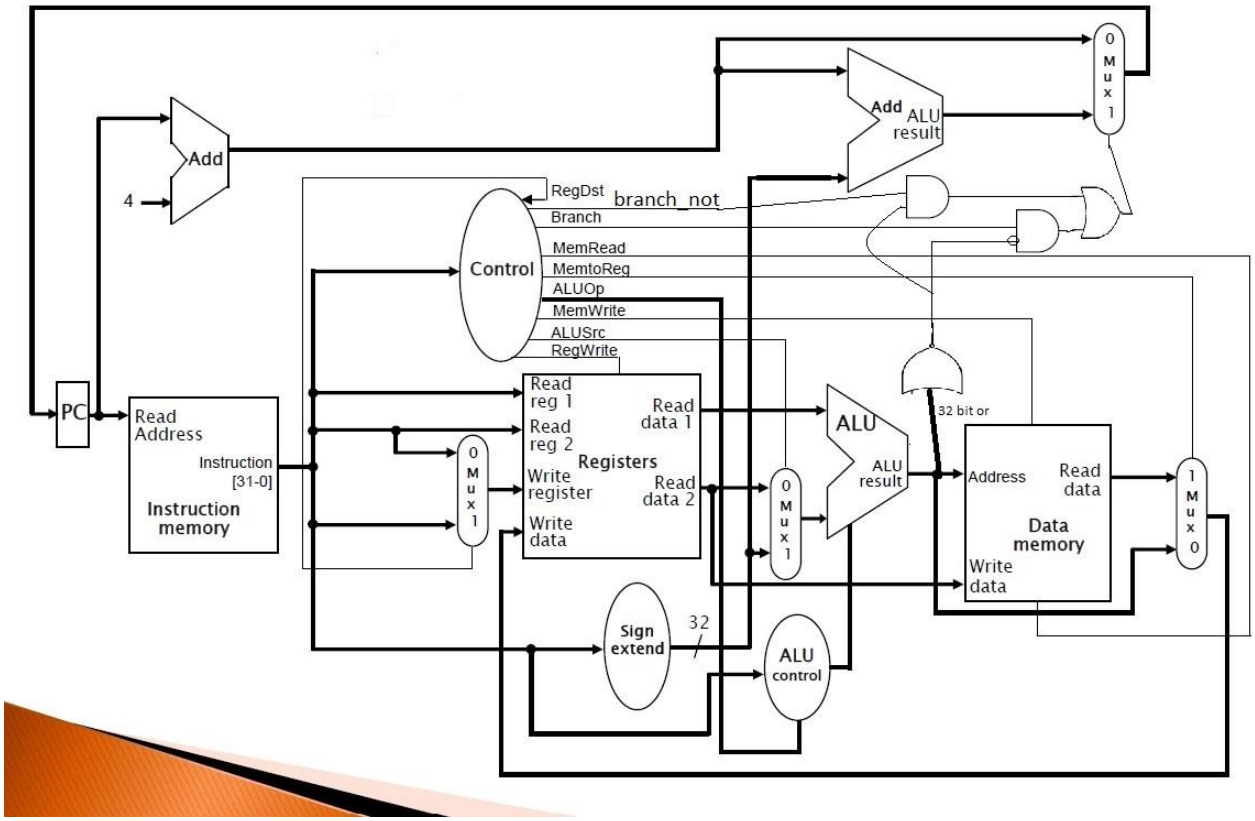
CSE 331 HW-4



My processor is exactly the same as above

Instruction opcode	P2 P1 P0 ALUop	F2 F1 F0 Function	Desired ALU action	C2 C1 C0 ALU control
R-TYPE	111	000	AND	110
R-TYPE	111	001	ADD	000
R-TYPE	111	010	SUB	010
R-TYPE	111	011	XOR	001
R-TYPE	111	100	NOR	101
R-TYPE	111	101	OR	111
ADDI	000	XXX	ADD	000
ANDI	001	XXX	AND	110
ORI	010	XXX	OR	111
NORI	011	XXX	NOR	101
BEQ	100	XXX	SUB	010
BNE	101	XXX	SUB	010
SLTI	110	XXX	SLT	100
LW	000	XXX	ADD	000
SW	000	XXX	ADD	000

Instr	Reg Dest	AluSrc	Memto Reg	Reg Wr	Mem Rd	Mem Wr	Branch	ALUop2	ALUop1	ALUop0
R-type	1	0	0	1	0	0	0	1	1	1
ADDI	0	1	0	1	0	0	0	0	0	0
ANDI	0	1	0	1	0	0	0	0	0	1
ORI	0	1	0	1	0	0	0	0	1	0
NORI	0	1	0	1	0	0	0	0	1	1
BEQ	X	0	X	0	0	0	1	1	0	0
BNE	X	0	X	0	0	0	1	1	0	1
SLTI	0	1	0	1	0	0	0	1	1	0
LW	0	1	1	1	1	0	0	0	0	0
SW	X	1	X	0	0	1	0	0	0	0

Test Instruction memory module

```
initial begin
    $readmemb("instruction.txt",ahmet.memory);
    pc=5'b00001;
    #`DELAY;
    pc=5'b00011;
    #`DELAY;
end
```

```
VSIM 6> run
# ,pc=00001,instruction=0000000101100000
# ,pc=00011,instruction=0000101011100001
```

```
1  0000010001100000
2  0000000101100000  <
3  0000010001100001
4  0000101011100001  <
5  0000010001100010
6  0000101011100010
7  0000010001100011
```

Test Registers module

```
$readmemb("register.txt",ahmet.reg_array);
#`DELAY;
read_reg_1=3'b001; read_reg_2=3'b010; write_reg=3'b011; RegWrite=1'b0; clk=1'b1;
#`DELAY;
read_reg_1=3'b101; read_reg_2=3'b011; write_reg=3'b001; RegWrite=1'b0; clk=1'b1;
```

```
# read_data_1=00000000000000000000000000000001,read_data_2=00000000000000000000000000000010, write_data=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx(Since it is in the registers module, there is no data yet),
#  read_reg_1=001, read_reg_2=010, write_reg=011, RegWrite=0, clk=1
# read_data_1=00000000000000000000000000000000,read_data_2=00000000000000000000000000000011, write_data=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx(Since it is in the registers module, there is no data yet),
#  read_reg_1=101, read_reg_2=011, write_reg=001, RegWrite=0, clk=1
```

```
1   00000000000000000000000000000000
2   00000000000000000000000000000001   <
3   00000000000000000000000000000010   <
4   00000000000000000000000000000011
5   00000000000000000000000000000100
6   00000000000000000000000000000000   <
7   00000000000000000000000000000110
8   00000000000000000000000000000111
```

Test Main Control module

```
op = 4'b0000;/*R-type*/
#`DELAY;
op = 4'b0001;/*I-type addı-ori..*/
#`DELAY;
op = 4'b0101;/*I-type BEQ*/
#`DELAY;
op = 4'b1000;/*I-type lw*/
#`DELAY;
op = 4'b1001;/*I-type SW*/
#`DELAY;
```

```
# op=0000,RegDst=1,ALUSrc=0,MemtoReg=0,RegWrite=1,MemRead=0,MemWrite=0,Branch=0,Branch_not=0,ALUop0=1,ALUop1=1,ALUop2=1
# op=0001,RegDst=0,ALUSrc=1,MemtoReg=0,RegWrite=1,MemRead=0,MemWrite=0,Branch=0,Branch_not=0,ALUop0=0,ALUop1=0,ALUop2=0
# op=0101,RegDst=0,ALUSrc=0,MemtoReg=0,RegWrite=0,MemRead=0,MemWrite=0,Branch=1,Branch_not=0,ALUop0=0,ALUop1=0,ALUop2=1
# op=1000,RegDst=0,ALUSrc=1,MemtoReg=1,RegWrite=1,MemRead=1,MemWrite=0,Branch=0,Branch_not=0,ALUop0=0,ALUop1=0,ALUop2=0
# op=1001,RegDst=0,ALUSrc=1,MemtoReg=0,RegWrite=0,MemRead=0,MemWrite=1,Branch=0,Branch_not=0,ALUop0=0,ALUop1=0,ALUop2=0
```

Test Alu Control module

```
ALUop2=1'b1; ALUop1=1'b1; ALUop0=1'b1; Func=3'b000;
#`DELAY;
ALUop2=1'b1; ALUop1=1'b1; ALUop0=1'b1; Func=3'b001;
#`DELAY;
ALUop2=1'b0; ALUop1=1'b0; ALUop0=1'b0;
#`DELAY;
ALUop2=1'b1; ALUop1=1'b0; ALUop0=1'b0;
#`DELAY;
ALUop2=1'b0; ALUop1=1'b1; ALUop0=1'b0;
#`DELAY;
ALUop2=1'b1; ALUop1=1'b0; ALUop0=1'b1;
#`DELAY;
```

```
# ALUop2=1,ALUop1=1,ALUop0=1,Func=000,C2=1,C1=1,C0=0
# ALUop2=1,ALUop1=1,ALUop0=1,Func=001,C2=0,C1=0,C0=0
# ALUop2=0,ALUop1=0,ALUop0=0,Func=001,C2=0,C1=0,C0=0
# ALUop2=1,ALUop1=0,ALUop0=0,Func=001,C2=0,C1=1,C0=0
# ALUop2=0,ALUop1=1,ALUop0=0,Func=001,C2=1,C1=1,C0=1
```

Test ALU module

```
    A = 32'b00000001000000010000000100000001;
    B = 32'b01100001011000010110000101100001;
    Ci=1'b0;
    S=3'b000;
    $monitor("time = %2d, A =%32b, B=%32b, A+B=%32b,S=%3b",$time, A, B,F,S);
#`DELAY;
    A = 32'b00000001000000010000000100000001;
    B = 32'b01100001011000010110000101100001;
    Ci=1'b0;
    S=3'b001;
    $monitor("time = %2d, A =%32b, B=%32b, A XOR B=%32b,S=%3b",$time, A, B,F,S);
#`DELAY;
    A = 32'b10000001000000010000000100000001;
    B = 32'b01100001011000010110000101100001;
    Ci=1'b1;
    S=3'b010;
    $monitor("time = %2d, A =%32b, B=%32b, A-B=%32b,S=%3b",$time, A, B,F,S);
#`DELAY;
    A = 32'b10000001000000010000000100000001;
    B = 32'b01100001011000010110000101100001;
    Ci=1'b0;
    S=3'b100;
    $monitor("time = %2d, A =%32b, B=%32b, A>B=%32b,S=%3b",$time, A, B,F,S);
#`DELAY;
    A = 32'b10000001000000010000000100000001;
    B = 32'b01100001011000010110000101100001;
    Ci=1'b0;
    S=3'b101;
    $monitor("time = %2d, A =%32b, B=%32b, A NOR B=%32b,S=%3b",$time, A, B,F,S);
#`DELAY;
    A = 32'b10000001000000010000000100000001;
    B = 32'b01100001011000010110000101100001;
    Ci=1'b0;
    S=3'b111;
```

```
VSIM 10> run
# time =  0, A =00000001000000010000000100000001, B=01100001011000010110000101100001, A+B=01100010011000100110001001100010,S=000
# time = 20, A =00000001000000010000000100000001, B=01100001011000010110000101100001, A XOR B=01100000011000000110000001100000,S=001
# time = 40, A =10000001000000010000000100000001, B=01100001011000010110000101100001, A-B=00011111100111111001111110100000,S=010
# time = 60, A =10000001000000010000000100000001, B=01100001011000010110000101100001, A>B=10000000000000000000000000000000,S=100
# time = 80, A =10000001000000010000000100000001, B=01100001011000010110000101100001, A NOR B=00011110100111101001111010011110,S=101
VSIM 11> run
# time = 100, A =10000001000000010000000100000001, B=01100001011000010110000101100001, A OR B=11100001011000010110000101100001,S=111
```

Test Data Memory module

```
$readmemb("data.txt",ahmet.memory);
access_addr=32'b00000000000000000000000000000011; mem_read=1'b1; mem_write=1'b0;
```

```
# access_addr=00000000000000000000000000000011,mem_read=1,mem_write=0,read_data=00000000000000000000000000000011
```

```
00000000000000000000000000000000
00000000000000000000000000010100
00000000000000000000000000000010
00000000000000000000000000000011 <
00000000000000000000000000000100
00000000000000000000000000000101
```

TEST ALL INSTRUCTION

```
# time:  0,pc=00000 ,ins: 0000010001100000
# ,read data 1=00000000000000000000000000000010,read data 2=00000000000000000000000000000001, alu input 2=00000000000000000000000000000001,write Data: 00000000000000000000000000000000
#
# time: 10,pc=00001 ,ins: 0000000101100000
# ,read data 1=00000000000000000000000000000000,read data 2=00000000000000000000000000000101, alu input 2=00000000000000000000000000000101,write Data: 00000000000000000000000000000000
#
# time: 30,pc=00010 ,ins: 0000010001100001
# ,read data 1=00000000000000000000000000000010,read data 2=00000000000000000000000000000001, alu input 2=00000000000000000000000000000001,write Data: 00000000000000000000000000000011
#
# time: 50,pc=00011 ,ins: 0000101011100001
# ,read data 1=00000000000000000000000000000101,read data 2=00000000000000000000000000000011, alu input 2=00000000000000000000000000000011,write Data: 00000000000000000000000000001000
#
# time: 70,pc=00100 ,ins: 0000010001100010
# ,read data 1=00000000000000000000000000000010,read data 2=00000000000000000000000000000001, alu input 2=00000000000000000000000000000001,write Data: 00000000000000000000000000000001
#
# time: 90,pc=00101 ,ins: 0000101011100010
# ,read data 1=00000000000000000000000000000101,read data 2=00000000000000000000000000000011, alu input 2=00000000000000000000000000000011,write Data: 00000000000000000000000000000010
#
# time: 110,pc=00110 ,ins: 0000010001100011
# ,read data 1=00000000000000000000000000000010,read data 2=00000000000000000000000000000001, alu input 2=00000000000000000000000000000001,write Data: 00000000000000000000000000000011
#
# time: 130,pc=00111 ,ins: 0000101011100011
# ,read data 1=00000000000000000000000000000101,read data 2=00000000000000000000000000000011, alu input 2=00000000000000000000000000000011,write Data: 00000000000000000000000000000110
#
# time: 150,pc=01000 ,ins: 0000010001100100
# ,read data 1=00000000000000000000000000000010,read data 2=00000000000000000000000000000001, alu input 2=00000000000000000000000000000001,write Data: 11111111111111111111111111111100
#
# time: 170,pc=01001 ,ins: 0000101011100100
# ,read data 1=00000000000000000000000000000101,read data 2=00000000000000000000000000000011, alu input 2=00000000000000000000000000000011,write Data: 11111111111111111111111111111000
#
# time: 190,pc=01010 ,ins: 0000010001100101
# ,read data 1=00000000000000000000000000000010,read data 2=00000000000000000000000000000001, alu input 2=00000000000000000000000000000001,write Data: 00000000000000000000000000000011
#
# time: 210,pc=01011 ,ins: 0000101011100101
# ,read data 1=00000000000000000000000000000101,read data 2=00000000000000000000000000000011, alu input 2=00000000000000000000000000000011,write Data: 00000000000000000000000000000111
#
# time: 230,pc=01100 ,ins: 0001010100100101
# ,read data 1=00000000000000000000000000000010,read data 2=00000000000000000000000000000111, alu input 2=00000000000000000000000000100101,write Data: 00000000000000000000000000100111
```

R2 AND R1

R0 AND R5

R2 ADD R1

R5 ADD R3

R2 SUB R1

R5 SUB R3

R2 XOR R1

R5 XOR R3

R2 NOR R1

R5 NOR R3

R2 OR R3

R5 OR R3

R1 ADDI imm(ALU İNP)

```
# time: 250,pc=01101 ,ins: 0001101100100101
# ,read data 1=00000000000000000000000000000101,read data 2=00000000000000000000000000100111, alu input 2=00000000000000000000000000100101,write Data: 00000000000000000000000000101010
#
# time: 270,pc=01110 ,ins: 0010010100100101
# ,read data 1=00000000000000000000000000000010,read data 2=00000000000000000000000000101010, alu input 2=00000000000000000000000000100101,write Data: 00000000000000000000000000000000
#
# time: 290,pc=01111 ,ins: 0010101100100101
# ,read data 1=00000000000000000000000000000101,read data 2=00000000000000000000000000000000, alu input 2=00000000000000000000000000100101,write Data: 00000000000000000000000000000101
#
# time: 310,pc=10000 ,ins: 0011010100100101
# ,read data 1=00000000000000000000000000000010,read data 2=00000000000000000000000000000101, alu input 2=00000000000000000000000000100101,write Data: 00000000000000000000000000100111
#
# time: 330,pc=10001 ,ins: 0011101100100101
# ,read data 1=00000000000000000000000000000101,read data 2=00000000000000000000000000100111, alu input 2=00000000000000000000000000100101,write Data: 00000000000000000000000000100101
#
# time: 350,pc=10010 ,ins: 0100010100100101
# ,read data 1=00000000000000000000000000000010,read data 2=00000000000000000000000000100101, alu input 2=00000000000000000000000000100101,write Data: 11111111111111111111111111011000
#
# time: 370,pc=10011 ,ins: 0100101100100101
# ,read data 1=00000000000000000000000000000101,read data 2=11111111111111111111111111011000, alu input 2=00000000000000000000000000100101,write Data: 11111111111111111111111111011010
#
# time: 390,pc=10100 ,ins: 0111010100100101
# ,read data 1=00000000000000000000000000000010,read data 2=11111111111111111111111111011010, alu input 2=00000000000000000000000000100101,write Data: 00000000000000000000000000000010
#
# time: 410,pc=10101 ,ins: 0111101100100101
# ,read data 1=00000000000000000000000000000101,read data 2=00000000000000000000000000000010, alu input 2=00000000000000000000000000100101,write Data: 00000000000000000000000000000000
#
# time: 430,pc=10110 ,ins: 1000010100000101
# ,read data 1=00000000000000000000000000000010,read data 2=00000000000000000000000000000000, alu input 2=00000000000000000000000000000101,write Data: 00000000000000000000000000000111
#
# time: 450,pc=10111 ,ins: 1000101100000011
```

R5 ADDI imm

R2 ANDI imm

R5 ANDI imm

R2 ORI imm

R5 ORI imm

R2 NORI imm

R5 NORI imm

R2 SLTI imm

R5 SLTI imm

Lw (R2+imm)

```
1  0000010001100000
2  0000000101100000
3  0000010001100001
4  0000101011100001
5  0000010001100010
6  0000101011100010
7  0000010001100011
8  0000101011100011
9  0000010001100100
10 0000101011100100
11 0000010001100101
12 0000101011100101
13 0001010100100101
14 0001101100100101
15 0010010100100101
16 0010101100100101
17 0011010100100101
18 0011101100100101
19 0100010100100101
20 0100101100100101
21 0111010100100101
22 0111101100100101
23 1000010100000101
24 1000101100000011
```

Instructions (Not include BEQ BNE SW)

```
1  00000000000000000000000000000000
2  00000000000000000000000000000001
3  00000000000000000000000000000010
4  00000000000000000000000000000011
5  00000000000000000000000000000100
6  00000000000000000000000000000101
7  00000000000000000000000000000110
8  00000000000000000000000000000111
```

Registers

BEQ,BNE TESTS

```
# time:  0,pc=00000 ,ins: 0101010001000010
# ,read data 1=00000000000000000000000000000010,read data 2=00000000000000000000000000000001, alu input 2=00000000000000000000000000000001,write Data: 00000000000000000000000000000001
#
# time: 10,pc=00001 ,ins: 0101000101000010
# ,read data 1=00000000000000000000000000000000,read data 2=00000000000000000000000000000000, alu input 2=00000000000000000000000000000000,write Data: 00000000000000000000000000000000
#
# time: 30,pc=00100 ,ins: 0000010001100001
# ,read data 1=00000000000000000000000000000010,read data 2=00000000000000000000000000000001, alu input 2=00000000000000000000000000000001,write Data: 00000000000000000000000000000011
#
# time: 50,pc=00101 ,ins: 0000101011100001
# ,read data 1=00000000000000000000000000000000,read data 2=00000000000000000000000000000011, alu input 2=00000000000000000000000000000011,write Data: 00000000000000000000000000000011
#
```

R2 BEQ R1 >FALSE

R0 BEQ R5 >TRUE

You can see the program counter increment!!

```
0101010001000010
0101000101000010        <
0000010001100000        |
0000000101100000        v
0000010001100001        <
0000101011100001
```

```
# time: 170,pc=01011 ,ins: 0110000101000010
# ,read data 1=00000000000000000000000000000000,read data 2=00000000000000000000000000000000, alu input 2=00000000000000000000000000000000,write Data: 00000000000000000000000000000000
#
# time: 190,pc=01100 ,ins: 0110000001000010
# ,read data 1=00000000000000000000000000000000,read data 2=00000000000000000000000000000001, alu input 2=00000000000000000000000000000001,write Data: 11111111111111111111111111111111
#
# time: 210,pc=01111 ,ins: 0000101011100101
# ,read data 1=00000000000000000000000000000000,read data 2=00000000000000000000000000000011, alu input 2=00000000000000000000000000000011,write Data: 00000000000000000000000000000011
#
# time: 230,pc=10000 ,ins: 0001010100100101
# ,read data 1=00000000000000000000000000000010,read data 2=00000000000000000000000000000011, alu input 2=00000000000000000000000000100101,write Data: 00000000000000000000000000100111
#
# time: 250,pc=10001 ,ins: 0001101100100101
# ,read data 1=00000000000000000000000000000000,read data 2=00000000000000000000000000100111, alu input 2=00000000000000000000000000100101,write Data: 00000000000000000000000000100101
#
```

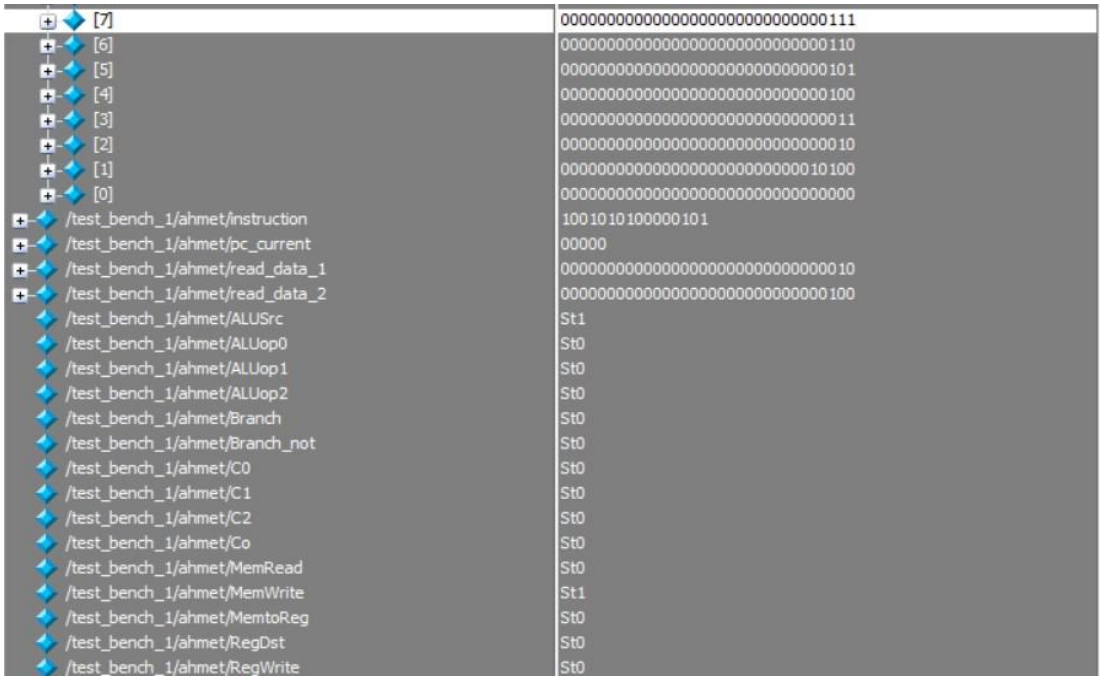
R0 BNE R5 >FALSE

R2 BE R1 >TRUE

```
0110000101000010
0110000001000010        <
0000101011100100        |
0000010001100101        v
0000101011100101        <
0001010100100101
0001101100100101
```

TEST SW

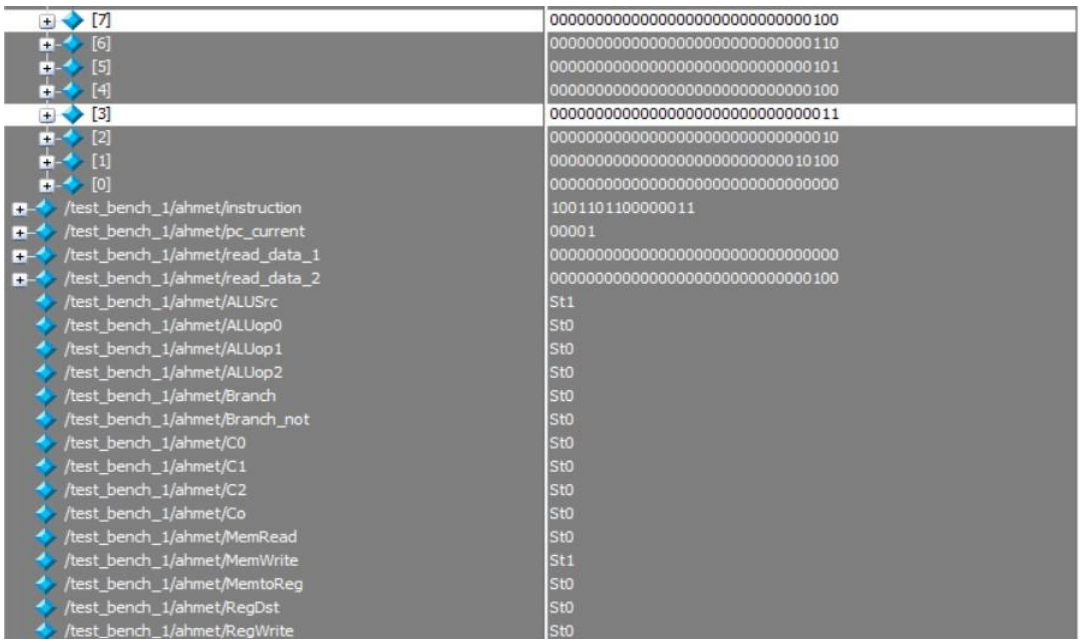
First sw test



[7]	00000000000000000000000000000111
[6]	00000000000000000000000000000110
[5]	00000000000000000000000000000101
[4]	00000000000000000000000000000100
[3]	00000000000000000000000000000011
[2]	00000000000000000000000000000010
[1]	00000000000000000000000000010100
[0]	00000000000000000000000000000000
/test_bench_1/ahmet/instruction	10010101000001 01
/test_bench_1/ahmet/pc_current	00000
/test_bench_1/ahmet/read_data_1	00000000000000000000000000000010
/test_bench_1/ahmet/read_data_2	00000000000000000000000000000100
/test_bench_1/ahmet/ALUSrc	St1
/test_bench_1/ahmet/ALUop0	St0
/test_bench_1/ahmet/ALUop1	St0
/test_bench_1/ahmet/ALUop2	St0
/test_bench_1/ahmet/Branch	St0
/test_bench_1/ahmet/Branch_not	St0
/test_bench_1/ahmet/C0	St0
/test_bench_1/ahmet/C1	St0
/test_bench_1/ahmet/C2	St0
/test_bench_1/ahmet/Co	St0
/test_bench_1/ahmet/MemRead	St0
/test_bench_1/ahmet/MemWrite	St1
/test_bench_1/ahmet/MemtoReg	St0
/test_bench_1/ahmet/RegDst	St0
/test_bench_1/ahmet/RegWrite	St0

Before memory content first sw

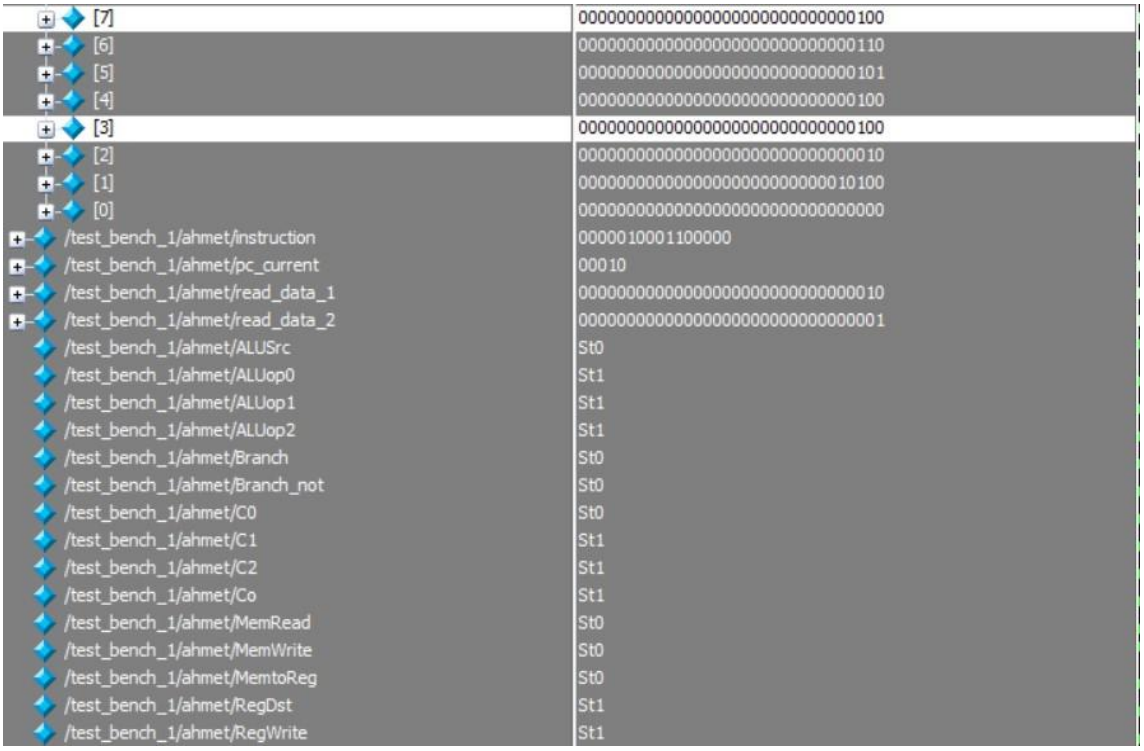
First sw result and second sw test



[7]	00000000000000000000000000000100
[6]	00000000000000000000000000000110
[5]	00000000000000000000000000000101
[4]	00000000000000000000000000000100
[3]	00000000000000000000000000000011
[2]	00000000000000000000000000000010
[1]	00000000000000000000000000010100
[0]	00000000000000000000000000000000
/test_bench_1/ahmet/instruction	10011011000000 11
/test_bench_1/ahmet/pc_current	00001
/test_bench_1/ahmet/read_data_1	00000000000000000000000000000000
/test_bench_1/ahmet/read_data_2	00000000000000000000000000000100
/test_bench_1/ahmet/ALUSrc	St1
/test_bench_1/ahmet/ALUop0	St0
/test_bench_1/ahmet/ALUop1	St0
/test_bench_1/ahmet/ALUop2	St0
/test_bench_1/ahmet/Branch	St0
/test_bench_1/ahmet/Branch_not	St0
/test_bench_1/ahmet/C0	St0
/test_bench_1/ahmet/C1	St0
/test_bench_1/ahmet/C2	St0
/test_bench_1/ahmet/Co	St0
/test_bench_1/ahmet/MemRead	St0
/test_bench_1/ahmet/MemWrite	St1
/test_bench_1/ahmet/MemtoReg	St0
/test_bench_1/ahmet/RegDst	St0
/test_bench_1/ahmet/RegWrite	St0

After memory content first sw

Before memory content second sw



After memory content second sw