FALL 2021 CSE331/503

HW4

1901042262

UĞUR ER

1-)Determination of alu control and control unit values

Instr	ALUop	Opcode	Func	ALUctr
AND	001	0000	000	110
ADD	001	0000	001	000
SUB	001	0000	010	010
XOR	001	0000	011	001
NOR	001	0000	100	101
OR	001	0000	101	111
ADDI	000	0001	XXX	000
ANDI	110	0010	XXX	110
ORI	111	0011	XXX	111
NORI	101	0100	XXX	101
BEQ	010	0101	XXX	010
BNE	010	0110	XXX	010
SLTI	100	0111	XXX	100
LW	000	1000	XXX	000
SW	000	1001	XXX	000

Note: Those who do the same alu process are the same color.

Control Unit Values:

Rtype instruction ALUop=001

For I type instructions, I set the aluctr values to aluop.

Alu Control Unit Values:

Aluctr<2>

```
and(r1,notF0,notF1,notF2); //000 condition
or(r2,r1,func[2]);           // 1xx or 000 condition
and(r3,r2,isRtype); // rtype
and(r4,notRtype,AluOp[2]);// if notRtype and AluOp[2]==1
or(Aluctr[2],r3,r4);
```

Aluctr<1>

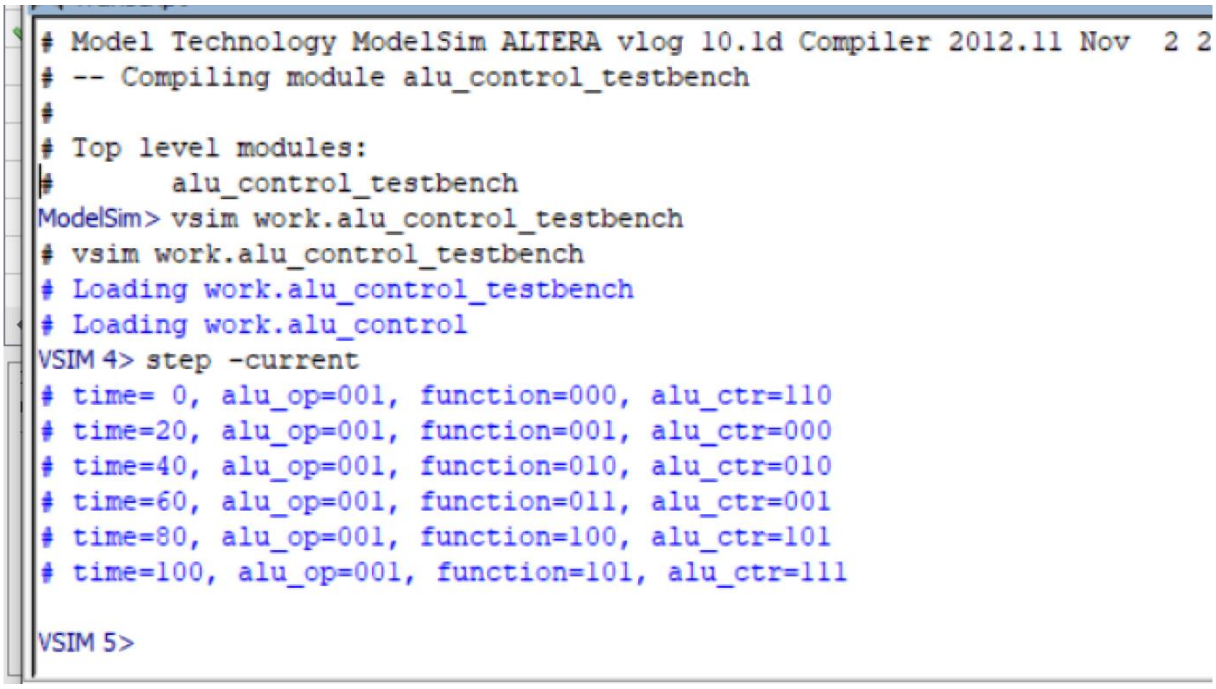
```
and(r5,notF0,notF2);  //0x0 condition
and(r6,func[0],func[2]); //1x1 cond
or(r7,r5,r6); //1x1 or 0x0
and(r8,r7,isRtype); // rtype
and(r9,notRtype,AluOp[1]); // if notRtype and AluOp[1]==1

or(Aluctr[1],r9,r8);
```

Aluctr<0>

```
and(r10,func[0],func[1]); // x11 cond

or(r11,r10,func[2]); // x11 or 1xx

and(r12,r11,isRtype); // rtype

and(r13,notRtype,AluOp[0]);// if notRtype and AluOp[0]==1

or(Aluctr[0],r13,r12);
```

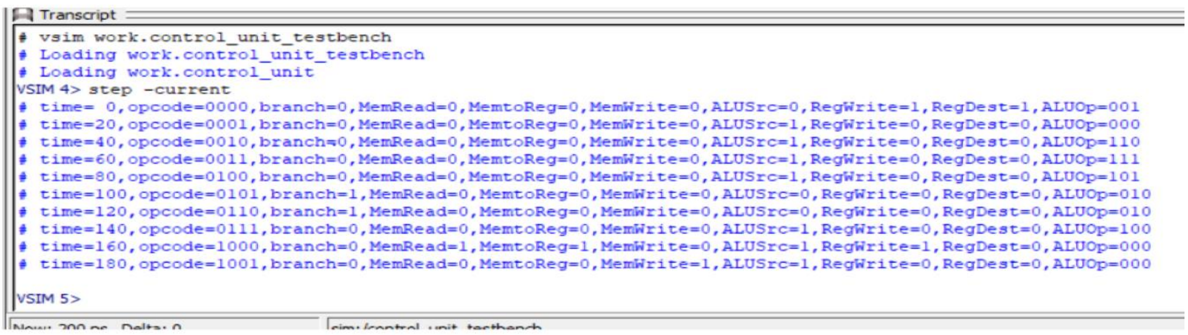
2-) Tests

1. Alu_control_testbench:



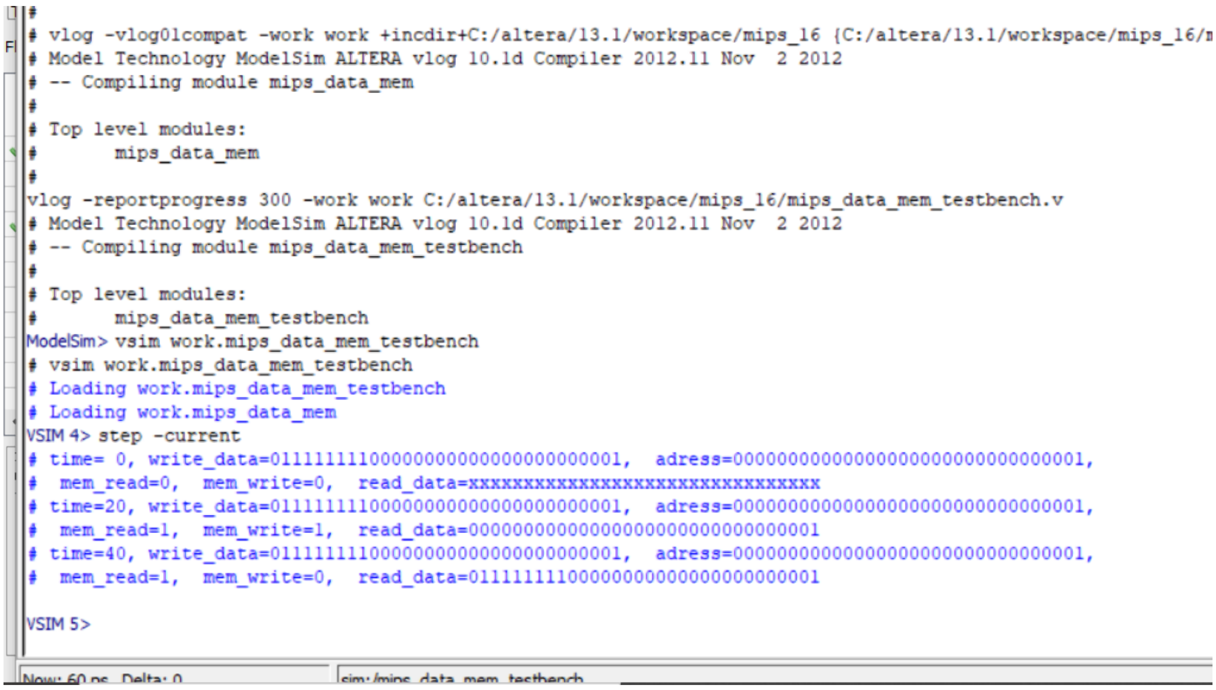
```
# Model Technology ModelSim ALTERA vlog 10.1d Compiler 2012.11 Nov  2 2
# -- Compiling module alu_control_testbench
#
# Top level modules:
#          alu_control_testbench
ModelSim> vsim work.alu_control_testbench
# vsim work.alu_control_testbench
# Loading work.alu_control_testbench
# Loading work.alu_control
VSIM 4> step -current
# time= 0, alu_op=001, function=000, alu_ctr=110
# time=20, alu_op=001, function=001, alu_ctr=000
# time=40, alu_op=001, function=010, alu_ctr=010
# time=60, alu_op=001, function=011, alu_ctr=001
# time=80, alu_op=001, function=100, alu_ctr=101
# time=100, alu_op=001, function=101, alu_ctr=111

VSIM 5>
```

2. Control_unit_testbench:



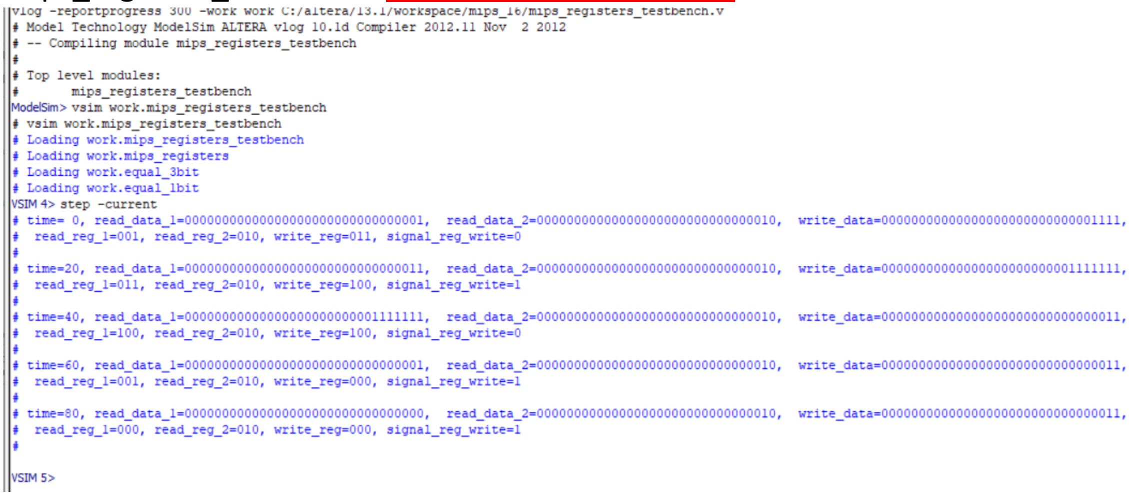
```
# vsim work.control_unit_testbench
# Loading work.control_unit_testbench
# Loading work.control_unit
VSIM 4> step -current
# time= 0,opcode=0000,branch=0,MemRead=0,MemtoReg=0,MemWrite=0,ALUSrc=0,RegWrite=1,RegDest=1,ALUOp=001
# time=20,opcode=0001,branch=0,MemRead=0,MemtoReg=0,MemWrite=0,ALUSrc=1,RegWrite=0,RegDest=0,ALUOp=000
# time=40,opcode=0010,branch=0,MemRead=0,MemtoReg=0,MemWrite=0,ALUSrc=1,RegWrite=0,RegDest=0,ALUOp=110
# time=60,opcode=0011,branch=0,MemRead=0,MemtoReg=0,MemWrite=0,ALUSrc=1,RegWrite=0,RegDest=0,ALUOp=111
# time=80,opcode=0100,branch=0,MemRead=0,MemtoReg=0,MemWrite=0,ALUSrc=1,RegWrite=0,RegDest=0,ALUOp=101
# time=100,opcode=0101,branch=1,MemRead=0,MemtoReg=0,MemWrite=0,ALUSrc=0,RegWrite=0,RegDest=0,ALUOp=010
# time=120,opcode=0110,branch=1,MemRead=0,MemtoReg=0,MemWrite=0,ALUSrc=0,RegWrite=0,RegDest=0,ALUOp=010
# time=140,opcode=0111,branch=0,MemRead=0,MemtoReg=0,MemWrite=0,ALUSrc=1,RegWrite=0,RegDest=0,ALUOp=100
# time=160,opcode=1000,branch=0,MemRead=1,MemtoReg=1,MemWrite=0,ALUSrc=1,RegWrite=1,RegDest=0,ALUOp=000
# time=180,opcode=1001,branch=0,MemRead=0,MemtoReg=0,MemWrite=1,ALUSrc=1,RegWrite=0,RegDest=0,ALUOp=000

VSIM 5>
```

3. Mips_data_mem_testbench:



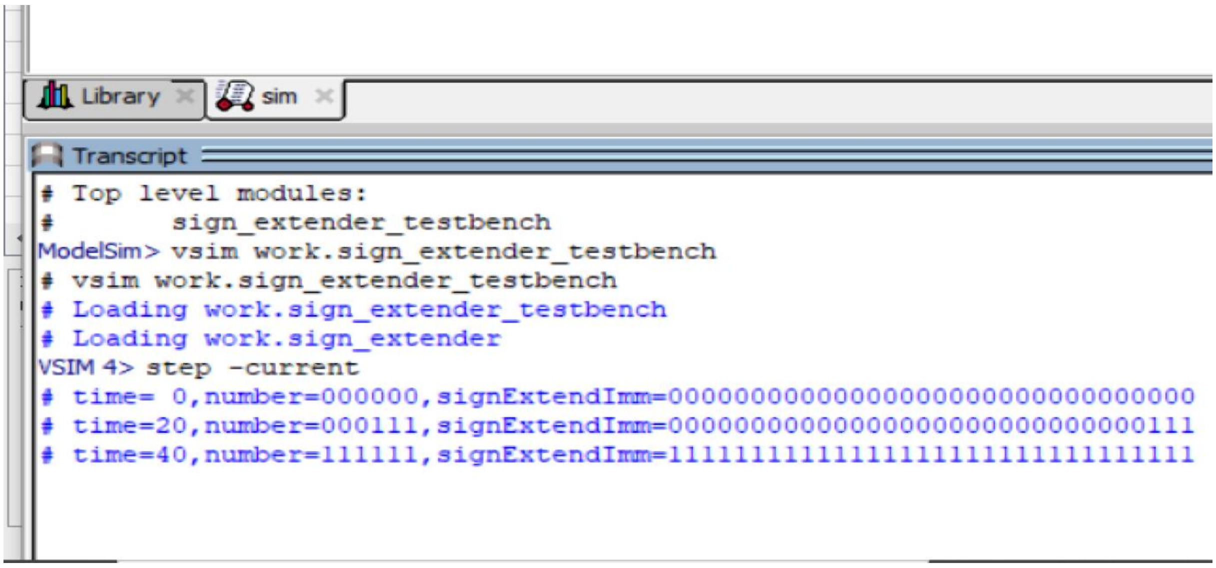
```
#
# vlog -vlog01compat -work work +incdir+C:/altera/13.1/workspace/mips_16 {C:/altera/13.1/workspace/mips_16/m
# Model Technology ModelSim ALTERA vlog 10.1d Compiler 2012.11 Nov  2 2012
# -- Compiling module mips_data_mem
#
# Top level modules:
#       mips_data_mem
#
vlog -reportprogress 300 -work work C:/altera/13.1/workspace/mips_16/mips_data_mem_testbench.v
# Model Technology ModelSim ALTERA vlog 10.1d Compiler 2012.11 Nov  2 2012
# -- Compiling module mips_data_mem_testbench
#
# Top level modules:
#       mips_data_mem_testbench
ModelSim> vsim work.mips_data_mem_testbench
# vsim work.mips_data_mem_testbench
# Loading work.mips_data_mem_testbench
# Loading work.mips_data_mem
VSIM 4> step -current
# time= 0, write_data=01111111110000000000000000000001,  adress=00000000000000000000000000000001,
#  mem_read=0,  mem_write=0,  read_data=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
# time=20, write_data=01111111110000000000000000000001,  adress=00000000000000000000000000000001,
#  mem_read=1,  mem_write=1,  read_data=00000000000000000000000000000001
# time=40, write_data=01111111110000000000000000000001,  adress=00000000000000000000000000000001,
#  mem_read=1,  mem_write=0,  read_data=01111111110000000000000000000001

VSIM 5>

Now: 60 ns  Delta: 0          sim:/mips_data_mem_testbench
```

4. Mips_registers_testbench: The r0 always remains 0.



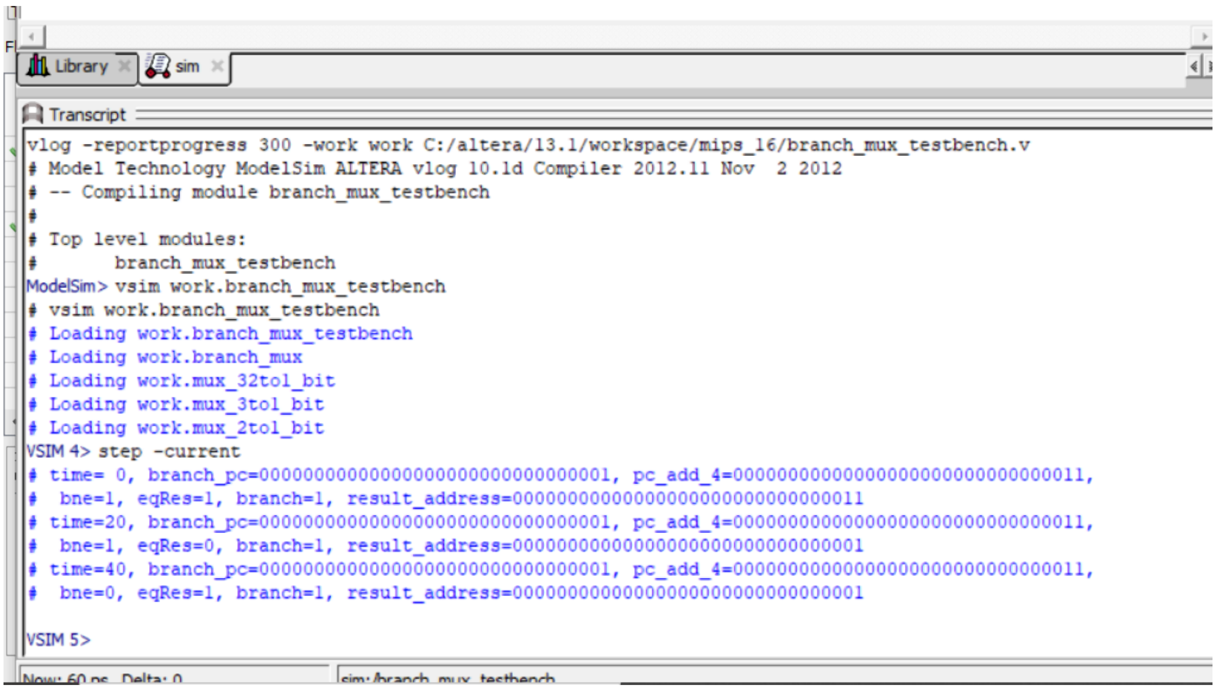
```
vlog -reportprogress 300 -work work C:/altera/13.1/workspace/mips_16/mips_registers_testbench.v
# Model Technology ModelSim ALTERA vlog 10.1d Compiler 2012.11 Nov  2 2012
# -- Compiling module mips_registers_testbench
#
# Top level modules:
#       mips_registers_testbench
ModelSim> vsim work.mips_registers_testbench
# vsim work.mips_registers_testbench
# Loading work.mips_registers_testbench
# Loading work.mips_registers
# Loading work.equal_3bit
# Loading work.equal_1bit
VSIM 4> step -current
# time= 0, read_data_1=00000000000000000000000000000001,  read_data_2=00000000000000000000000000000010,  write_data=00000000000000000000000000001111,
#  read_reg_1=001, read_reg_2=010, write_reg=011, signal_reg_write=0
#
# time=20, read_data_1=00000000000000000000000000000011,  read_data_2=00000000000000000000000000000010,  write_data=00000000000000000000000001111111,
#  read_reg_1=011, read_reg_2=010, write_reg=100, signal_reg_write=1
#
# time=40, read_data_1=00000000000000000000000001111111,  read_data_2=00000000000000000000000000000010,  write_data=00000000000000000000000000000011,
#  read_reg_1=100, read_reg_2=010, write_reg=100, signal_reg_write=0
#
# time=60, read_data_1=00000000000000000000000000000001,  read_data_2=00000000000000000000000000000010,  write_data=00000000000000000000000000000011,
#  read_reg_1=001, read_reg_2=010, write_reg=000, signal_reg_write=1
#
# time=80, read_data_1=00000000000000000000000000000000,  read_data_2=00000000000000000000000000000010,  write_data=00000000000000000000000000000011,
#  read_reg_1=000, read_reg_2=010, write_reg=000, signal_reg_write=1
#

VSIM 5>
```

5. Sign_extender_testbench:



```
# Top level modules:
#       sign_extender_testbench
ModelSim> vsim work.sign_extender_testbench
# vsim work.sign_extender_testbench
# Loading work.sign_extender_testbench
# Loading work.sign_extender
VSIM 4> step -current
# time= 0,number=000000,signExtendImm=00000000000000000000000000000000
# time=20,number=000111,signExtendImm=00000000000000000000000000000111
# time=40,number=111111,signExtendImm=11111111111111111111111111111111
```

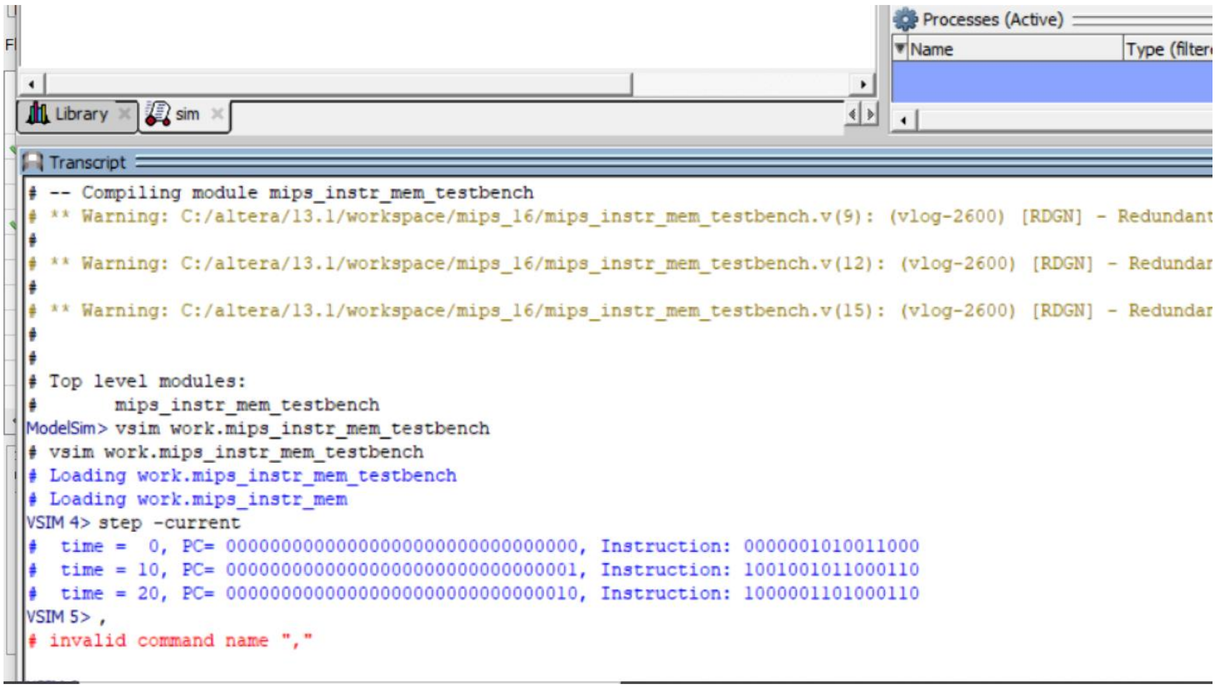
6. Branch_mux_testbench:



```
vlog -reportprogress 300 -work work C:/altera/13.1/workspace/mips_16/branch_mux_testbench.v
# Model Technology ModelSim ALTERA vlog 10.1d Compiler 2012.11 Nov  2 2012
# -- Compiling module branch_mux_testbench
#
# Top level modules:
#       branch_mux_testbench
ModelSim> vsim work.branch_mux_testbench
# vsim work.branch_mux_testbench
# Loading work.branch_mux_testbench
# Loading work.branch_mux
# Loading work.mux_32to1_bit
# Loading work.mux_3to1_bit
# Loading work.mux_2to1_bit
VSIM 4> step -current
# time= 0, branch_pc=00000000000000000000000000000001, pc_add_4=00000000000000000000000000000011,
#  bne=1, eqRes=1, branch=1, result_address=00000000000000000000000000000011
# time=20, branch_pc=00000000000000000000000000000001, pc_add_4=00000000000000000000000000000011,
#  bne=1, eqRes=0, branch=1, result_address=00000000000000000000000000000001
# time=40, branch_pc=00000000000000000000000000000001, pc_add_4=00000000000000000000000000000011,
#  bne=0, eqRes=1, branch=1, result_address=00000000000000000000000000000001

VSIM 5>
```

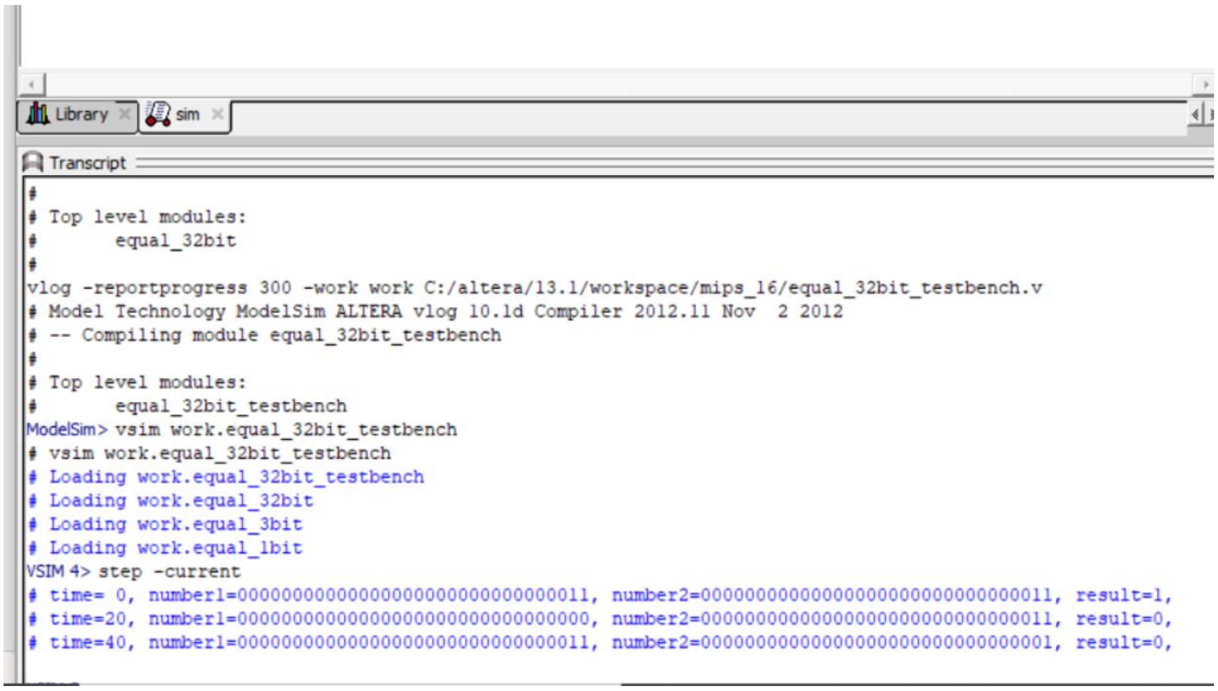
Now: 60 ns Delta: 0   sim:/branch_mux_testbench

7. Mips_instr_mem_testbench:



```
# -- Compiling module mips_instr_mem_testbench
# ** Warning: C:/altera/13.1/workspace/mips_16/mips_instr_mem_testbench.v(9): (vlog-2600) [RDGN] - Redundant
#
# ** Warning: C:/altera/13.1/workspace/mips_16/mips_instr_mem_testbench.v(12): (vlog-2600) [RDGN] - Redundar
#
# ** Warning: C:/altera/13.1/workspace/mips_16/mips_instr_mem_testbench.v(15): (vlog-2600) [RDGN] - Redundar
#
#
# Top level modules:
#       mips_instr_mem_testbench
ModelSim> vsim work.mips_instr_mem_testbench
# vsim work.mips_instr_mem_testbench
# Loading work.mips_instr_mem_testbench
# Loading work.mips_instr_mem
VSIM 4> step -current
#  time =  0, PC= 00000000000000000000000000000000, Instruction: 0000001010011000
#  time = 10, PC= 00000000000000000000000000000001, Instruction: 1001001011000110
#  time = 20, PC= 00000000000000000000000000000010, Instruction: 1000001101000110
VSIM 5> ,
# invalid command name ","
```

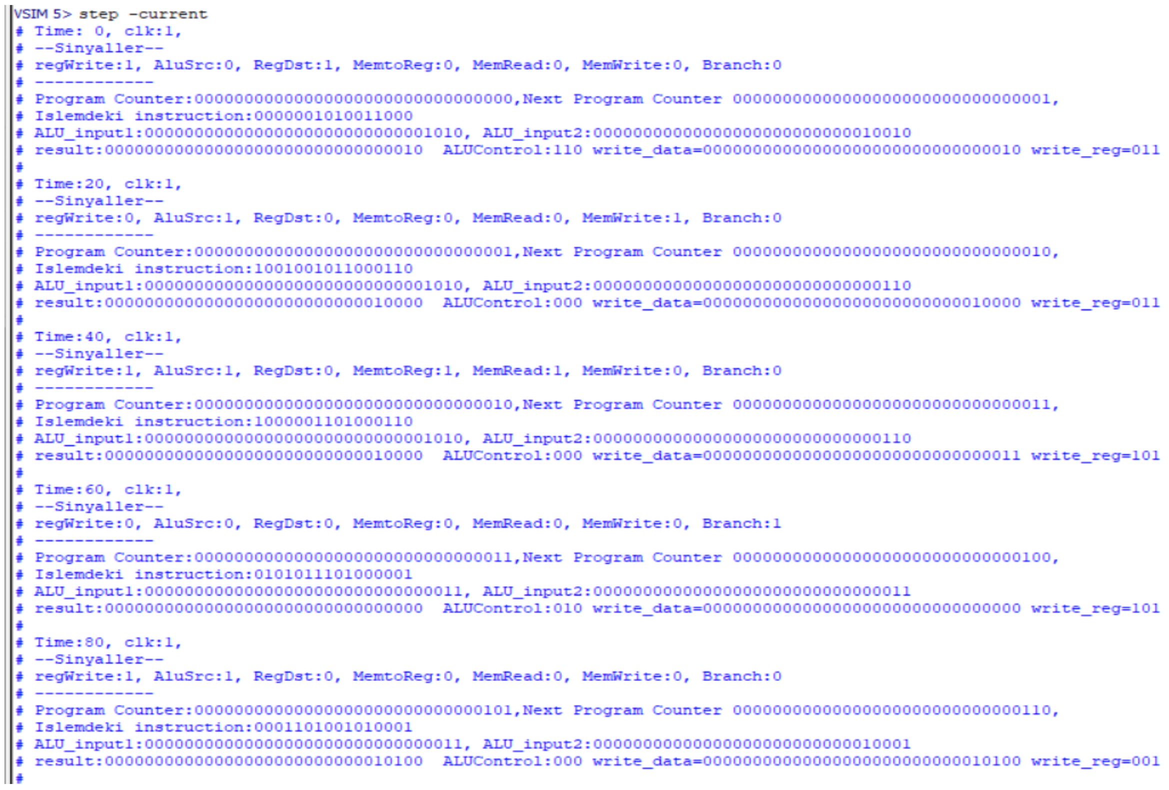
8. Equal_32bit_testbench:



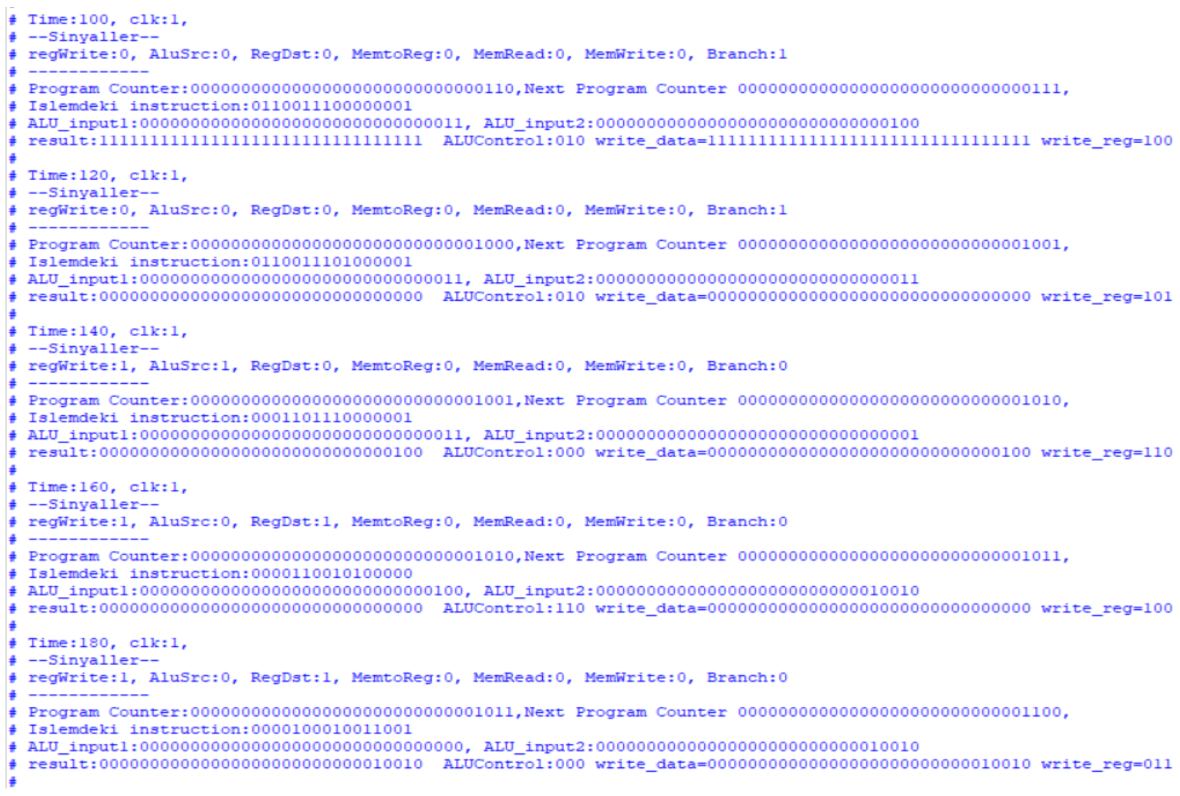
```
#
# Top level modules:
#       equal_32bit
#
vlog -reportprogress 300 -work work C:/altera/13.1/workspace/mips_16/equal_32bit_testbench.v
# Model Technology ModelSim ALTERA vlog 10.1d Compiler 2012.11 Nov  2 2012
# -- Compiling module equal_32bit_testbench
#
# Top level modules:
#       equal_32bit_testbench
ModelSim> vsim work.equal_32bit_testbench
# vsim work.equal_32bit_testbench
# Loading work.equal_32bit_testbench
# Loading work.equal_32bit
# Loading work.equal_3bit
# Loading work.equal_1bit
VSIM 4> step -current
# time= 0, number1=00000000000000000000000000000011, number2=00000000000000000000000000000011, result=1,
# time=20, number1=00000000000000000000000000000000, number2=00000000000000000000000000000011, result=0,
# time=40, number1=00000000000000000000000000000011, number2=00000000000000000000000000000001, result=0,
```

Not: There are other testbenches, but I only put the tests of those directly related to mips.

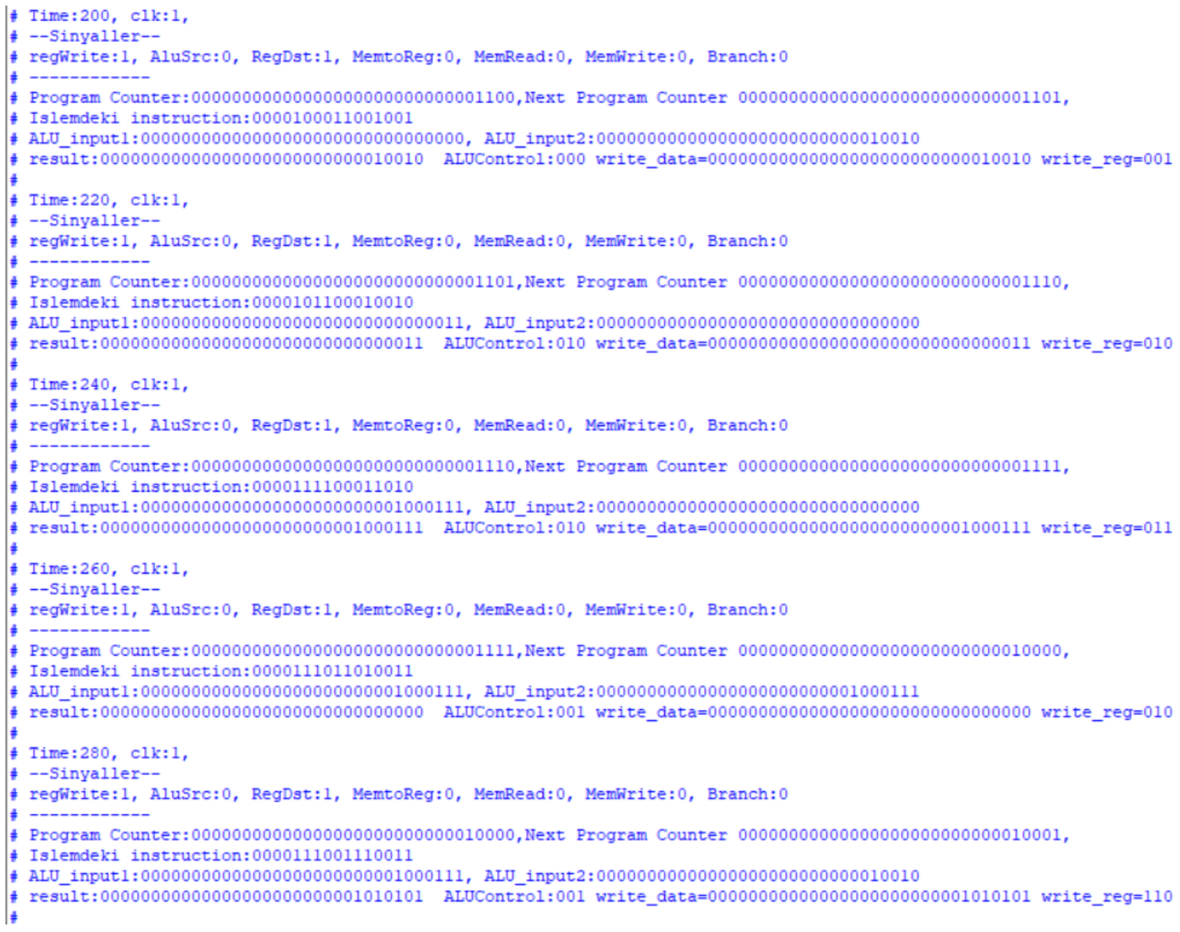
9. Mips_testbench: All the instructions have been tried at least 2 times.



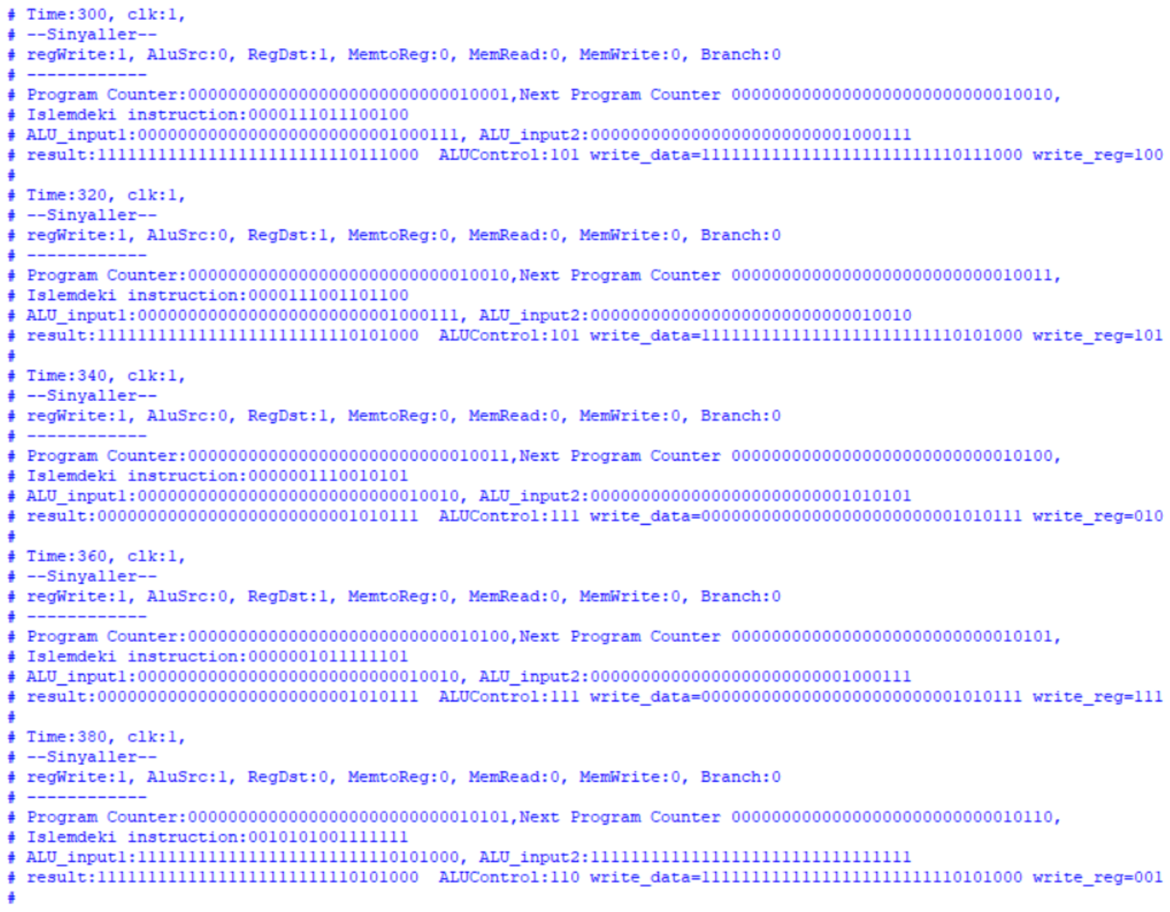
```
VSIM 5> step -current
# Time: 0, clk:1,
# --Sinyaller--
# regWrite:1, AluSrc:0, RegDst:1, MemtoReg:0, MemRead:0, MemWrite:0, Branch:0
# ------------
# Program Counter:00000000000000000000000000000000,Next Program Counter 00000000000000000000000000000001,
# Islemdeki instruction:0000001010011000
# ALU_input1:00000000000000000000000000001010, ALU_input2:00000000000000000000000000010010
# result:00000000000000000000000000000010  ALUControl:110 write_data=00000000000000000000000000000010 write_reg=011
#
# Time:20, clk:1,
# --Sinyaller--
# regWrite:0, AluSrc:1, RegDst:0, MemtoReg:0, MemRead:0, MemWrite:1, Branch:0
# ------------
# Program Counter:00000000000000000000000000000001,Next Program Counter 00000000000000000000000000000010,
# Islemdeki instruction:1001001011000110
# ALU_input1:00000000000000000000000000001010, ALU_input2:00000000000000000000000000000110
# result:00000000000000000000000000010000  ALUControl:000 write_data=00000000000000000000000000010000 write_reg=011
#
# Time:40, clk:1,
# --Sinyaller--
# regWrite:1, AluSrc:1, RegDst:0, MemtoReg:1, MemRead:1, MemWrite:0, Branch:0
# ------------
# Program Counter:00000000000000000000000000000010,Next Program Counter 00000000000000000000000000000011,
# Islemdeki instruction:1000001101000110
# ALU_input1:00000000000000000000000000001010, ALU_input2:00000000000000000000000000000110
# result:00000000000000000000000000010000  ALUControl:000 write_data=00000000000000000000000000000011 write_reg=101
#
# Time:60, clk:1,
# --Sinyaller--
# regWrite:0, AluSrc:0, RegDst:0, MemtoReg:0, MemRead:0, MemWrite:0, Branch:1
# ------------
# Program Counter:00000000000000000000000000000011,Next Program Counter 00000000000000000000000000000100,
# Islemdeki instruction:0101011101000001
# ALU_input1:00000000000000000000000000000011, ALU_input2:00000000000000000000000000000011
# result:00000000000000000000000000000000  ALUControl:010 write_data=00000000000000000000000000000000 write_reg=101
#
# Time:80, clk:1,
# --Sinyaller--
# regWrite:1, AluSrc:1, RegDst:0, MemtoReg:0, MemRead:0, MemWrite:0, Branch:0
# ------------
# Program Counter:00000000000000000000000000000101,Next Program Counter 00000000000000000000000000000110,
# Islemdeki instruction:0001101001010001
# ALU_input1:00000000000000000000000000000011, ALU_input2:00000000000000000000000000010001
# result:00000000000000000000000000010100  ALUControl:000 write_data=00000000000000000000000000010100 write_reg=001
#
```



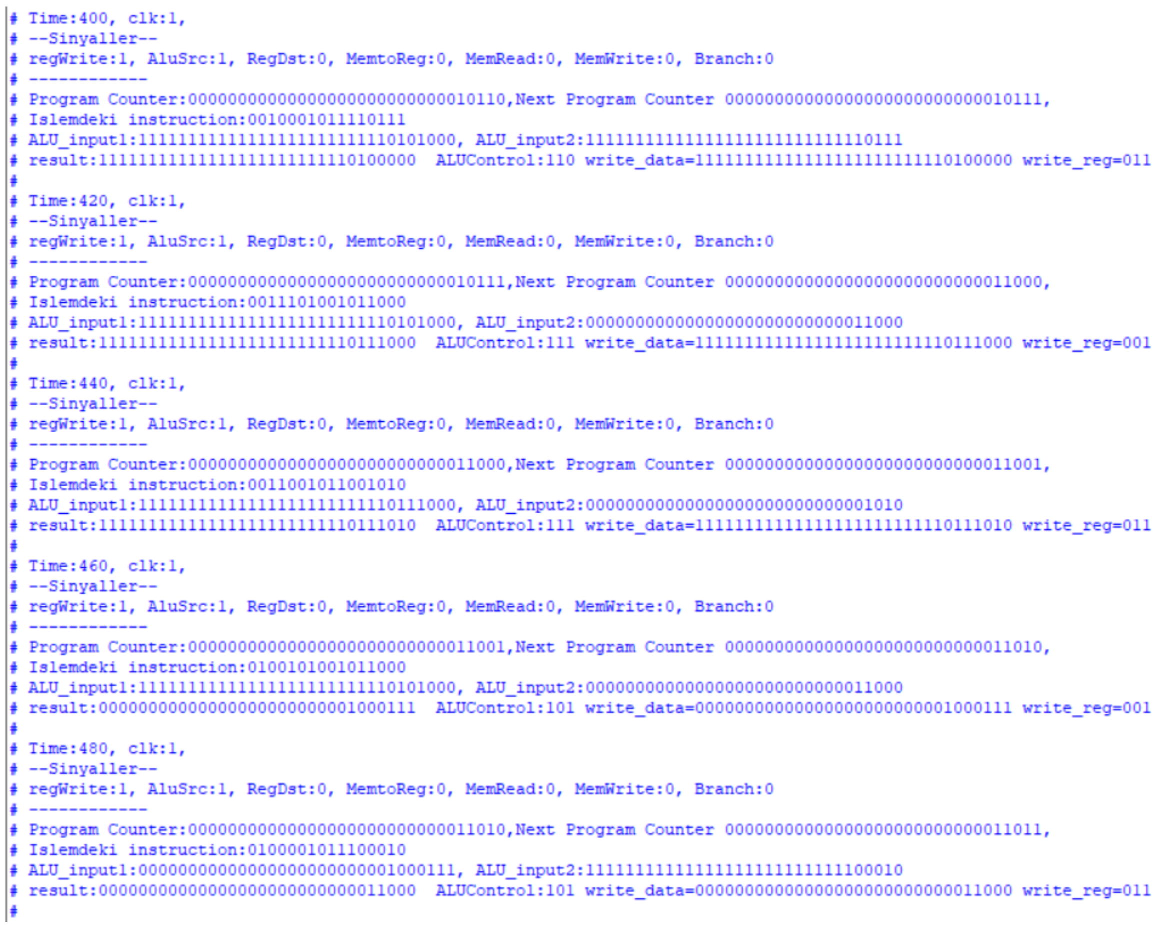
```
# Time:100, clk:1,
# --Sinyaller--
# regWrite:0, AluSrc:0, RegDst:0, MemtoReg:0, MemRead:0, MemWrite:0, Branch:1
# ------------
# Program Counter:00000000000000000000000000000110,Next Program Counter 00000000000000000000000000000111,
# Islemdeki instruction:0110011100000001
# ALU_input1:00000000000000000000000000000011, ALU_input2:00000000000000000000000000000100
# result:11111111111111111111111111111111  ALUControl:010 write_data=11111111111111111111111111111111 write_reg=100
#
# Time:120, clk:1,
# --Sinyaller--
# regWrite:0, AluSrc:0, RegDst:0, MemtoReg:0, MemRead:0, MemWrite:0, Branch:1
# ------------
# Program Counter:00000000000000000000000000001000,Next Program Counter 00000000000000000000000000001001,
# Islemdeki instruction:0110011101000001
# ALU_input1:00000000000000000000000000000011, ALU_input2:00000000000000000000000000000011
# result:00000000000000000000000000000000  ALUControl:010 write_data=00000000000000000000000000000000 write_reg=101
#
# Time:140, clk:1,
# --Sinyaller--
# regWrite:1, AluSrc:1, RegDst:0, MemtoReg:0, MemRead:0, MemWrite:0, Branch:0
# ------------
# Program Counter:00000000000000000000000000001001,Next Program Counter 00000000000000000000000000001010,
# Islemdeki instruction:0001101110000001
# ALU_input1:00000000000000000000000000000011, ALU_input2:00000000000000000000000000000001
# result:00000000000000000000000000000100  ALUControl:000 write_data=00000000000000000000000000000100 write_reg=110
#
# Time:160, clk:1,
# --Sinyaller--
# regWrite:1, AluSrc:0, RegDst:1, MemtoReg:0, MemRead:0, MemWrite:0, Branch:0
# ------------
# Program Counter:00000000000000000000000000001010,Next Program Counter 00000000000000000000000000001011,
# Islemdeki instruction:0000110010100000
# ALU_input1:00000000000000000000000000000100, ALU_input2:00000000000000000000000000010010
# result:00000000000000000000000000000000  ALUControl:110 write_data=00000000000000000000000000000000 write_reg=100
#
# Time:180, clk:1,
# --Sinyaller--
# regWrite:1, AluSrc:0, RegDst:1, MemtoReg:0, MemRead:0, MemWrite:0, Branch:0
# ------------
# Program Counter:00000000000000000000000000001011,Next Program Counter 00000000000000000000000000001100,
# Islemdeki instruction:0000100010011001
# ALU_input1:00000000000000000000000000000000, ALU_input2:00000000000000000000000000010010
# result:00000000000000000000000000010010  ALUControl:000 write_data=00000000000000000000000000010010 write_reg=011
#
```



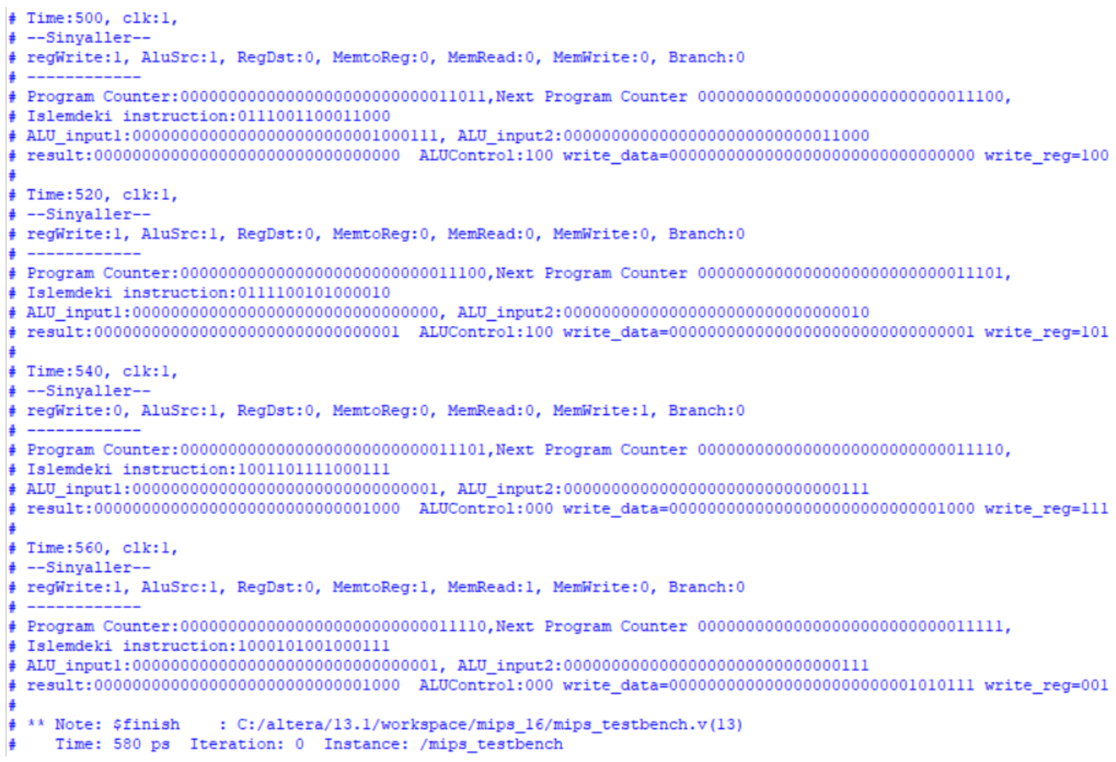
```
# Time:200, clk:1,
# --Sinyaller--
# regWrite:1, AluSrc:0, RegDst:1, MemtoReg:0, MemRead:0, MemWrite:0, Branch:0
# ------------
# Program Counter:00000000000000000000000000001100,Next Program Counter 00000000000000000000000000001101,
# Islemdeki instruction:0000100011001001
# ALU_input1:00000000000000000000000000000000, ALU_input2:00000000000000000000000000010010
# result:00000000000000000000000000010010  ALUControl:000 write_data=00000000000000000000000000010010 write_reg=001
#
# Time:220, clk:1,
# --Sinyaller--
# regWrite:1, AluSrc:0, RegDst:1, MemtoReg:0, MemRead:0, MemWrite:0, Branch:0
# ------------
# Program Counter:00000000000000000000000000001101,Next Program Counter 00000000000000000000000000001110,
# Islemdeki instruction:0000101100010010
# ALU_input1:00000000000000000000000000000011, ALU_input2:00000000000000000000000000000000
# result:00000000000000000000000000000011  ALUControl:010 write_data=00000000000000000000000000000011 write_reg=010
#
# Time:240, clk:1,
# --Sinyaller--
# regWrite:1, AluSrc:0, RegDst:1, MemtoReg:0, MemRead:0, MemWrite:0, Branch:0
# ------------
# Program Counter:00000000000000000000000000001110,Next Program Counter 00000000000000000000000000001111,
# Islemdeki instruction:0000111100011010
# ALU_input1:00000000000000000000000001000111, ALU_input2:00000000000000000000000000000000
# result:00000000000000000000000001000111  ALUControl:010 write_data=00000000000000000000000001000111 write_reg=011
#
# Time:260, clk:1,
# --Sinyaller--
# regWrite:1, AluSrc:0, RegDst:1, MemtoReg:0, MemRead:0, MemWrite:0, Branch:0
# ------------
# Program Counter:00000000000000000000000000001111,Next Program Counter 00000000000000000000000000010000,
# Islemdeki instruction:0000111011010011
# ALU_input1:00000000000000000000000001000111, ALU_input2:00000000000000000000000001000111
# result:00000000000000000000000000000000  ALUControl:001 write_data=00000000000000000000000000000000 write_reg=010
#
# Time:280, clk:1,
# --Sinyaller--
# regWrite:1, AluSrc:0, RegDst:1, MemtoReg:0, MemRead:0, MemWrite:0, Branch:0
# ------------
# Program Counter:00000000000000000000000000010000,Next Program Counter 00000000000000000000000000010001,
# Islemdeki instruction:0000111001110011
# ALU_input1:00000000000000000000000001000111, ALU_input2:00000000000000000000000000010010
# result:00000000000000000000000001010101  ALUControl:001 write_data=00000000000000000000000001010101 write_reg=110
#
```



```
# Time:300, clk:1,
# --Sinyaller--
# regWrite:1, AluSrc:0, RegDst:1, MemtoReg:0, MemRead:0, MemWrite:0, Branch:0
# ------------
# Program Counter:00000000000000000000000000010001,Next Program Counter 00000000000000000000000000010010,
# Islemdeki instruction:0000111011100100
# ALU_input1:00000000000000000000000001000111, ALU_input2:00000000000000000000000001000111
# result:11111111111111111111111110111000  ALUControl:101 write_data=11111111111111111111111110111000 write_reg=100
#
# Time:320, clk:1,
# --Sinyaller--
# regWrite:1, AluSrc:0, RegDst:1, MemtoReg:0, MemRead:0, MemWrite:0, Branch:0
# ------------
# Program Counter:00000000000000000000000000010010,Next Program Counter 00000000000000000000000000010011,
# Islemdeki instruction:0000111001101100
# ALU_input1:00000000000000000000000001000111, ALU_input2:00000000000000000000000000010010
# result:11111111111111111111111110101000  ALUControl:101 write_data=11111111111111111111111110101000 write_reg=101
#
# Time:340, clk:1,
# --Sinyaller--
# regWrite:1, AluSrc:0, RegDst:1, MemtoReg:0, MemRead:0, MemWrite:0, Branch:0
# ------------
# Program Counter:00000000000000000000000000010011,Next Program Counter 00000000000000000000000000010100,
# Islemdeki instruction:0000001110010101
# ALU_input1:00000000000000000000000000010010, ALU_input2:00000000000000000000000001010101
# result:00000000000000000000000001010111  ALUControl:111 write_data=00000000000000000000000001010111 write_reg=010
#
# Time:360, clk:1,
# --Sinyaller--
# regWrite:1, AluSrc:0, RegDst:1, MemtoReg:0, MemRead:0, MemWrite:0, Branch:0
# ------------
# Program Counter:00000000000000000000000000010100,Next Program Counter 00000000000000000000000000010101,
# Islemdeki instruction:0000001011111101
# ALU_input1:00000000000000000000000000010010, ALU_input2:00000000000000000000000001000111
# result:00000000000000000000000001010111  ALUControl:111 write_data=00000000000000000000000001010111 write_reg=111
#
# Time:380, clk:1,
# --Sinyaller--
# regWrite:1, AluSrc:1, RegDst:0, MemtoReg:0, MemRead:0, MemWrite:0, Branch:0
# ------------
# Program Counter:00000000000000000000000000010101,Next Program Counter 00000000000000000000000000010110,
# Islemdeki instruction:0010101001111111
# ALU_input1:11111111111111111111111110101000, ALU_input2:11111111111111111111111111111111
# result:11111111111111111111111110101000  ALUControl:110 write_data=11111111111111111111111110101000 write_reg=001
#
```



```
# Time:400, clk:1,
# --Sinyaller--
# regWrite:1, AluSrc:1, RegDst:0, MemtoReg:0, MemRead:0, MemWrite:0, Branch:0
# ------------
# Program Counter:00000000000000000000000000010110,Next Program Counter 00000000000000000000000000010111,
# Islemdeki instruction:0010001011110111
# ALU_input1:11111111111111111111111110101000, ALU_input2:11111111111111111111111111110111
# result:11111111111111111111111110100000  ALUControl:110 write_data=11111111111111111111111110100000 write_reg=011
#
# Time:420, clk:1,
# --Sinyaller--
# regWrite:1, AluSrc:1, RegDst:0, MemtoReg:0, MemRead:0, MemWrite:0, Branch:0
# ------------
# Program Counter:00000000000000000000000000010111,Next Program Counter 00000000000000000000000000011000,
# Islemdeki instruction:0011101001011000
# ALU_input1:11111111111111111111111110101000, ALU_input2:00000000000000000000000000011000
# result:11111111111111111111111110111000  ALUControl:111 write_data=11111111111111111111111110111000 write_reg=001
#
# Time:440, clk:1,
# --Sinyaller--
# regWrite:1, AluSrc:1, RegDst:0, MemtoReg:0, MemRead:0, MemWrite:0, Branch:0
# ------------
# Program Counter:00000000000000000000000000011000,Next Program Counter 00000000000000000000000000011001,
# Islemdeki instruction:0011001011001010
# ALU_input1:11111111111111111111111110111000, ALU_input2:00000000000000000000000000001010
# result:11111111111111111111111110111010  ALUControl:111 write_data=11111111111111111111111110111010 write_reg=011
#
# Time:460, clk:1,
# --Sinyaller--
# regWrite:1, AluSrc:1, RegDst:0, MemtoReg:0, MemRead:0, MemWrite:0, Branch:0
# ------------
# Program Counter:00000000000000000000000000011001,Next Program Counter 00000000000000000000000000011010,
# Islemdeki instruction:0100101001011000
# ALU_input1:11111111111111111111111110101000, ALU_input2:00000000000000000000000000011000
# result:00000000000000000000000001000111  ALUControl:101 write_data=00000000000000000000000001000111 write_reg=001
#
# Time:480, clk:1,
# --Sinyaller--
# regWrite:1, AluSrc:1, RegDst:0, MemtoReg:0, MemRead:0, MemWrite:0, Branch:0
# ------------
# Program Counter:00000000000000000000000000011010,Next Program Counter 00000000000000000000000000011011,
# Islemdeki instruction:0100001011100010
# ALU_input1:00000000000000000000000001000111, ALU_input2:11111111111111111111111111100010
# result:00000000000000000000000000011000  ALUControl:101 write_data=00000000000000000000000000011000 write_reg=011
#
```



```
# Time:500, clk:1,
# --Sinyaller--
# regWrite:1, AluSrc:1, RegDst:0, MemtoReg:0, MemRead:0, MemWrite:0, Branch:0
# ------------
# Program Counter:00000000000000000000000000011011,Next Program Counter 00000000000000000000000000011100,
# Islemdeki instruction:0111001100011000
# ALU_input1:00000000000000000000000001000111, ALU_input2:00000000000000000000000000011000
# result:00000000000000000000000000000000  ALUControl:100 write_data=00000000000000000000000000000000 write_reg=100
#
# Time:520, clk:1,
# --Sinyaller--
# regWrite:1, AluSrc:1, RegDst:0, MemtoReg:0, MemRead:0, MemWrite:0, Branch:0
# ------------
# Program Counter:00000000000000000000000000011100,Next Program Counter 00000000000000000000000000011101,
# Islemdeki instruction:0111100101000010
# ALU_input1:00000000000000000000000000000000, ALU_input2:00000000000000000000000000000010
# result:00000000000000000000000000000001  ALUControl:100 write_data=00000000000000000000000000000001 write_reg=101
#
# Time:540, clk:1,
# --Sinyaller--
# regWrite:0, AluSrc:1, RegDst:0, MemtoReg:0, MemRead:0, MemWrite:1, Branch:0
# ------------
# Program Counter:00000000000000000000000000011101,Next Program Counter 00000000000000000000000000011110,
# Islemdeki instruction:1001101111000111
# ALU_input1:00000000000000000000000000000001, ALU_input2:00000000000000000000000000000111
# result:00000000000000000000000000001000  ALUControl:000 write_data=00000000000000000000000000001000 write_reg=111
#
# Time:560, clk:1,
# --Sinyaller--
# regWrite:1, AluSrc:1, RegDst:0, MemtoReg:1, MemRead:1, MemWrite:0, Branch:0
# ------------
# Program Counter:00000000000000000000000000011110,Next Program Counter 00000000000000000000000000011111,
# Islemdeki instruction:1000101001000111
# ALU_input1:00000000000000000000000000000001, ALU_input2:00000000000000000000000000000111
# result:00000000000000000000000000001000  ALUControl:000 write_data=00000000000000000000000001010111 write_reg=001
#
# ** Note: $finish    : C:/altera/13.1/workspace/mips_16/mips_testbench.v(13)
#    Time: 580 ps  Iteration: 0  Instance: /mips_testbench
```

registers.mem:

00000000000000000000000000000000

00000000000000000000000000001010

00000000000000000000000000010010

00000000000000000000000000000011

00000000000000000000000000000100

00000000000000000000000000100101

00000000000000001100000011111101

00000000000000000000000001000111

res_registers.mem:

00000000000000000000000000000000

00000000000000000000000001010111

00000000000000000000000001010111

00000000000000000000000000011000

00000000000000000000000000000000

00000000000000000000000000000001

00000000000000000000000001010101

00000000000000000000000001010111

Not: There is also the res_data.mem file for data memory.