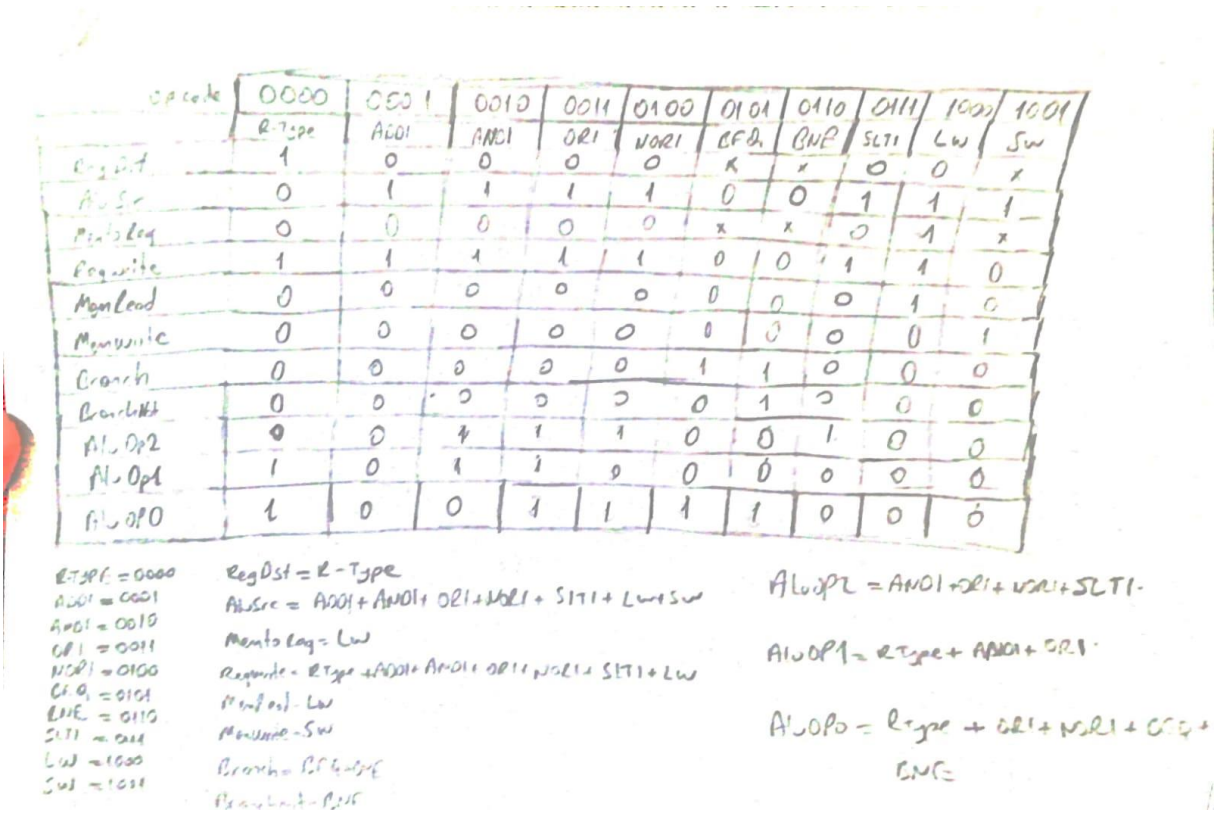
Önemli Not : Daha iyi anlatabilmek amacıyla Türkçe yazıyorum hocam şöyle bir sıkıntı mevcut normalde projeyi çalıştırdığım zaman çalıştı fakat qar dosyasına çevirip , qar dosyasından projeyi denemek için çalıştırdığım zaman sonuçlar hep xxxxxxxxx olarak geldi . Simulation modelsim içindeki register ve data dosyasınada xxxx olarak yazıldı. Çözümünü bulamadım hocam şöyle bir şey yaptım. Simulation modelsim içinde xxxxx yazdıktan sonra ilk değerleri (register ve data dosyasında sizin bize gönderdiğiniz değerler) register ve data dosyasına bir daha atıp çalıştırdım bu sefer çalıştı. Bu yüzden size projeyi atarken qar dosyayı yanında qar dosyasına basınca oluşan dosyalarıda attım şuanda bu haliyle çalışıyor. Bu şekilde deneyebilirsiniz hocam yada attığım dosyaları silip sadece qar dosyasını bir kez modelsimde çalıştırdıktan sonra oluşan dosyalarda simulation modelsim içindeki değerleri attığım (sizinde bize ilk olarak verdiğiniz register ve data dosyalarının) değerlerle güncelledikten sonra simulasyonda 2. Denemede çalışıyor hocam. Sebebini bilmiyorum o yüzden bir çözümde bulamadım bu şekilde attım özür diliyorum. Umarım iyi anlatabilmişimdir.

SIGNAL TABLE



Opcode	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001
	R-Type	ADDI	ANDI	ORI	NORI	CFQ	BNE	SLTI	LW	SW
RegDst	1	0	0	0	0	x	x	0	0	x
ALUSrc	0	1	1	1	1	0	0	1	1	1
MemtoReg	0	0	0	0	0	x	x	0	1	x
RegWrite	1	1	1	1	1	0	0	1	1	0
MemRead	0	0	0	0	0	0	0	0	1	0
MemWrite	0	0	0	0	0	0	0	0	0	1
Branch	0	0	0	0	0	1	1	0	0	0
BranchNE	0	0	0	0	0	0	1	0	0	0
ALUOp2	0	0	1	1	1	0	0	1	0	0
ALUOp1	1	0	1	1	0	0	0	0	0	0
ALUOP0	1	0	0	1	1	1	1	0	0	0

RTYPE = 0000
ADDI = 0001
ANDI = 0010
ORI = 0011
NORI = 0100
CFQ = 0101
BNE = 0110
SLTI = 0111
LW = 1000
SW = 1001

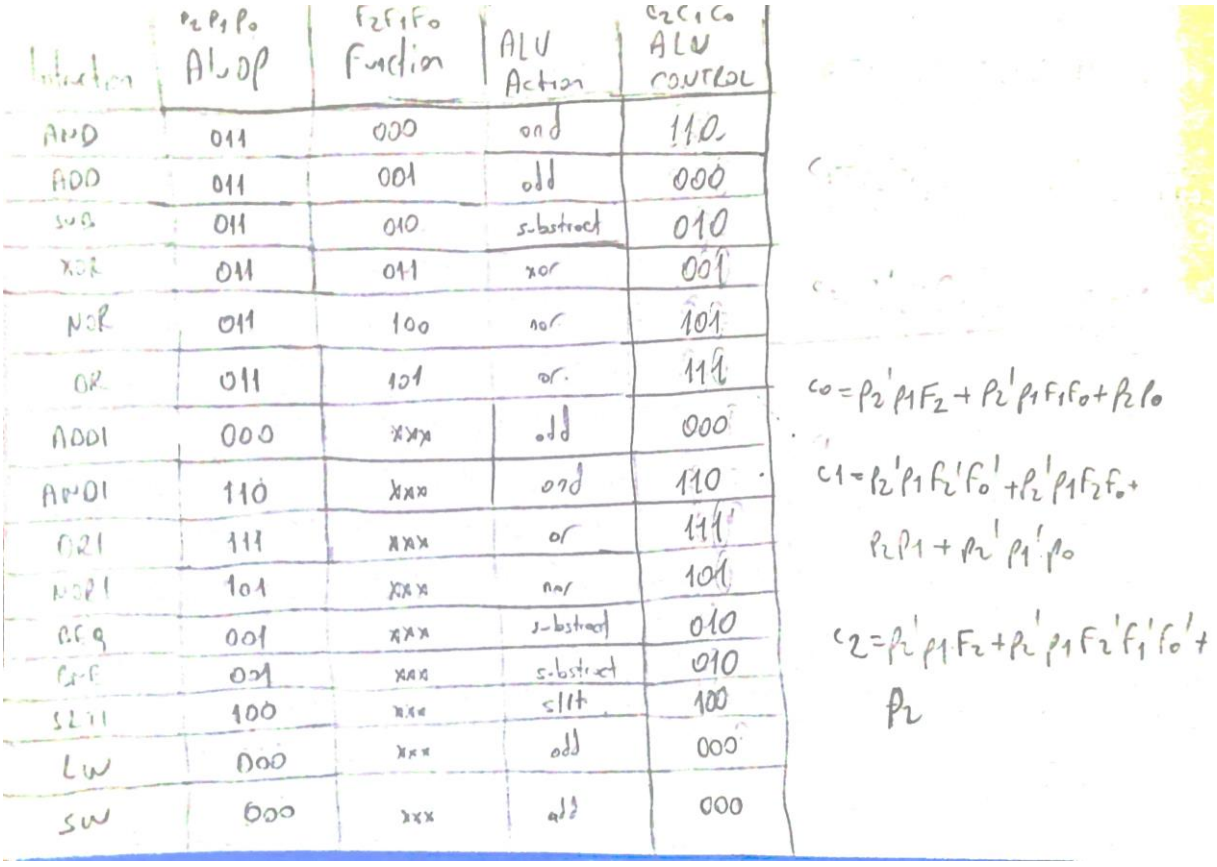
RegDst = R-Type
ALUSrc = ADDI + ANDI + ORI + NORI + SLTI + LW + SW
MemtoReg = LW
RegWrite = RType + ADDI + ANDI + ORI + NORI + SLTI + LW
MemRead = LW
MemWrite = SW
Branch = CFQ + BNE
BranchNE = BNE

ALUOP2 = ANDI + ORI + NORI + SLTI

ALUOP1 = RType + ANDI + ORI

ALUOP0 = RType + ORI + NORI + CFQ + BNE

ALU CONTROL TABLE



Instruction	$P_2P_1P_0$ ALUOp	$F_2F_1F_0$ Function	ALU Action	$C_2C_1C_0$ ALU CONTROL
AND	011	000	and	110
ADD	011	001	add	000
SUB	011	010	substract	010
XOR	011	011	xor	001
NOR	011	100	nor	101
OR	011	101	or	111
ADDI	000	xxx	add	000
ANDI	110	xxx	and	110
ORI	111	xxx	or	111
NORI	101	xxx	nor	101
SUBI	001	xxx	substract	010
SLTI	001	xxx	substract	010
SLTI	100	xxx	slt	100
LW	000	xxx	add	000
SW	000	xxx	add	000

$C_0 = P_2'P_1F_2 + P_2'P_1F_1F_0 + P_2P_0$

$C_1 = P_2'P_1F_2'F_0' + P_2'P_1F_2F_0 + P_2P_1 + P_2'P_1'P_0$

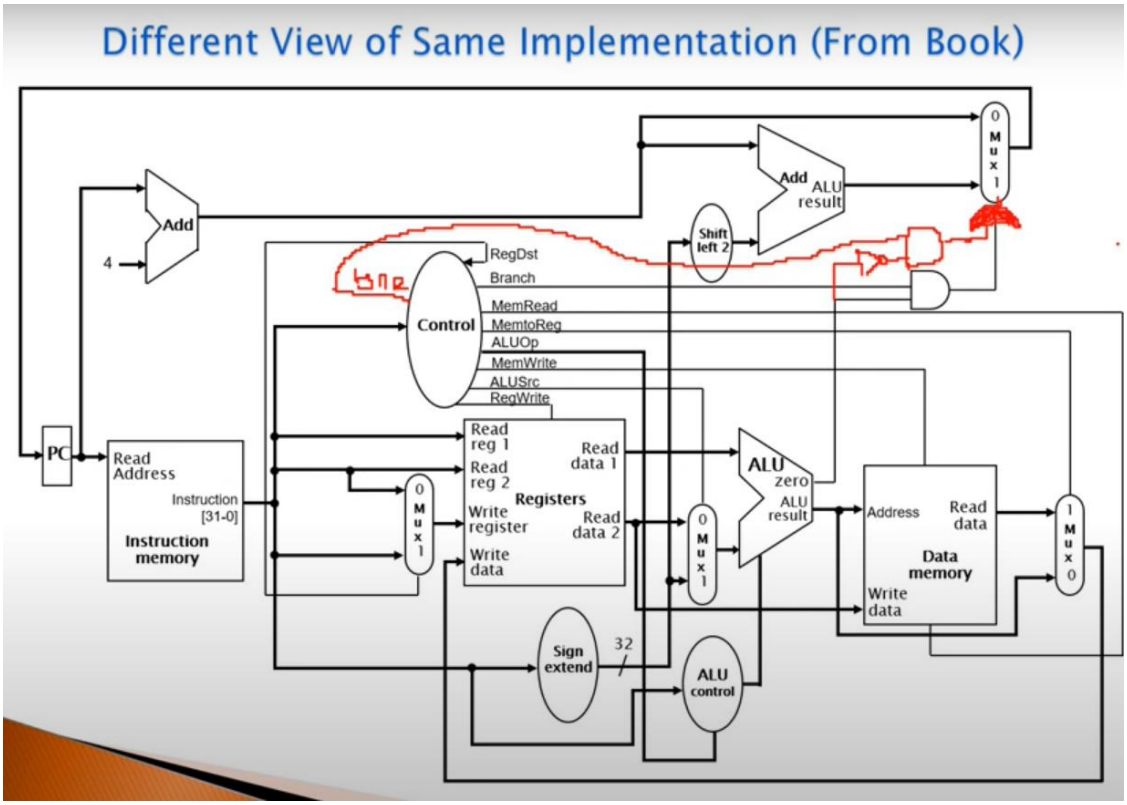
$C_2 = P_2'P_1F_2 + P_2'P_1F_2'F_1'F_0' + P_2$

Missing Parts

Branch , Branch not equal is not working.I tested Instruction memory module but when i add instruction memory to the system it was working but there is clk , finish , delay conflict and i did not branch instruction so that i delete it from the system but it is working i tested it and i put it on to the pdf , 32 bit comparator shift left 2bit module instruction memory module is working.

Due to the clk delay finish problem, I could not test the instructions sequentially and show you the change of the register, because the program had problems stopping and stopping in the wrong place. Maybe because I couldn't set the clk finish delay exactly, that's why I could only put the final versions, but before that, I showed that all the instructions were working one by one, and the data memory of the register was working.

MIPS SYSTEM



CONTROL UNIT TEST

```
sim:/control_testbench/ALUOp
VSIM 5> step -current
# opcode=0, MemRead=0, MemToReg=0, MemWrite=0, RegWrite=1, AluOp=3,RegDst=1,AluSrc=0,Branch=0,BranchNot=0
# opcode=1, MemRead=0, MemToReg=0, MemWrite=0, RegWrite=1, AluOp=0,RegDst=0,AluSrc=1,Branch=0,BranchNot=0
# opcode=2, MemRead=0, MemToReg=0, MemWrite=0, RegWrite=1, AluOp=6,RegDst=0,AluSrc=1,Branch=0,BranchNot=0
# opcode=3, MemRead=0, MemToReg=0, MemWrite=0, RegWrite=1, AluOp=7,RegDst=0,AluSrc=1,Branch=0,BranchNot=0
# opcode=4, MemRead=0, MemToReg=0, MemWrite=0, RegWrite=1, AluOp=5,RegDst=0,AluSrc=1,Branch=0,BranchNot=0
# opcode=5, MemRead=0, MemToReg=0, MemWrite=0, RegWrite=0, AluOp=1,RegDst=0,AluSrc=0,Branch=1,BranchNot=0
# opcode=6, MemRead=0, MemToReg=0, MemWrite=0, RegWrite=0, AluOp=1,RegDst=0,AluSrc=0,Branch=1,BranchNot=1
# opcode=7, MemRead=0, MemToReg=0, MemWrite=0, RegWrite=1, AluOp=4,RegDst=0,AluSrc=1,Branch=0,BranchNot=0
# opcode=8, MemRead=1, MemToReg=1, MemWrite=0, RegWrite=1, AluOp=0,RegDst=0,AluSrc=1,Branch=0,BranchNot=0
# opcode=9, MemRead=0, MemToReg=0, MemWrite=1, RegWrite=0, AluOp=0,RegDst=0,AluSrc=1,Branch=0,BranchNot=0

VSIM 6>
```

ALU CONTROL TEST

```
initial begin
    ALUOp = 3'b011;
    functionfield=3'b000;
    #`DELAY;
    ALUOp = 3'b011;
    functionfield=3'b001;
    #`DELAY;
    ALUOp = 3'b011;
    functionfield=3'b010;
    #`DELAY;
    ALUOp = 3'b011;
    functionfield=3'b011;
    #`DELAY;
    ALUOp = 3'b011;
    functionfield=3'b100;
    #`DELAY;
    ALUOp = 3'b011;
    functionfield=3'b101;
    #`DELAY;
    ALUOp = 3'b000;
    functionfield=3'bxxx;
    #`DELAY;
    ALUOp = 3'b110;
    functionfield=3'bxxx;
```

Pano Resim Araçlar

Library sim

Transcript

```
# Loading work.ALU_control_testbench
# Loading work.ALU_control
add wave -position insertpoint  \
sim:/ALU_control_testbench/ALUOp \
sim:/ALU_control_testbench/functionfield \
sim:/ALU_control_testbench/ALUControlOp
VSIM 5> step -current
# ALUOp=3, functionfiled=0,ALUControlOp=6
# ALUOp=3, functionfiled=1,ALUControlOp=0
# ALUOp=3, functionfiled=2,ALUControlOp=2
# ALUOp=3, functionfiled=3,ALUControlOp=1
# ALUOp=3, functionfiled=4,ALUControlOp=5
# ALUOp=3, functionfiled=5,ALUControlOp=7
# ALUOp=0, functionfiled=x,ALUControlOp=0
# ALUOp=6, functionfiled=x,ALUControlOp=6
# ALUOp=7, functionfiled=x,ALUControlOp=7
# ALUOp=5, functionfiled=x,ALUControlOp=5
```

Instruction Memory Test

```
m:/instruction_memory_testbench/PC
M 6> step -current
Inst 001,Opcode=0000, Rs=000, Rt=001, Imm=010000
Inst 010,Opcode=0000, Rs=011, Rt=111, Imm=000000
Inst 011,Opcode=0000, Rs=001, Rt=011, Imm=101100
Inst 100,Opcode=0000, Rs=101, Rt=010, Imm=101101
Inst 101,Opcode=0000, Rs=101, Rt=011, Imm=100001
Inst 110,Opcode=0000, Rs=101, Rt=001, Imm=101011
Inst 111,Opcode=0000, Rs=101, Rt=001, Imm=101011
Inst 000,Opcode=0000, Rs=111, Rt=101, Imm=101001
```

Dosya Düzen Biçim

```
0000000001010000
0000011111000000
0000001011101100
0000101010101101
0000101011100001
0000101001101011
0000101001101011
0000111101101001
```

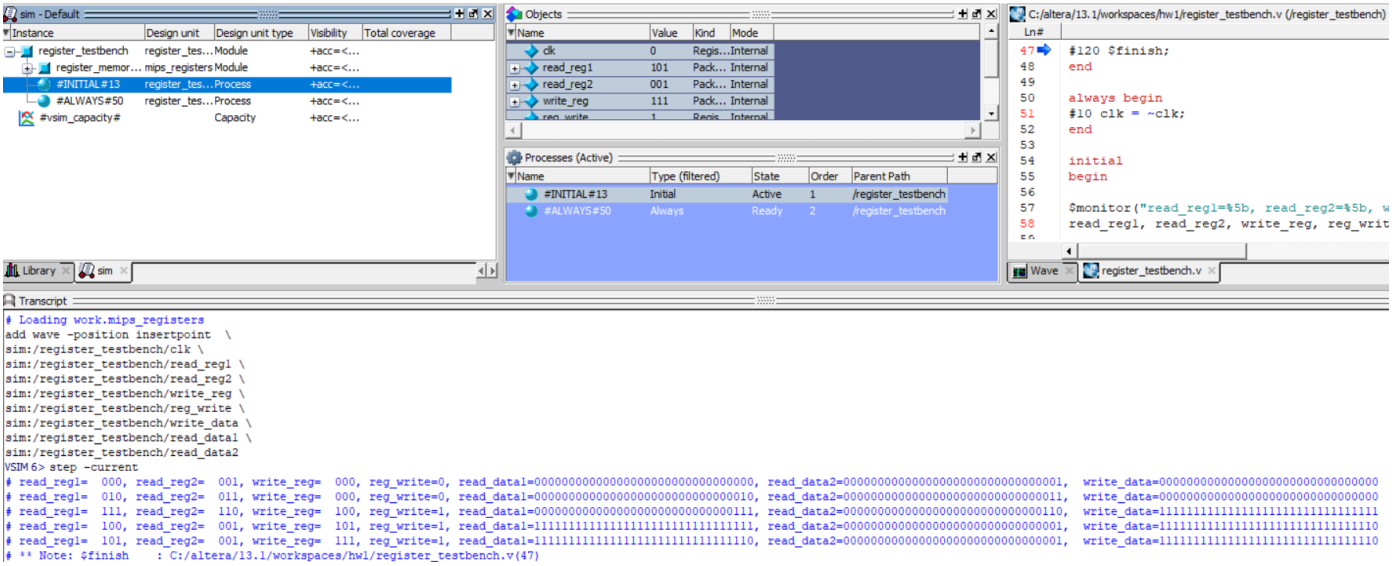
Comparator 32_bit test

```
add wave -position insertpoint  \
sim:/comparator_32bit_testbench/input1 \
sim:/comparator_32bit_testbench/input2 \
sim:/comparator_32bit_testbench/gt \
sim:/comparator_32bit_testbench/eq
VSIM 6> step -current
# input1=  858993459, input2=  858993459, gt=0, eq=1
# input1= 1145324612, input2=  572662306, gt=1, eq=0
```

Shift left 2bit test

```
add wave -position insertpoint  \
sim:/left_shifter_2bit_testbench/input1 \
sim:/left_shifter_2bit_testbench/output1
/SIM 6> step -current
# input=00110011001100110011001100110011, output=11001100110011001100110011001100,
# input=01000100010001000100010001000100, output=00010001000100010001000100010000,
```

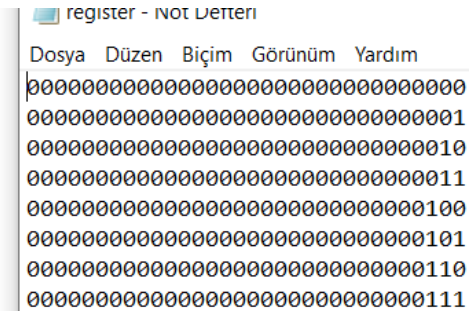
REGISTER MODULE TEST



MIPS TEST RESULT

Before And instruction

```
initial begin
clk = 1;
PC = 0;

//AND
instruction_set = 16'b0000000001010000;
#`DELAY;
instruction_set = 16'b0000011111000000;
#`DELAY;

//ADD
```



After and instruction

```
add wave -position insertpoint  \
sim:/MiniMIPS_testbench/clk \
sim:/MiniMIPS_testbench/PC \
sim:/MiniMIPS_testbench/instruction_set \
sim:/MiniMIPS_testbench/result
VSIM 5> step -current
# instruction_set=0000000001010000, result=00000000000000000000000000000000
# instruction_set=0000011111000000, result=00000000000000000000000000000011
```

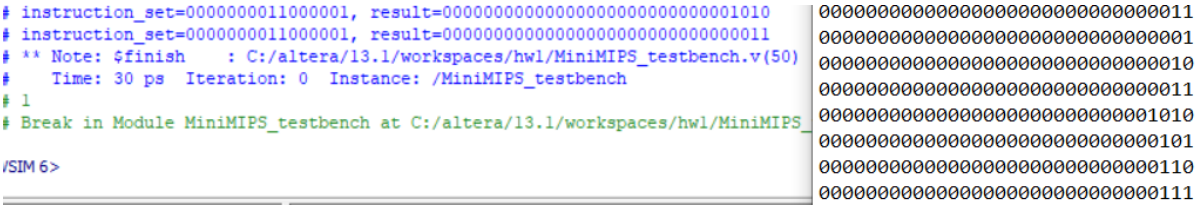
```
00000000000000000000000000000011
00000000000000000000000000000001
00000000000000000000000000000000
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000111
```

Before add instruction

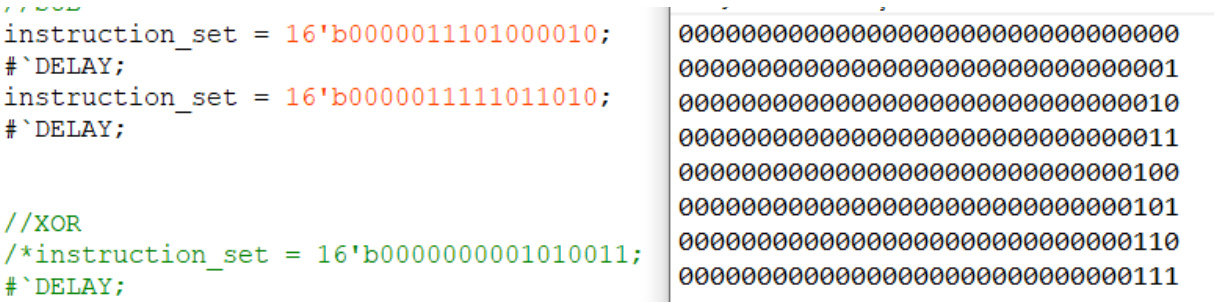
```
//ADD
instruction_set = 16'b0000111011100001;
#`DELAY;
instruction_set = 16'b0000000011000001;
#`DELAY;

//SUB
/*instruction_set = 16'b0000000001010010;
```

```
00000000000000000000000000000000
00000000000000000000000000000001
00000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000111
```

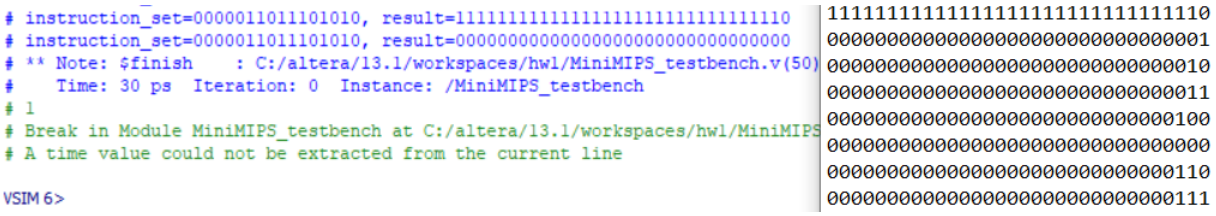
After add instruction



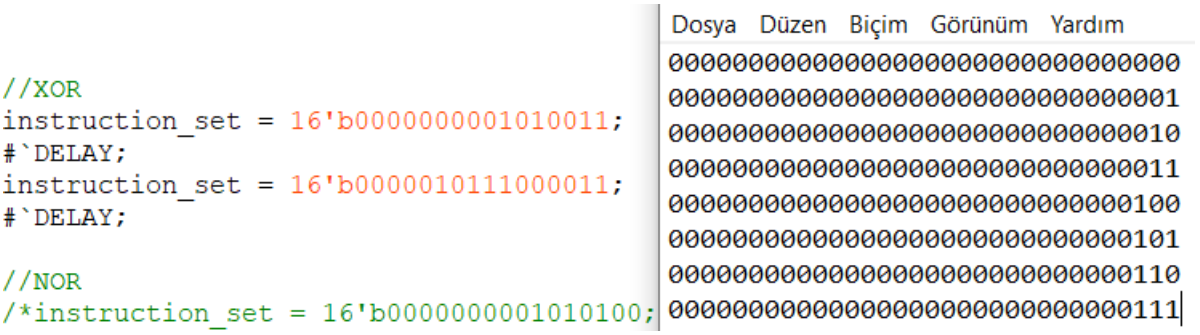
Before Sub instruction



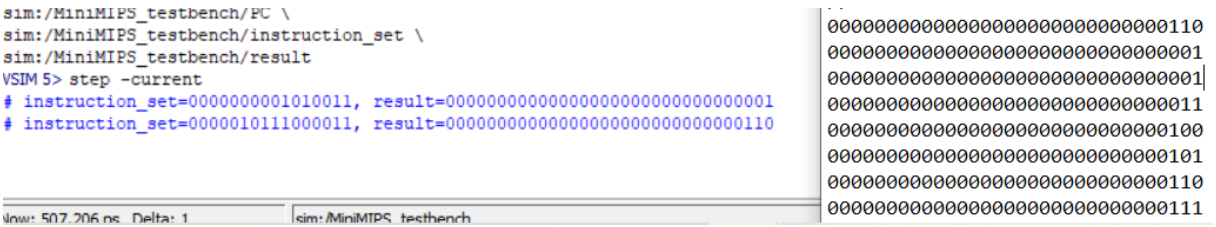
After sub instruction



Before Xor instruction



After Xor instruction



Before Nor instruction

```
//NOR
instruction_set = 16'b0000101001010100;
#`DELAY;
instruction_set = 16'b0000011111000100;
#`DELAY;

//OR
/*instruction_set = 16'b0000000001010101;
```

```
00000000000000000000000000000000
00000000000000000000000000000001
00000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000111
```

After Nor instruction

```
add wave -position insertpoint  \
sim:/MiniMIPS_testbench/clk \
sim:/MiniMIPS_testbench/PC \
sim:/MiniMIPS_testbench/instruction_set \
sim:/MiniMIPS_testbench/result
VSIM 5> step -current
# instruction_set=0000101001010100, result=11111111111111111111111111111010
# instruction_set=0000011111000100, result=11111111111111111111111111111000
```

```
11111111111111111111111111111000
00000000000000000000000000000001
11111111111111111111111111111010
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000111
```

Before or instruction

```
#`DELAY;*/

//OR
instruction_set = 16'b0000000001010101;
#`DELAY;
instruction_set = 16'b0000011111000101;
#`DELAY;

//#10 $finish;
end
```

```
Dosya  Düzen  Biçim  Görünüm  Yardım
00000000000000000000000000000000
00000000000000000000000000000001
00000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000111
```

After or instruction

```
sim:/MiniMIPS_testbench/clk \
sim:/MiniMIPS_testbench/PC \
sim:/MiniMIPS_testbench/instruction_set \
sim:/MiniMIPS_testbench/result
/SIM 5> step -current
# instruction_set=0000000001010101, result=00000000000000000000000000000001
# instruction_set=0000011111000101, result=00000000000000000000000000000111
```

```
00000000000000000000000000000111
00000000000000000000000000000001
00000000000000000000000000000001
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000111
```

Before ADDI instruction

```
//ADDI
instruction_set = 16'b0001000010000001;
#`DELAY;
instruction_set = 16'b0001011111000101;
#`DELAY;

//ANDI
/*instruction_set = 16'b0000000001010101;
```

```
00000000000000000000000000000000
00000000000000000000000000000001
00000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000111
```

After ADDI instruction

```
sim:/MiniMIPS_testbench/PC \
sim:/MiniMIPS_testbench/instruction_set \
sim:/MiniMIPS_testbench/result
VSIM 5> step -current
# instruction_set=0001000010000001, result=00000000000000000000000000000001
# instruction_set=0001011111000101, result=00000000000000000000000000001000
```

```
00000000000000000000000000000000
00000000000000000000000000000001
00000000000000000000000000000001
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000001000
```

Before Andi instruction

```
//ANDI
instruction_set = 16'b0010000001010101;
#`DELAY;
instruction_set = 16'b0010011111000101;
#`DELAY;

//ORI
/*instruction_set = 16'b0000000001010101;
```

```
00000000000000000000000000000000
00000000000000000000000000000001
00000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000111
```

After Andi instruction

```
sim:/MiniMIPS_testbench/clk \
sim:/MiniMIPS_testbench/PC \
sim:/MiniMIPS_testbench/instruction_set \
sim:/MiniMIPS_testbench/result
VSIM 5> step -current
# instruction_set=0010000001010101, result=00000000000000000000000000000000
# instruction_set=0010011111000101, result=00000000000000000000000000000001
```

```
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000001
```

Before Ori instruction

```
//ORI
instruction_set = 16'b0011000001010101;
#`DELAY;
instruction_set = 16'b0011011111000101;
#`DELAY;

//NORI
/*instruction_set = 16'b0000000001010101;
#`DELAY;
```

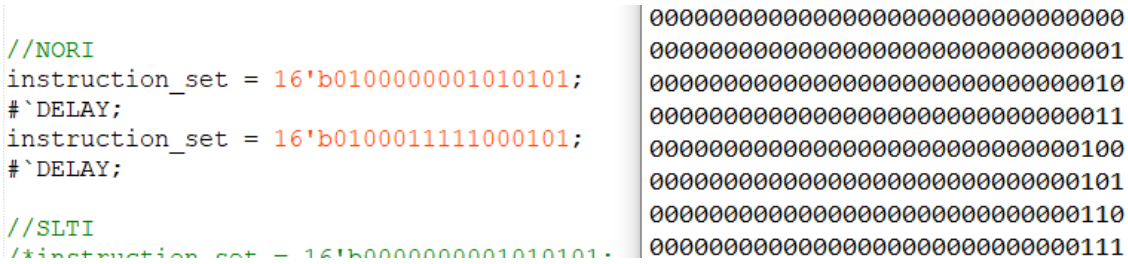
```
00000000000000000000000000000000
00000000000000000000000000000001
00000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000111
```

After Ori instruction

```
sim:/MiniMIPS_testbench/PC \
sim:/MiniMIPS_testbench/instruction_set \
sim:/MiniMIPS_testbench/result
VSIM 5> step -current
# instruction_set=0011000001010101, result=00000000000000000000000000010101
# instruction_set=0011011111000101, result=00000000000000000000000000000111
```

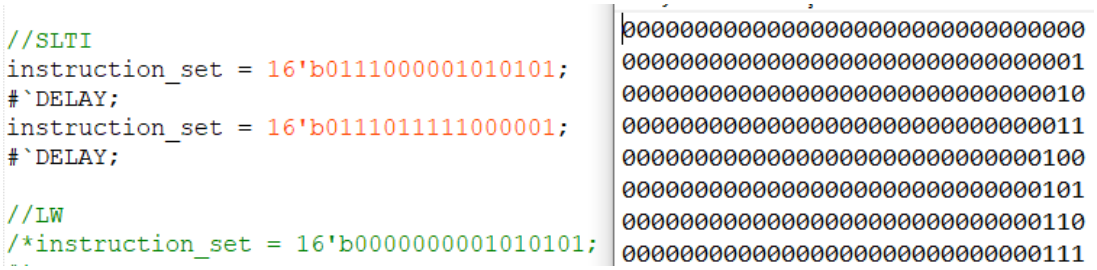
```
00000000000000000000000000000000
00000000000000000000000000010101
00000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000111
```

Before Nori instruction

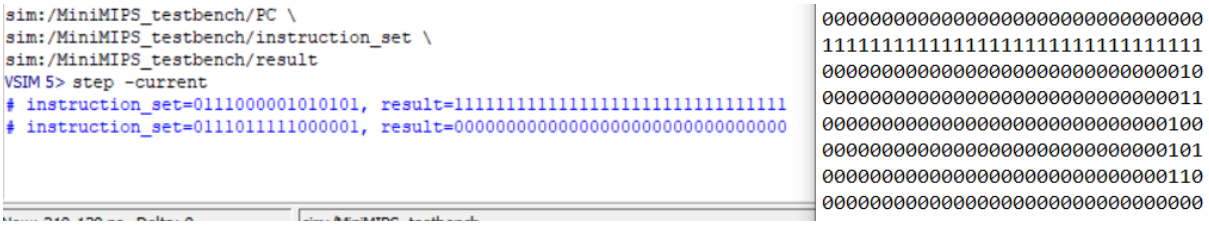


After Nori instruction

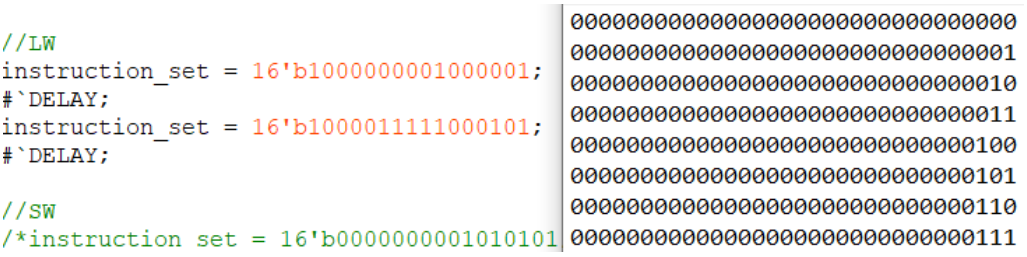
Before Slti instruction



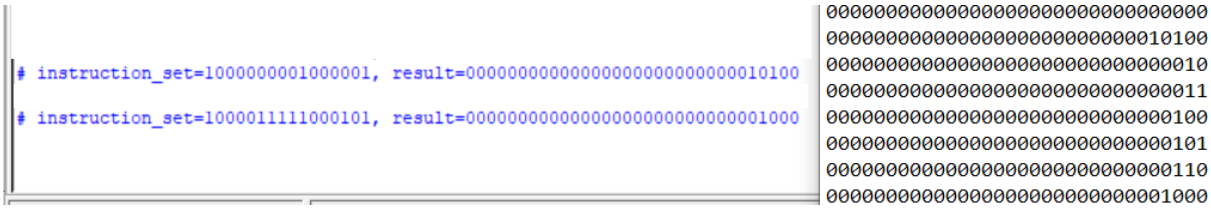
After Slti instruction



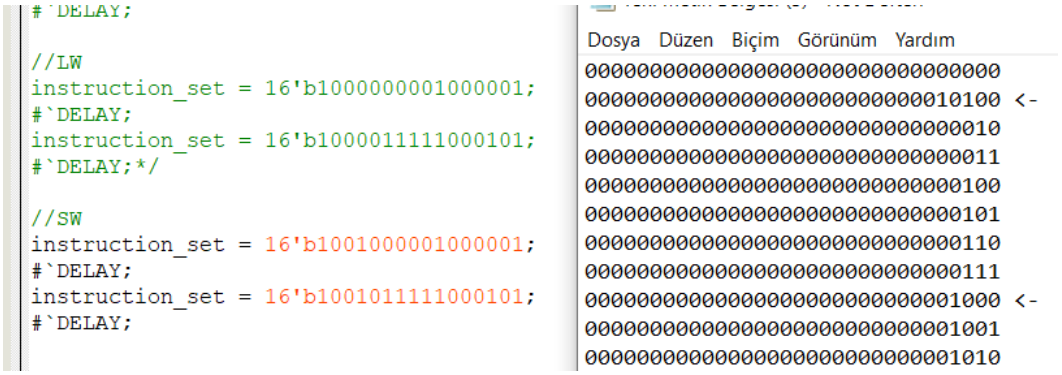
Before LW instruction



After LW instruction



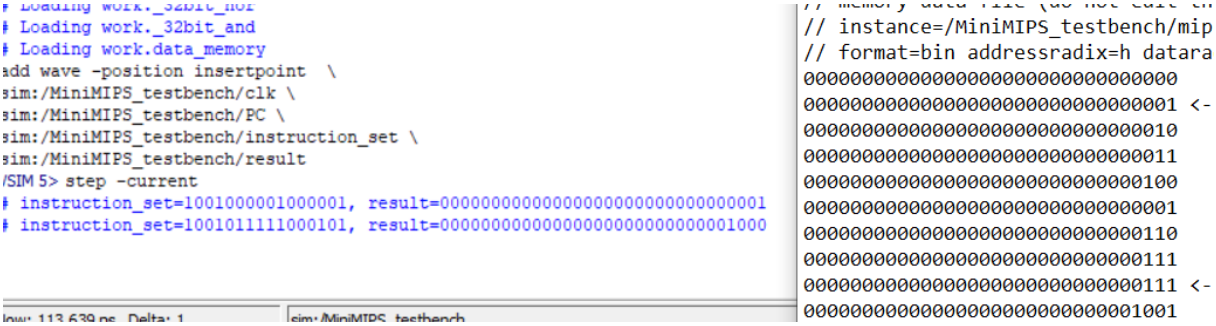
Before SW instruction



```
#`DELAY;

//LW
instruction_set = 16'b1000000001000001;
#`DELAY;
instruction_set = 16'b1000011111000101;
#`DELAY;*/

//SW
instruction_set = 16'b1001000001000001;
#`DELAY;
instruction_set = 16'b1001011111000101;
#`DELAY;
```

```
Dosya  Düzen  Biçim  Görünüm  Yardım
00000000000000000000000000000000
00000000000000000000000000010100 <-
00000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000111
00000000000000000000000000001000 <-
00000000000000000000000000001001
00000000000000000000000000001010
```

After SW instruction

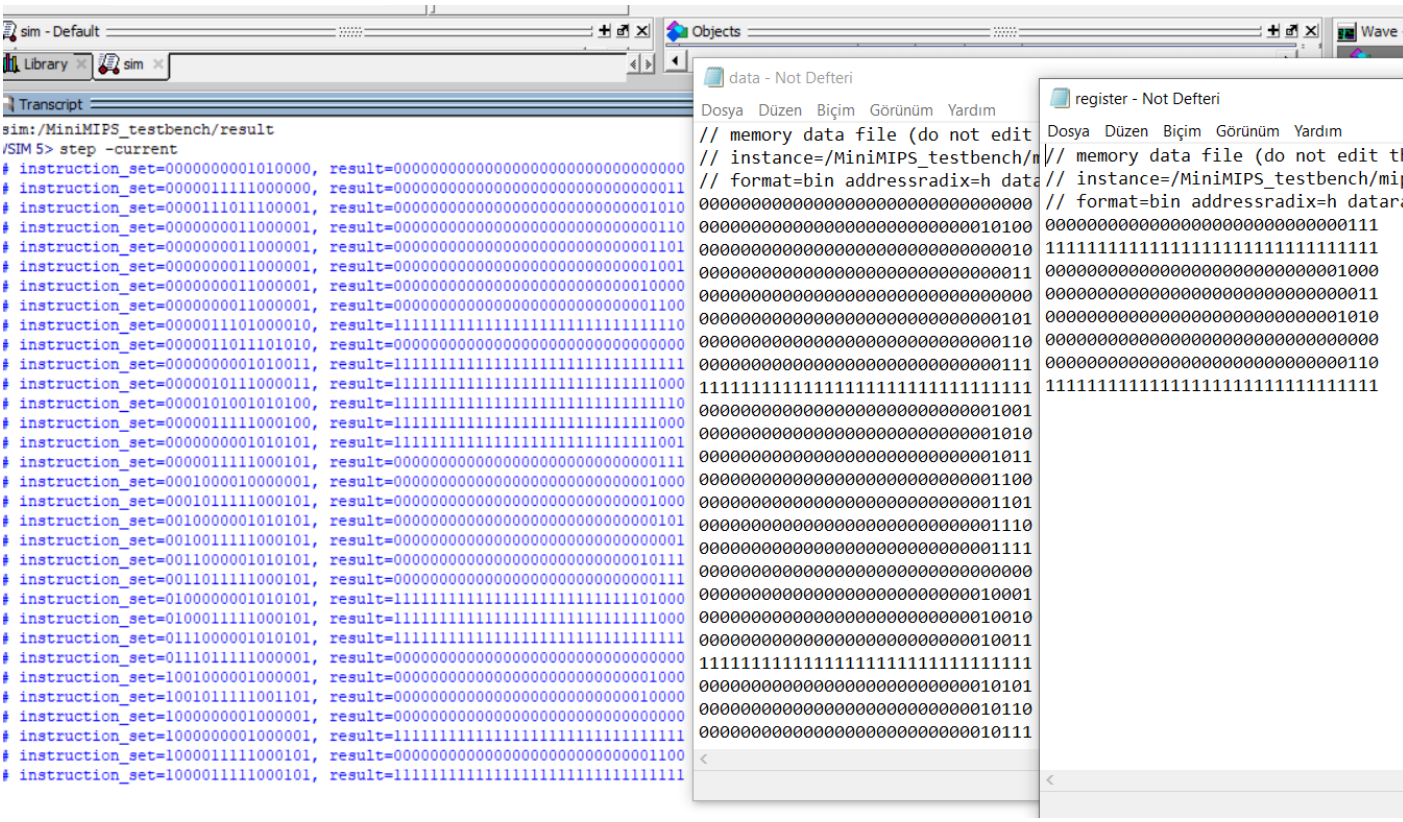


```
# Loading work._32bit_and
# Loading work.data_memory
add wave -position insertpoint  \
sim:/MiniMIPS_testbench/clk \
sim:/MiniMIPS_testbench/PC \
sim:/MiniMIPS_testbench/instruction_set \
sim:/MiniMIPS_testbench/result
VSIM 5> step -current
# instruction_set=1001000001000001, result=00000000000000000000000000000001
# instruction_set=1001011111000101, result=00000000000000000000000000001000
```

```
// instance=/MiniMIPS_testbench/mip
// format=bin addressradix=h datara
00000000000000000000000000000000
00000000000000000000000000000001 <-
00000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000000100
00000000000000000000000000000001
00000000000000000000000000000110
00000000000000000000000000000111
00000000000000000000000000000111 <-
00000000000000000000000000001001
```

All Mıps Instruction and result

Here last instruction is sw but it working two times one instruction (i dont know why) but results is second and last one .



```
sim - Default
Library  sim
Transcript
sim:/MiniMIPS_testbench/result
VSIM 5> step -current
# instruction_set=0000000001010000, result=00000000000000000000000000000000
# instruction_set=0000011111000000, result=00000000000000000000000000000011
# instruction_set=0000111011100001, result=00000000000000000000000000001010
# instruction_set=0000000011000001, result=00000000000000000000000000000110
# instruction_set=0000000011000001, result=00000000000000000000000000001101
# instruction_set=0000000011000001, result=00000000000000000000000000001001
# instruction_set=0000000011000001, result=00000000000000000000000000010000
# instruction_set=0000000011000001, result=00000000000000000000000000001100
# instruction_set=0000011101000010, result=11111111111111111111111111111110
# instruction_set=0000011011101010, result=00000000000000000000000000000000
# instruction_set=0000000001010011, result=11111111111111111111111111111111
# instruction_set=0000010111000011, result=11111111111111111111111111111000
# instruction_set=0000101001010100, result=11111111111111111111111111111110
# instruction_set=0000011111000100, result=11111111111111111111111111111000
# instruction_set=0000000001010101, result=11111111111111111111111111111001
# instruction_set=0000011111000101, result=00000000000000000000000000000111
# instruction_set=0001000010000001, result=00000000000000000000000000001000
# instruction_set=0001011111000101, result=00000000000000000000000000001000
# instruction_set=0010000001010101, result=00000000000000000000000000000101
# instruction_set=0010011111000101, result=00000000000000000000000000000001
# instruction_set=0011000001010101, result=00000000000000000000000000010111
# instruction_set=0011011111000101, result=00000000000000000000000000000111
# instruction_set=0100000001010101, result=11111111111111111111111111101000
# instruction_set=0100011111000101, result=11111111111111111111111111111000
# instruction_set=0111000001010101, result=11111111111111111111111111111111
# instruction_set=0111011111000001, result=00000000000000000000000000000000
# instruction_set=1001000001000001, result=00000000000000000000000000001000
# instruction_set=1001011111001101, result=00000000000000000000000000010000
# instruction_set=1000000001000001, result=00000000000000000000000000000000
# instruction_set=1000000001000001, result=11111111111111111111111111111111
# instruction_set=1000011111000101, result=00000000000000000000000000001100
# instruction_set=1000011111000101, result=11111111111111111111111111111111
```

```
Objects
data - Not Defteri
Dosya  Düzen  Biçim  Görünüm  Yardım
// memory data file (do not edit
// instance=/MiniMIPS_testbench/m
// format=bin addressradix=h data
00000000000000000000000000000000
00000000000000000000000000010100
00000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000000000
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000111
11111111111111111111111111111111
00000000000000000000000000001001
00000000000000000000000000001010
00000000000000000000000000001011
00000000000000000000000000001100
00000000000000000000000000001101
00000000000000000000000000001110
00000000000000000000000000001111
00000000000000000000000000000000
00000000000000000000000000010001
00000000000000000000000000010010
00000000000000000000000000010011
11111111111111111111111111111111
00000000000000000000000000010101
00000000000000000000000000010110
00000000000000000000000000010111
```

```
register - Not Defteri
Dosya  Düzen  Biçim  Görünüm  Yardım
// memory data file (do not edit th
// instance=/MiniMIPS_testbench/mip
// format=bin addressradix=h datara
00000000000000000000000000000111
11111111111111111111111111111111
00000000000000000000000000001000
00000000000000000000000000000011
00000000000000000000000000001010
00000000000000000000000000000000
00000000000000000000000000000110
11111111111111111111111111111111
```