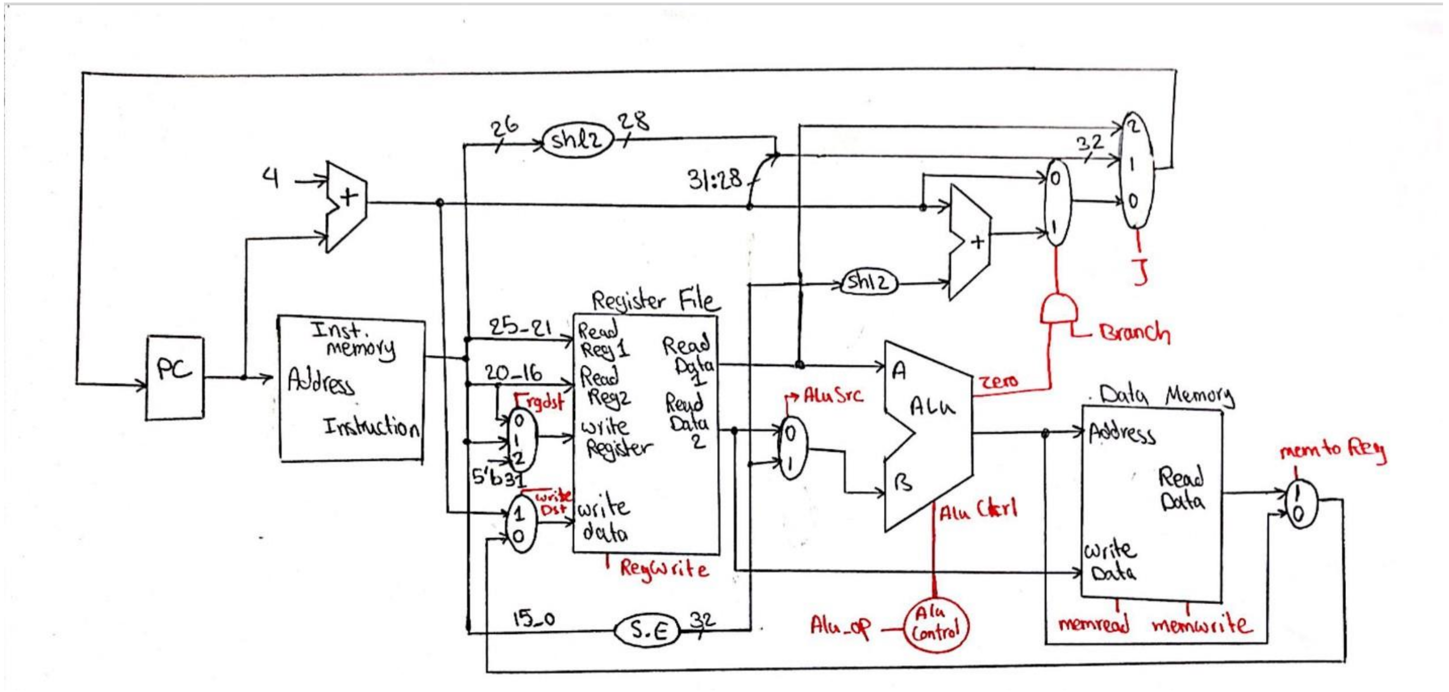
Testing Jal:



```
    // END:    sw     R10, 200(R0)
    //         Jal    12

    {mem[3], mem[2], mem[1], mem[0]}     = {6'h00, 5'd0, 5'd0, 5'd10, 5'd0, 6'h20};
    {mem[7], mem[6], mem[5], mem[4]}     = {6'h09, 5'd0, 5'd1, 16'd20};
    {mem[11], mem[10], mem[9], mem[8]}   = {6'h04, 5'd1, 5'd0, 16'd4};
    {mem[15], mem[14], mem[13], mem[12]} = {6'h23, 5'd1, 5'd11, 16'd100};
    {mem[19], mem[18], mem[17], mem[16]} = {6'h00, 5'd10, 5'd11, 5'd10, 5'd0, 6'h20};
    {mem[23], mem[22], mem[21], mem[20]} = {6'h09, 5'd1, 5'd1, -16'd4};
    {mem[27], mem[26], mem[25], mem[24]} = {6'h04, 5'd0, 5'd0, -16'd5};
    {mem[31], mem[30], mem[29], mem[28]} = {6'h2B, 5'd0, 5'd10, 16'd200};
    {mem[35], mem[34], mem[33], mem[32]} = {6'b000011, 26'd3};
  end
```

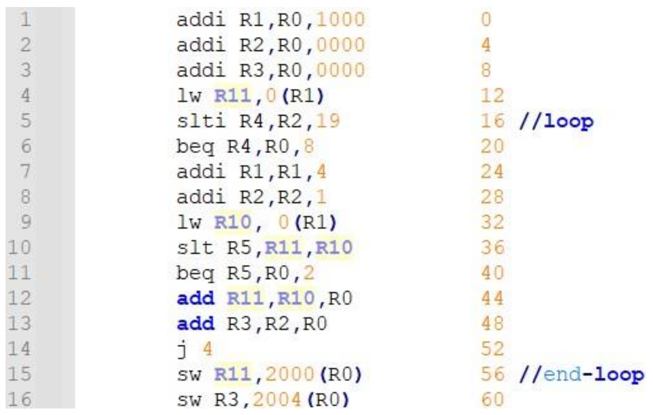
Msgs
/mips_tb/inst_adr 12 20 24 8 28 32 12 16
/mips_tb/inst 8c2b0064 2421fffc 1000fffb 10200004 ac0a00c8 0c000003 8c2b0064 014b5020
/mips_tb/data_adr 100 0 0 200 0 100 15
/mips_tb/data_in 1 4 0 15 0 1 0
/mips_tb/data_out 0 0 0 0
/mips_tb/mem_read 1
/mips_tb/mem_write 0
/mips_tb/clk 0
/mips_tb/rst 0
Now 520 ps
Cursor 1 498 ps
410 ps 420 ps 430 ps 440 ps 450 ps 460 ps 470 ps 480 ps 490 ps 500 ps 510 ps 520 ps
498 ps

Datapath:



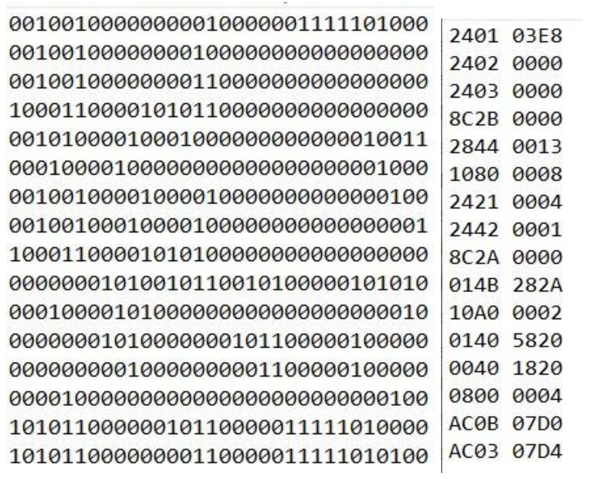
A demanded text file was demanded that would find the smallest element in an array, the smallest array index is supposed to be set on 2000 and 2004

The assembly codes:



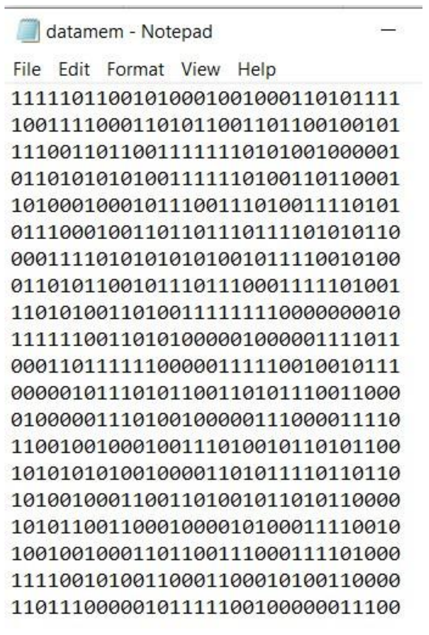
```
        addi R1,R0,1000          0
        addi R2,R0,0000          4
        addi R3,R0,0000          8
        lw R11,0(R1)             12
        slti R4,R2,19            16 //loop
        beq R4,R0,8              20
        addi R1,R1,4             24
        addi R2,R2,1             28
        lw R10, 0(R1)            32
        slt R5,R11,R10           36
        beq R5,R0,2              40
        add R11,R10,R0           44
        add R3,R2,R0             48
        j 4                      52
        sw R11,2000(R0)          56 //end-loop
        sw R3,2004(R0)           60
```

Instruction according to the assembly code and hexadecimal equivalent:



```
00100100000000010000001111101000   2401 03E8
00100100000000100000000000000000   2402 0000
00100100000000110000000000000000   2403 0000
10001100001010110000000000000000   8C2B 0000
00101000010001000000000000010011   2844 0013
00010000100000000000000000001000   1080 0008
00100100001000010000000000000100   2421 0004
00100100010000100000000000000001   2442 0001
10001100001010100000000000000000   8C2A 0000
00000001010010110010100000101010   014B 282A
00010000101000000000000000000010   10A0 0002
00000001010000000101100000100000   0140 5820
00000000010000000001100000100000   0040 1820
00001000000000000000000000000100   0800 0004
10101100000010110000011111010000   AC0B 07D0
10101100000000110000011111010100   AC03 07D4
```

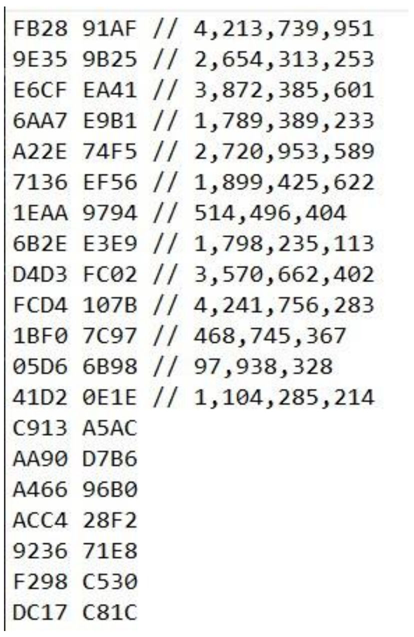
Data memory file:



datamem - Notepad

File Edit Format View Help

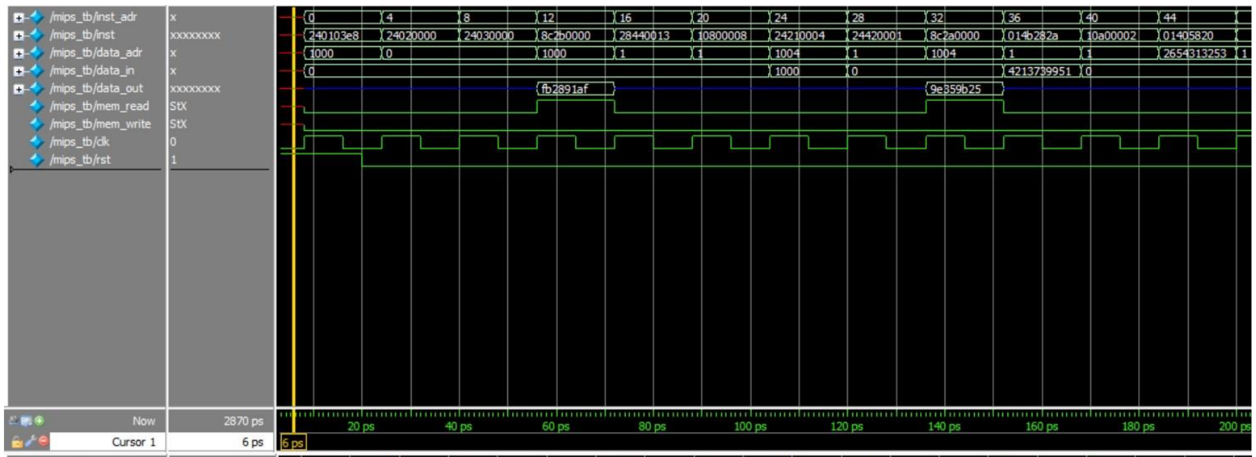
```
11111011001010001001000110101111
10011110001101011001101100100101
11100110110011111110101001000001
01101010101001111110100110110001
10100010001011100111010011110101
01110001001101101110111101010110
00011110101010101001011110010100
01101011001011101110001111101001
11010100110100111111110000000010
11111100110101000001000001111011
00011011111100000111110010010111
00000101110101100110101110011000
01000001110100100000111000011110
11001001000100111010010110101100
10101010100100001101011110110110
10100100011001101001011010110000
10101100110001000010100011110010
10010010001101100111000111101000
11110010100110001100010100110000
11011100000101111100100000011100
```

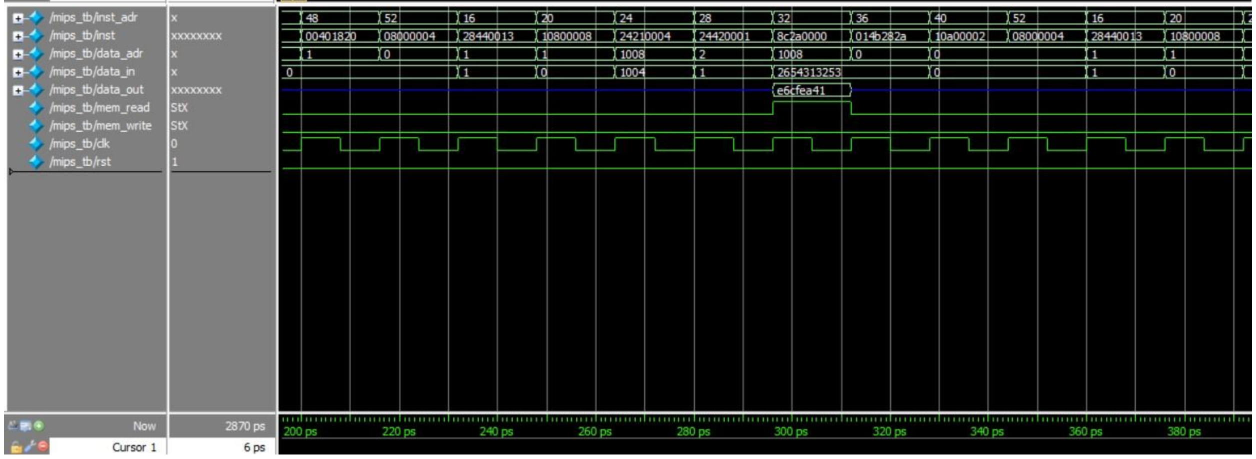


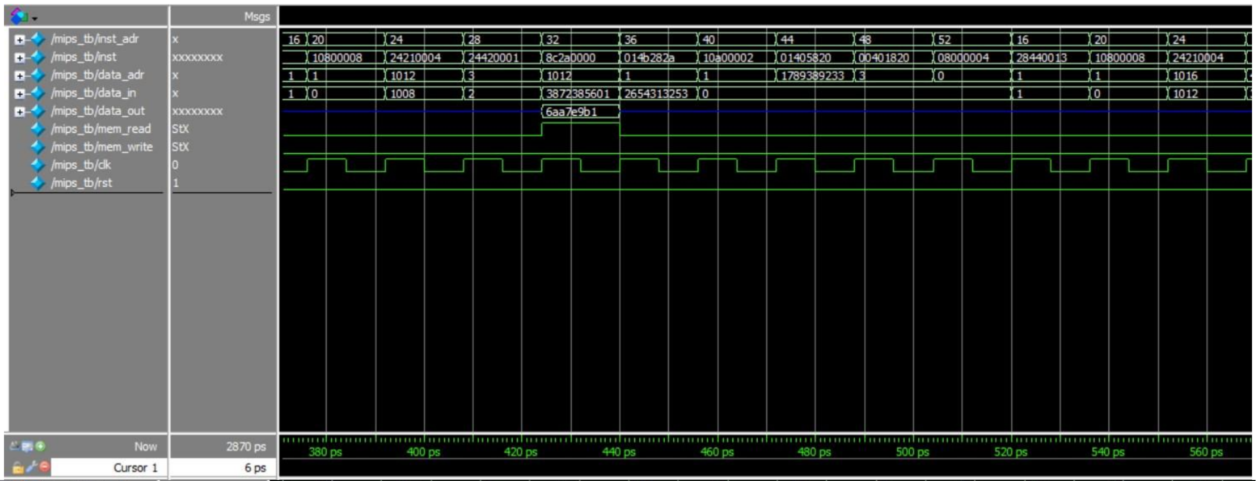
```
FB28 91AF // 4,213,739,951
9E35 9B25 // 2,654,313,253
E6CF EA41 // 3,872,385,601
6AA7 E9B1 // 1,789,389,233
A22E 74F5 // 2,720,953,589
7136 EF56 // 1,899,425,622
1EAA 9794 // 514,496,404
6B2E E3E9 // 1,798,235,113
D4D3 FC02 // 3,570,662,402
FCD4 107B // 4,241,756,283
1BF0 7C97 // 468,745,367
05D6 6B98 // 97,938,328
41D2 0E1E // 1,104,285,214
C913 A5AC
AA90 D7B6
A466 96B0
ACC4 28F2
9236 71E8
F298 C530
DC17 C81C
```

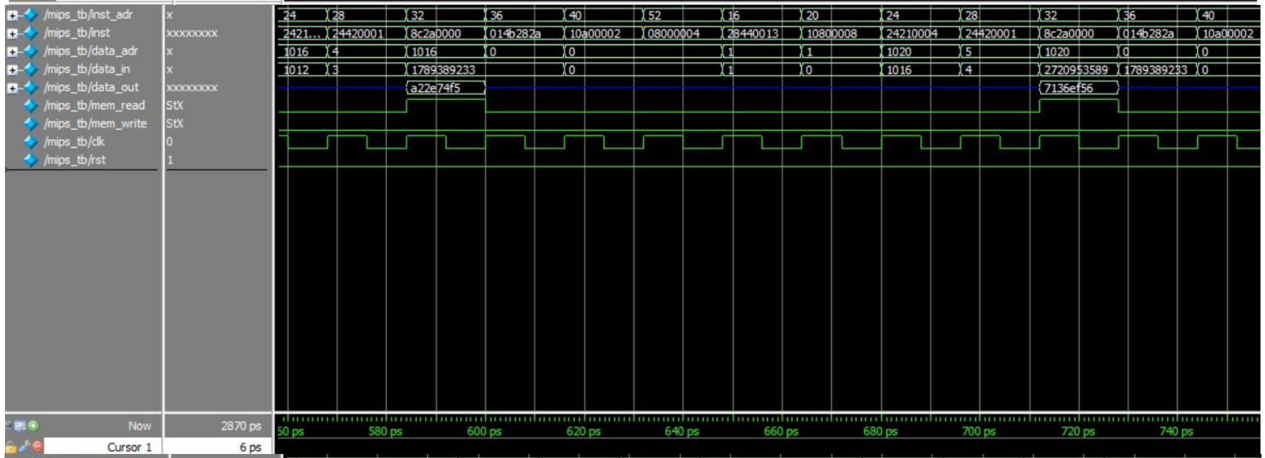
The data memory contains random data, and the smallest value is chosen among them, the smallest value is 97938328 and its index number is 11 which is stored in the 2000 and 2004 house of array in the last image of the images shown below.

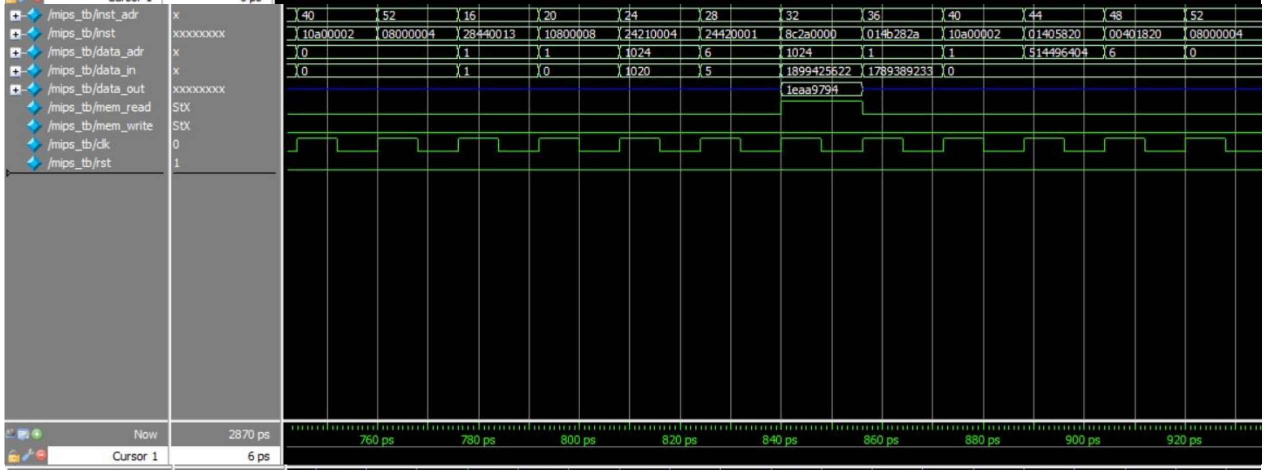
The waveform result:

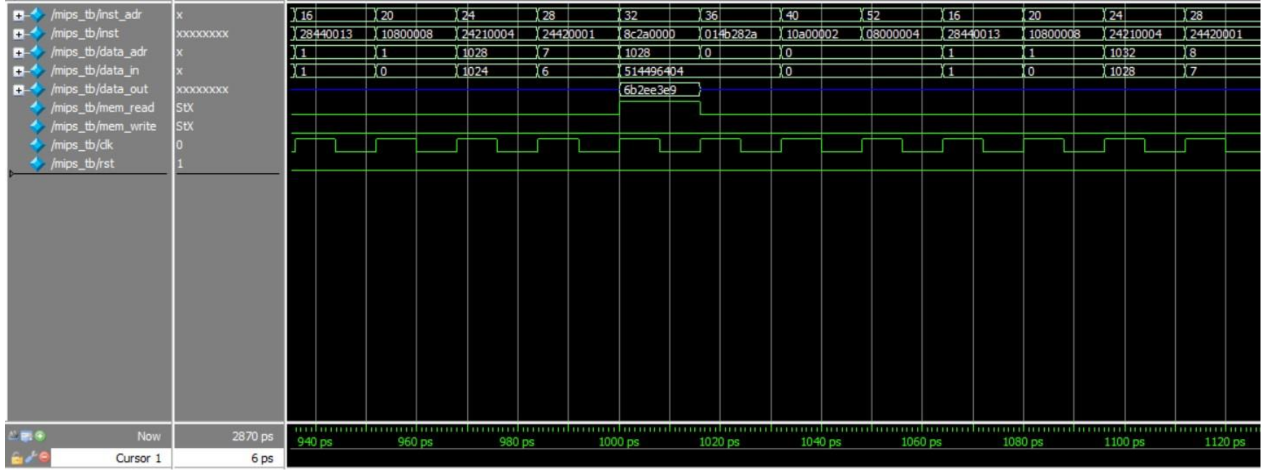

/mips_tb/inst_adr x 0 4 8 12 16 20 24 28 32 36 40 44
/mips_tb/inst xxxxxxxx 240103e8 24020000 24030000 8c2b0000 28440013 10800008 24210004 24420001 8c2a0000 014b282a 10a00002 01405820
/mips_tb/data_adr x 1000 0 1000 1 1 1004 1 1004 1 1 2654313253 1
/mips_tb/data_in x 0 1000 0 4213739951 0
/mips_tb/data_out xxxxxxxx fb2891af 9e359b25
/mips_tb/mem_read StX
/mips_tb/mem_write StX
/mips_tb/clk 0
/mips_tb/rst 1
Now 2870 ps
Cursor 1 6 ps
6 ps
20 ps 40 ps 60 ps 80 ps 100 ps 120 ps 140 ps 160 ps 180 ps 200 ps

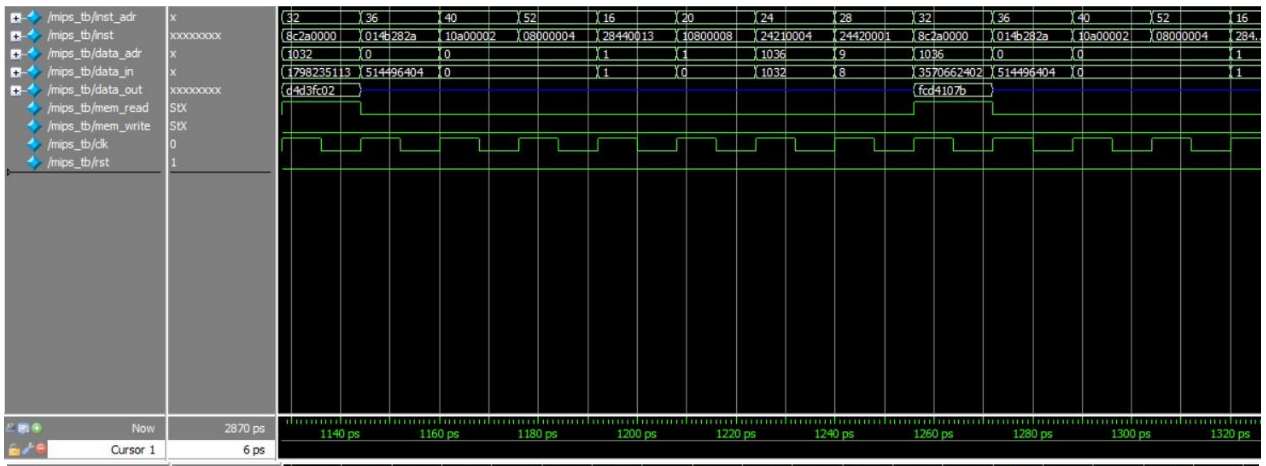

/mips_tb/inst_adr x 48 52 16 20 24 28 32 36 40 52 16 20
/mips_tb/inst xxxxxxxx 00401820 08000004 28440013 10800008 24210004 24420001 8c2a0000 014b282a 10a00002 08000004 28440013 10800008
/mips_tb/data_adr x 1 0 1 1 1008 2 1008 0 0 1 1
/mips_tb/data_in x 0 1 0 1004 1 2654313253 0 1 0
/mips_tb/data_out xxxxxxxx e6cfea41
/mips_tb/mem_read StX
/mips_tb/mem_write StX
/mips_tb/clk 0
/mips_tb/rst 1
Now 2870 ps
Cursor 1 6 ps
200 ps 220 ps 240 ps 260 ps 280 ps 300 ps 320 ps 340 ps 360 ps 380 ps

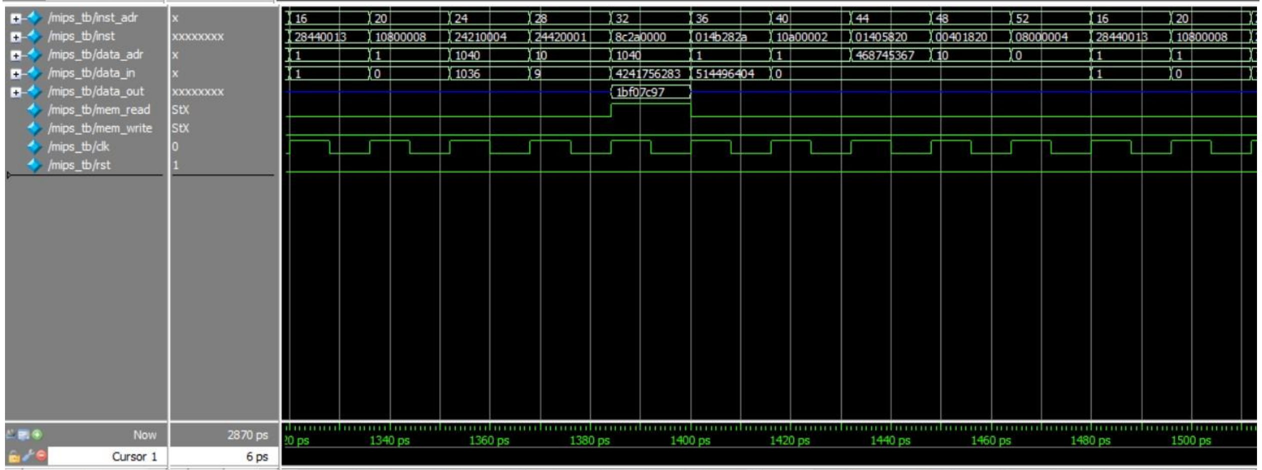

Msgs
/mips_tb/inst_adr x 16 20 24 28 32 36 40 44 48 52 16 20 24
/mips_tb/inst xxxxxxxx 10800008 24210004 24420001 8c2a0000 014b282a 10a00002 01405820 00401820 08000004 28440013 10800008 24210004
/mips_tb/data_adr x 1 1 1012 3 1012 1 1 1789389233 3 0 1 1 1016
/mips_tb/data_in x 1 0 1008 2 3872385601 2654313253 0 1 0 1012
/mips_tb/data_out xxxxxxxx 6aa7e9b1
/mips_tb/mem_read StX
/mips_tb/mem_write StX
/mips_tb/clk 0
/mips_tb/rst 1
Now 2870 ps
Cursor 1 6 ps
380 ps 400 ps 420 ps 440 ps 460 ps 480 ps 500 ps 520 ps 540 ps 560 ps

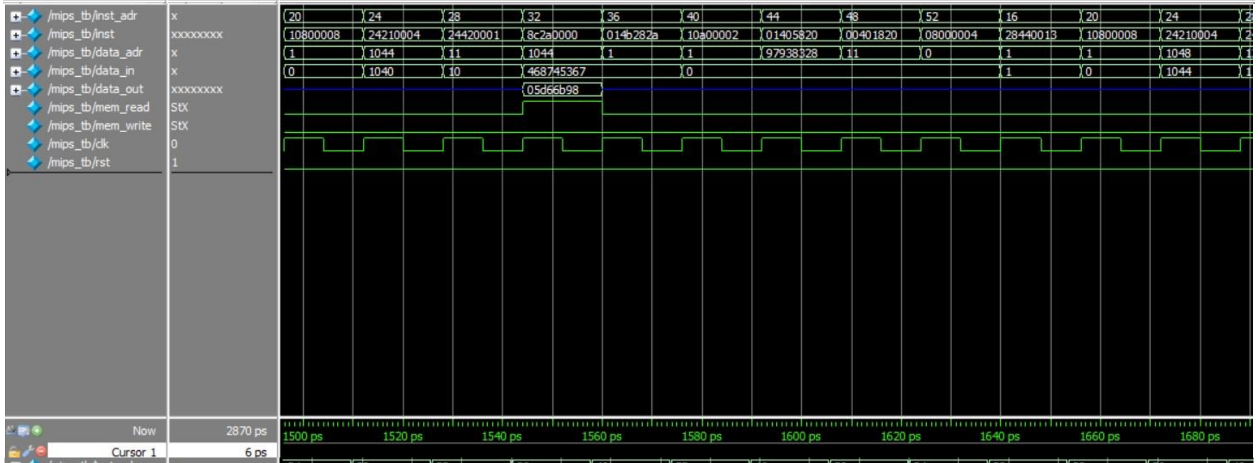

/mips_tb/inst_adr x 24 28 32 36 40 52 16 20 24 28 32 36 40
/mips_tb/inst xxxxxxxx 2421... 24420001 8c2a0000 014b282a 10a00002 08000004 28440013 10800008 24210004 24420001 8c2a0000 014b282a 10a00002
/mips_tb/data_adr x 1016 4 1016 0 0 1 1 1020 5 1020 0 0
/mips_tb/data_in x 1012 3 1789389233 0 1 0 1016 4 2720953589 1789389233 0
/mips_tb/data_out xxxxxxxx a22e74f5 7136ef56
/mips_tb/mem_read StX
/mips_tb/mem_write StX
/mips_tb/clk 0
/mips_tb/rst 1
Now 2870 ps
Cursor 1 6 ps
50 ps 580 ps 600 ps 620 ps 640 ps 660 ps 680 ps 700 ps 720 ps 740 ps

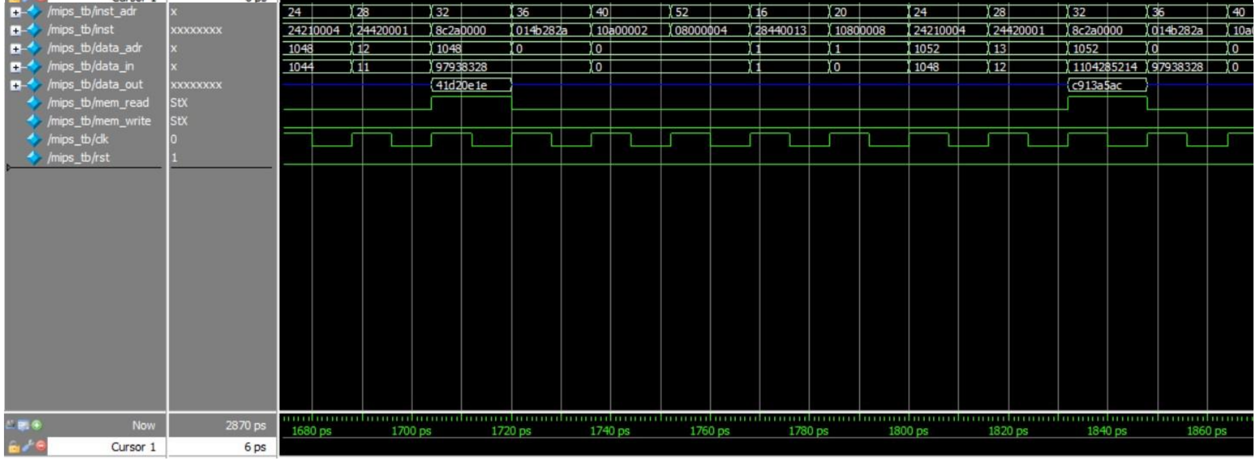

/mips_tb/inst_adr x 40 52 16 20 24 28 32 36 40 44 48 52
/mips_tb/inst xxxxxxxx 10a00002 08000004 28440013 10800008 24210004 24420001 8c2a0000 014b282a 10a00002 01405820 00401820 08000004
/mips_tb/data_adr x 0 1 1 1024 6 1024 1 1 514496404 6 0
/mips_tb/data_in x 0 1 0 1020 5 1899425622 1789389233 0
/mips_tb/data_out xxxxxxxx 1eaa9794
/mips_tb/mem_read StX
/mips_tb/mem_write StX
/mips_tb/clk 0
/mips_tb/rst 1
Now 2870 ps
Cursor 1 6 ps
760 ps 780 ps 800 ps 820 ps 840 ps 860 ps 880 ps 900 ps 920 ps

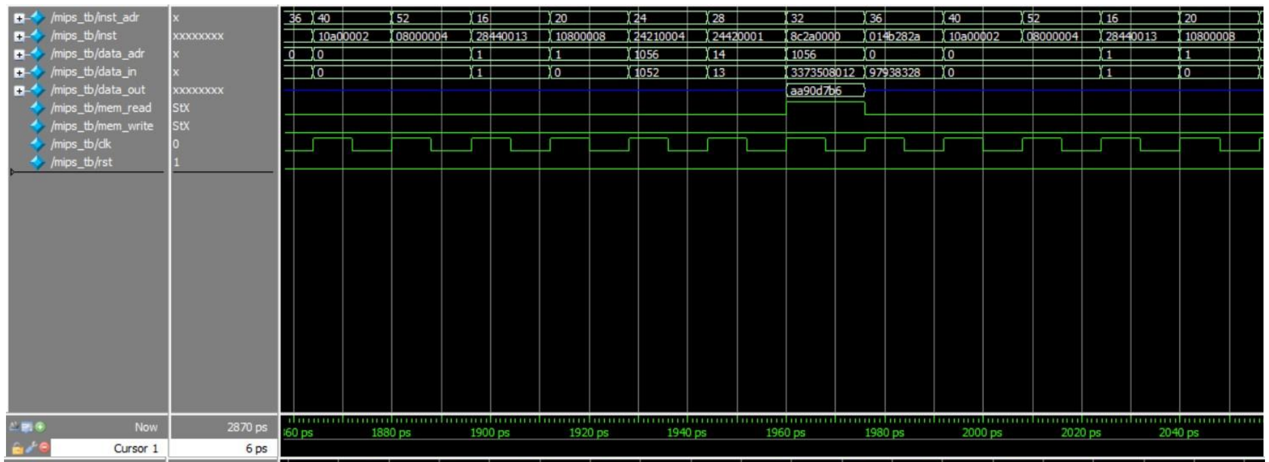

/mips_tb/inst_adr x 16 20 24 28 32 36 40 52 16 20 24 28
/mips_tb/inst xxxxxxxx 28440013 10800008 24210004 24420001 8c2a0000 014b282a 10a00002 08000004 28440013 10800008 24210004 24420001
/mips_tb/data_adr x 1 1 1028 7 1028 0 0 1 1 1032 8
/mips_tb/data_in x 1 0 1024 6 514496404 0 1 0 1028 7
/mips_tb/data_out xxxxxxxx 6b2ee3e9
/mips_tb/mem_read StX
/mips_tb/mem_write StX
/mips_tb/clk 0
/mips_tb/rst 1
Now 2870 ps
Cursor 1 6 ps
940 ps 960 ps 980 ps 1000 ps 1020 ps 1040 ps 1060 ps 1080 ps 1100 ps 1120 ps

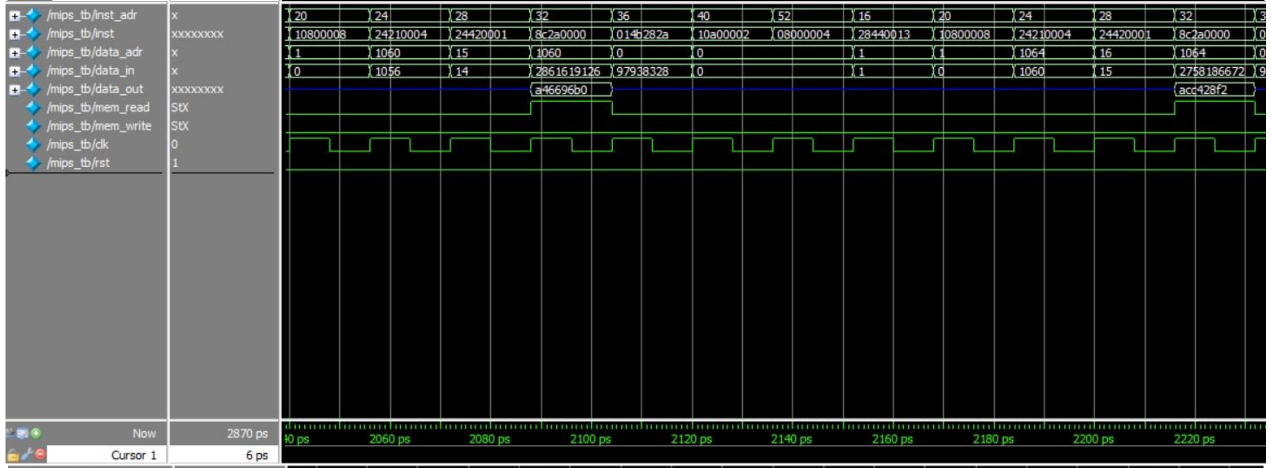

/mips_tb/inst_adr x
32 36 40 52 16 20 24 28 32 36 40 52 16
/mips_tb/inst xxxxxxxx
8c2a0000 014b282a 10a00002 08000004 28440013 10800008 24210004 24420001 8c2a0000 014b282a 10a00002 08000004 284...
/mips_tb/data_adr x
1032 0 0 1 1 1036 9 1036 0 0 1
/mips_tb/data_in x
1798235113 514496404 0 1 0 1032 8 3570662402 514496404 0 1
/mips_tb/data_out xxxxxxxx
d4d3fc02 fcd4107b
/mips_tb/mem_read StX
/mips_tb/mem_write StX
/mips_tb/clk 0
/mips_tb/rst 1
Now 2870 ps
Cursor 1 6 ps
1140 ps 1160 ps 1180 ps 1200 ps 1220 ps 1240 ps 1260 ps 1280 ps 1300 ps 1320 ps

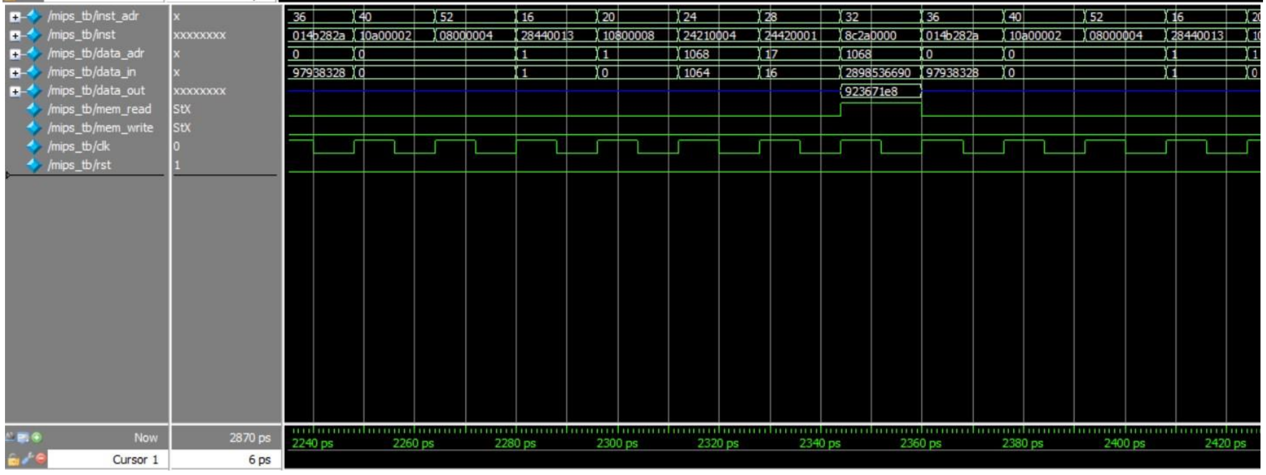

/mips_tb/inst_adr x
16 20 24 28 32 36 40 44 48 52 16 20
/mips_tb/inst xxxxxxxx
28440013 10800008 24210004 24420001 8c2a0000 014b282a 10a00002 01405820 00401820 08000004 28440013 10800008
/mips_tb/data_adr x
1 1 1040 10 1040 1 1 468745367 10 0 1 1
/mips_tb/data_in x
1 0 1036 9 4241756283 514496404 0 1 0
/mips_tb/data_out xxxxxxxx
1bf07c97
/mips_tb/mem_read StX
/mips_tb/mem_write StX
/mips_tb/clk 0
/mips_tb/rst 1
Now 2870 ps
Cursor 1 6 ps
0 ps 1340 ps 1360 ps 1380 ps 1400 ps 1420 ps 1440 ps 1460 ps 1480 ps 1500 ps

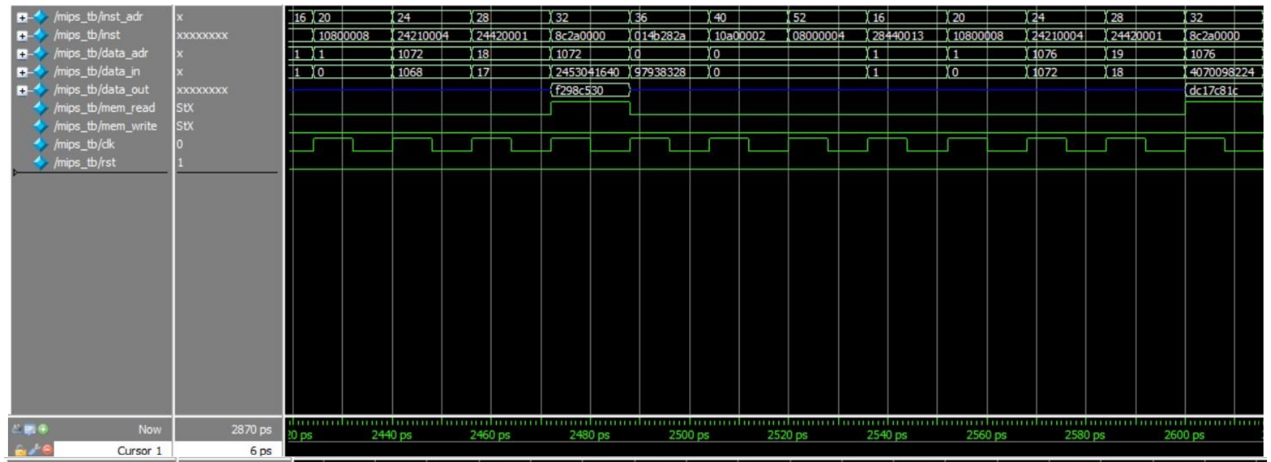

/mips_tb/inst_adr x
20 24 28 32 36 40 44 48 52 16 20 24
/mips_tb/inst xxxxxxxx
10800008 24210004 24420001 8c2a0000 014b282a 10a00002 01405820 00401820 08000004 28440013 10800008 24210004
/mips_tb/data_adr x
1 1044 11 1044 1 1 97938328 11 0 1 1 1048
/mips_tb/data_in x
0 1040 10 468745367 0 1 0 1044
/mips_tb/data_out xxxxxxxx
05d66b98
/mips_tb/mem_read StX
/mips_tb/mem_write StX
/mips_tb/clk 0
/mips_tb/rst 1
Now 2870 ps
Cursor 1 6 ps
1500 ps 1520 ps 1540 ps 1560 ps 1580 ps 1600 ps 1620 ps 1640 ps 1660 ps 1680 ps

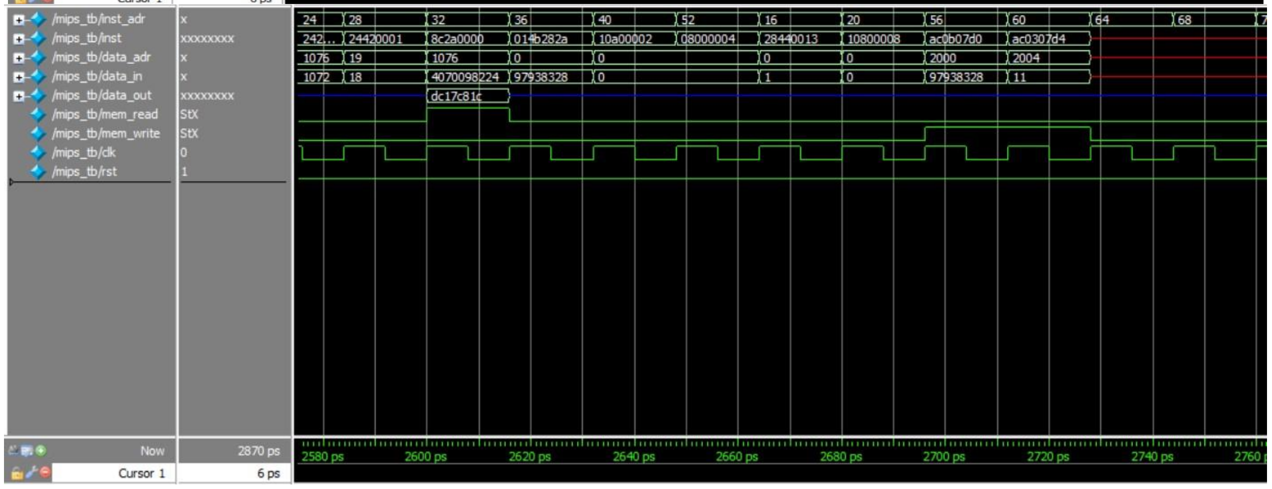

/mips_tb/inst_adr x
24 28 32 36 40 52 16 20 24 28 32 36 40
/mips_tb/inst xxxxxxxx
24210004 24420001 8c2a0000 014b282a 10a00002 08000004 28440013 10800008 24210004 24420001 8c2a0000 014b282a 10a
/mips_tb/data_adr x
1048 12 1048 0 0 1 1 1052 13 1052 0 0
/mips_tb/data_in x
1044 11 97938328 0 1 0 1048 12 1104285214 97938328 0
/mips_tb/data_out xxxxxxxx
41d20e1e c913a5ac
/mips_tb/mem_read StX
/mips_tb/mem_write StX
/mips_tb/clk 0
/mips_tb/rst 1
Now 2870 ps
Cursor 1 6 ps
1680 ps 1700 ps 1720 ps 1740 ps 1760 ps 1780 ps 1800 ps 1820 ps 1840 ps 1860 ps


/mips_tb/inst_adr
/mips_tb/inst
/mips_tb/data_adr
/mips_tb/data_in
/mips_tb/data_out
/mips_tb/mem_read
/mips_tb/mem_write
/mips_tb/clk
/mips_tb/rst
x
xxxxxxxx
x
x
xxxxxxxx
StX
StX
0
1
36 40 52 16 20 24 28 32 36 40 52 16 20
10a00002 08000004 28440013 10800008 24210004 24420001 8c2a0000 014b282a 10a00002 08000004 28440013 10800008
0 0 1 1 1056 14 1056 0 0 1 1
0 1 0 1052 13 3373508012 97938328 0 1 0
aa90d7b6
Now 2870 ps
Cursor 1 6 ps
60 ps 1880 ps 1900 ps 1920 ps 1940 ps 1960 ps 1980 ps 2000 ps 2020 ps 2040 ps


/mips_tb/inst_adr
/mips_tb/inst
/mips_tb/data_adr
/mips_tb/data_in
/mips_tb/data_out
/mips_tb/mem_read
/mips_tb/mem_write
/mips_tb/clk
/mips_tb/rst
x
xxxxxxxx
x
x
xxxxxxxx
StX
StX
0
1
20 24 28 32 36 40 52 16 20 24 28 32
10800008 24210004 24420001 8c2a0000 014b282a 10a00002 08000004 28440013 10800008 24210004 24420001 8c2a0000
1 1060 15 1060 0 0 1 1 1064 16 1064
0 1056 14 2861619126 97938328 0 1 0 1060 15 2758186672
a46696b0
acc428f2
Now 2870 ps
Cursor 1 6 ps
0 ps 2060 ps 2080 ps 2100 ps 2120 ps 2140 ps 2160 ps 2180 ps 2200 ps 2220 ps


/mips_tb/inst_adr
/mips_tb/inst
/mips_tb/data_adr
/mips_tb/data_in
/mips_tb/data_out
/mips_tb/mem_read
/mips_tb/mem_write
/mips_tb/clk
/mips_tb/rst
x
xxxxxxxx
x
x
xxxxxxxx
StX
StX
0
1
36 40 52 16 20 24 28 32 36 40 52 16
014b282a 10a00002 08000004 28440013 10800008 24210004 24420001 8c2a0000 014b282a 10a00002 08000004 28440013
0 0 1 1 1068 17 1068 0 0 1 1
97938328 0 1 0 1064 16 2898536690 97938328 0 1 0
923671e8
Now 2870 ps
Cursor 1 6 ps
2240 ps 2260 ps 2280 ps 2300 ps 2320 ps 2340 ps 2360 ps 2380 ps 2400 ps 2420 ps


/mips_tb/inst_adr
/mips_tb/inst
/mips_tb/data_adr
/mips_tb/data_in
/mips_tb/data_out
/mips_tb/mem_read
/mips_tb/mem_write
/mips_tb/clk
/mips_tb/rst
Now 2870 ps
Cursor 1 6 ps
2440 ps
2460 ps
2480 ps
2500 ps
2520 ps
2540 ps
2560 ps
2580 ps
2600 ps


/mips_tb/inst_adr
/mips_tb/inst
/mips_tb/data_adr
/mips_tb/data_in
/mips_tb/data_out
/mips_tb/mem_read
/mips_tb/mem_write
/mips_tb/clk
/mips_tb/rst
Now 2870 ps
Cursor 1 6 ps
2580 ps
2600 ps
2620 ps
2640 ps
2660 ps
2680 ps
2700 ps
2720 ps
2740 ps
2760 ps